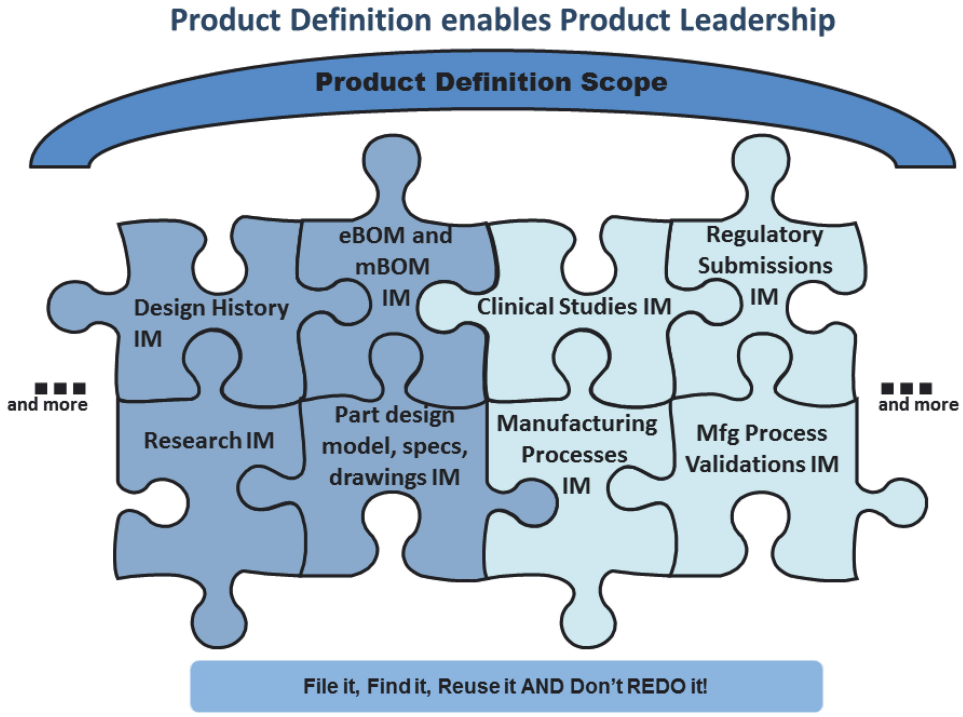
SE | RWTH AACHEN UNIVERSITY

Dan McKay Matheson

SAMEM: A Methodology for the Elicitation and Specification of Requirements for Agile Model-driven Engineering of Large Software Solutions



Aachener Informatik-Berichte,
Software Engineering

Band 38

Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe

SAMEM:
A Methodology for the Elicitation and Specification of Requirements for Agile Model-driven Engineering of Large Software Solutions

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Dan McKay Matheson,
Master of Science Computer Science
aus Des Moines, Iowa, USA

Berichter: Universitätsprofessor Dr. Bernhard Rumpe
Professor Dr. Michael Goedicke

Tag der mündlichen Prüfung: 18. Januar 2019

[Mat19] D. McKay Matheson:
SAMEM: A Methodology for the Elicitation and Specification of Requirements for Agile Model-driven Engineering of Large Software Solutions.
Shaker Verlag, ISBN 978-3-8440-6518-3. Aachener Informatik-Berichte, Software Engineering, Band 38. Februar 2019.
www.se-rwth.de/publications/

Kurzfassung

Die Software Agile Modeling and Engineering Methodology (SAMEM) ist eine pragmatische Methodologie zur Entwicklung von Lösungen, die darauf ausgerichtet ist, einen iterativen und auf Zuwachs bezogenen (incremental) Projektmanagement-Ansatz (agile) zu kombinieren mit dem Schwerpunkt auf Modellbildung von Projekt-Artefakten (Model-driven Development oder MDD). Die Modellbildung beginnt bei SAMEM am Anfang des Projekts mit den Lösungszielen und setzt sich fort in der Implementierungsarbeit. Zur effektiven und effizienten Kommunikation wird die Verwendung von graphischen oder visuellen Modellen zur Darstellung der Projekt-Artefakte maximiert Die Entwicklung der Artefakte erfolgt über viele kurze Zyklen (iterations) und den Aufbau in mehreren kleinen Schritten (increments).

Der iterative & incremental Ansatz ist eine Technik zur Risikominderung, um durch eine kurze Feedback-Schleife ein Scheitern des Projekts zu verhindern. Projekte haben eine Dynamik, die auf der Ansammlung von Entscheidungen und der geleisteten Arbeit beruht. Die Bereitschaft von Menschen, schlechte Entscheidungen oder mangelhafte Arbeit zuzugeben, die Arbeit, besonders nach erheblichen Investitionen, zu verwerfen und neu anzufangen, ist selten. Je umfangreicher die Arbeiten waren, die verworfen werden, desto schwieriger ist es, erneut anzufangen. Kleine Schritte mit der Überprüfung der Korrektheit der Arbeit und der Übereinstimmung mit den Projektzielen schaffen eine Situation, in der möglicherweise nur eine geringe Investition verworfen wird. Kleine Verwerfungen sind leichter zu akzeptieren und können oft als Designuntersuchungen gewertet werden.SAMEM wendet den iterative & incremental Ansatz auf allen Ebenen für die gesamte Lebensdauer des Projekts an.

Modellbildung oder graphische Representationen der Mehrheit der Projekt-Artefakte ist eine Risikominderung, die inkorrekte Lösung zu erschaffen und die Lösung inkorrekt zu erstellen. Es ist allgemein bekannt, dass visuelle Gestaltung Information in vielen Situationen genauer übermitteln kann als Text. Die bekannte Redewendung “Ein Bild sagt mehr als tausend Worte”, die auch in der kognitiven Psychologieforschung vertreten wird, ist die Basis für die Hervorhebung der Modellbildung. Die Auswirkungen der Risikominderung auf die Erstellung der inkorrekten Lösung, zumindest aus der Perspektive der Interessenvertreter, erfolgen durch die verbesserte und effektivere Kommunikation, die durch die Modellbildung oder graphische Representationen erbracht werden. Ein Beitrag zu graphischer Repräsentation oder Modellbildung ist die Idee einer Lösungs-Überblick-Zeichnung (Solution Overview Drawing, SOD), die aus einer Zusammenfassung auf einer einzigen Seite besteht, indem sie mehrfache Sichtweisen von Anforderungen oder Designspezifikationen integriert.

Zur Sicherstellung der Effizienz der Risikominderung der Techniken von iterative & incremental Projektmanagement und Modellbildung sind zusätzliche Komponenten erforderlich. Während SAMEM auf den gesamten Projektprozess anwendbar ist, besteht

eine Fokusierung auf die Anfangsphasen des Projekts. Die Motivation zur Fokusierung auf die Anfangsphasen des Projekts besteht in den allseits bekannten Kosten der Fehlerbehebung im Verlauf des Projekts. Ein weiterer Beitrag dieser Forschung besteht darin, einen Mechanismus hinzuzufügen, der die iteration & increment Arbeit von dem ursprünglichen Ausdruck der Lösungsmöglichkeiten oder Ziele zur abschließenden Implementierung führt. Der leitende Mechanismus durchquert die Abstraktionshierarchie von den anfänglichen vagen Ideen zur abschließenden Implementierung. Die Durchquerung ist eine Sammlung von Entscheidungen, wie zum Beispiel die Wahl von einem Lösungsziel gegenüber einem anderen, von einer Anforderung gegenüber einer anderen, von einem Lösungs-Architektur-Design gegenüber einem anderen und einer Implementierungs Technologie gegenüber einer anderen. Die Entscheidungen sind als Projekt-Artifakte repräsentiert und viele können gut als Modelle oder graphisch dargestellt werden. Der leitende Mechanismus für SAMEM ist das Open Distributed Processing – Reference Model ISO standard 10746 (RM-ODP).

Ein weiterer Beitrag, der sicherstellen soll, dass die Risikominderung von iterative & incremental Projektmanagement und Modellbildung effektiv ist, ist eine Gruppe von engineering Prinzipien, die zur Qualitätsüberprüfung der Arbeit beitragen. Diese Komponente von SAMEM ist eine Gruppe von Prinzipien, die während des ganzen Projekts angewendet werden kann zur Prüfung der Entscheidungen und Artifakte auf Richtigkeit und Qualität. Die Prinzipien sind auch von Nutzen bei der Stimulierung von Innovation durch Erklärungen von Annahmen und infrage stellenden Verhaltensweisen. Die Prinzipien sind im Hinblick auf Domänunabhängigkeit konzipiert, aber mit Fokus auf Software zentrierte Lösungen und daher der Name Software Engineering First Principles (SEFP).

Der Nachweis von SAMEM als effektive Methodologie setzt verschiedene Mittel ein. Empirischer Beweis, der durch Umfrage unter den Projektteilnehmern ermittelt wurde, die eine frühe Version von SAMEM in verschiedenen Industrieprojekten über mehrere Jahre hin einsetzten, ist die wichtigste Stütze. Kognitive Effektivitätsmaßnahmen werden bewertet im Hinblick auf die graphischen Kommunikations-Artifakte von SAMEM. Eine Bewertung von SAMEM mit der neuen und sich entwickelnden standardmäßigen Software Engineering Method and Theory (SEMAT) liefert einen Vergleich der Stärken und Unterschiede. Da SAMEM eine praktische und pragmatische Software Lösungsmethodologie sein soll, wird eine Diskussion über die erreichten geschäftlichen Vorteile in den Projekten präsentiert.

Abstract

The Software Agile Modeling and Engineering Methodology (SAMEM) is a pragmatic solution development methodology that focuses on combining an iterative & incremental project management approach (agile) with an emphasis on the modeling of project artifacts (Model-driven Development or MDD). The SAMEM starts the modeling at the beginning of the project with the solution goals and continues through the implementation work. For effective and efficient communication, the use of graphical or visual models is maximized for the representation of project artifacts. The development of the artifacts occurs over many short cycles (iterations) and they are built up in several small steps (increments).

The iterative & incremental approach is a risk mitigation technique to prevent the project from going off track via a short feedback loop. Projects have momentum that is a collection of the decisions and work accomplished. The willingness of people to admit to poor decisions or poor work, to discard that work especially after significant investment, and start over is rare. The larger the body of work to be discarded, the harder it is to restart. Small steps with verification that the work is correct and consistent with the project goals create a situation where potentially only a small investment is discarded. Small discards are easier to accept and can often be viewed as design explorations. The SAMEM applies the iterative & incremental approach at all levels across the complete project lifetime.

Modeling or graphical representations of the majority of project artifacts is mitigation against creating the incorrect solution and building the solution incorrectly. It is well known that visual expression can communicate information more accurately in many situations versus text. The well-known idiom of "a picture is worth a thousand words," which has been supported by cognitive psychology research, is the basis for the modeling emphasis. The mitigation effects against creating the incorrect solution, at least from the perspective of the stakeholders, are through the improved and more effective communication provided by models or graphical representations. One contribution for graphical representation or modeling is the idea of a Solution Overview Drawing (SOD) that provides a high-level single-page summary by integrating multiple views of requirements or design specifications.

To ensure that the mitigation techniques of iterative & incremental project management and modeling are effective, additional components are needed. While the SAMEM is applicable to the entire project process, there is a focus on the initial phases of the project. The motivation for focusing on the initial project phases are the well-documented costs of error fixing as the project progresses. Another contribution of this research is adding a mechanism to guide the iteration & increment work from the initial expression of the solution opportunities or goals to the final implementation. The guidance mechanism traverses the abstraction hierarchy from the initial vague ideas through the final implementation. The traversal is a collection of decisions, such as choosing one solution goal over another, choosing one requirement over another, choosing one solution architecture design over another, and choosing one implementation technology

over another. The decisions are represented as project artifacts and many can be well represented as models or graphically. The guidance mechanism for the SAMEM is the Open Distributed Processing – Reference Model ISO standard 10746 (RM-ODP).

An additional contribution that helps to ensure that the mitigation of iterative & incremental project management and modeling are effective is a set of engineering principles that provide quality guidance of the work. This component of the SAMEM is a set of principles that can be used throughout the project to examine the decisions and artifacts for correctness and quality. The principles are also useful in stimulating innovation by shedding light on assumptions and questioning habits. The principles are designed with the intention of domain independence, but focused on software centric solutions, which lead to their label as Software Engineering First Principles (SEFP).

The substantiation of the SAMEM as an effective methodology employs several means. Empirical evidence gathered through surveys from project members that used an early version of the SAMEM on multiple projects over several years is the main support. Cognitive effectiveness measures are evaluated against the graphical communication artifacts of the SAMEM. An evaluation of the SAMEM with the recent and evolving standard Software Engineering Method and Theory (SEMAT) provides a comparison of the strengths and differences. As the SAMEM is intended to be a practical and pragmatic software solution methodology, a discussion of the business benefits achieved in the projects is presented.

Acknowledgments

Firstly, I would like to thank my adviser, Prof. Dr. rer. nat. Bernhard Rumpe, for his help and guidance. He helped focus the research on the methodology. Most important was his generous offer to become my adviser after the passing of my first Ph.D. adviser, Prof. Robert France.

I would like to thank my first adviser, the late Prof. Robert France for his encouragement to pursue a Ph.D. Prof. France made me realize that I had much to offer from my industry experiences and the innovations in the practical software engineering I had developed.

Prof. Michel Chaudron provided very helpful feedback and improvements on the case study survey questions and format. I learned a great deal through his tutorial on Empirical Research in Model-based Software Engineering at the Models 2016 conference in Saint Malo, France.

I have had many colleagues and experiences over the 35 years of industry work (Hewlett-Packard and Integware) that have contributed in various ways to the views embodied in the SAMEM. One of the earliest influences was the introduction to Abstract Data Types just after a failed product. The product failure started me asking questions about avoiding such situations and the waste of effort. Abstract Data Types, an early version of Objects, offered a possible approach to improved product design. The work with Ralph Maderholtz in developing an internal training course for our fellow Hewlett-Packard R&D engineers in Structured Analysis and Structured Design helped to put me on the path of continuous software engineering improvement. The practice of analyzing each project at the end in order to make improvements and sustain good practices was a major influence on the continuous software engineering process improvement I use.

The cooperation on establishing standards with many people at the Object Management Group (OMG) in the late 1990s improved my ability at object-oriented thinking. Some of the more significant influencers on my object-oriented understanding were Jishnu Mukerji, Larry Johnson, and Andrew Watson. I started using the UML in 1996 in of my industry projects for design expressions. My personal use of the UML evolved into the education of my fellow workers, which further evolved into the realization that the UML can be used for more than code design, such as requirements specification. There are many dozens of colleagues that helped refine my understanding of modeling via the project interactions we had.

I would like to thank my colleagues at Integware who were the first users of the early version of the SAMEM, the Paper Prototype. Specifically, they helped refine the graphical presentations for the non-computer oriented customers by questioning the clarity of some of the UML diagram notations, contributing to a consistent style of presentation, and providing feedback on transitioning the expression of the requirements to inputs needed for design, testing, and documentation. The people that contributed significantly were Rob Ulrich, the project manager, Chris Ridout, a developer, Mike Hake, a developer, and Susan Tibbetts, a technical writer.

Most importantly, I want to thank my wife, Margrit Heuber-Matheson, for her support throughout the process. She spent many hours helping by proofreading multiple drafts. As a non-computer person, she offered many wording improvement suggestions.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 2
- 1.2 General Solution Direction . . . 3
- 1.3 Scope and Targets of the Thesis . . . 5
- 1.4 Organization of the Thesis . . . 5
- 1.5 Conventions Used . . . 7

2 Requirements Methodology State of the Art and Practice **9**
- 2.1 Survey of Existing Requirements Methodologies and Processes . . . 10
  - 2.1.1 Rational Unified Process (RUP) Methodology . . . 10
  - 2.1.2 Volere Requirements Process . . . 16
  - 2.1.3 Software & Systems Requirements Engineering in Practice Book . 19
  - 2.1.4 Requirements in Engineering Projects . . . 22
  - 2.1.5 System Modeling Language . . . 23
  - 2.1.6 V-Model . . . 25
  - 2.1.7 Software Engineering Method and Theory . . . 26
  - 2.1.8 Software Process Engineering Metamodel . . . 27
- 2.2 Survey of Existing Agile Methodologies and Processes . . . 28
  - 2.2.1 Agile Modeling . . . 28
  - 2.2.2 eXtreme Programming (XP) . . . 30
  - 2.2.3 SCRUM . . . 31
- 2.3 Limitations of Text-based Requirement Formats . . . 32
  - 2.3.1 A Diagram is (Sometimes) Worth 10,000 Words . . . 32
  - 2.3.2 Cognitive Effectiveness of Visual Notations . . . 33
  - 2.3.3 User Requirements Notation . . . 34
  - 2.3.4 Proposals for Requirements Models . . . 34
- 2.4 Summary of State of the Practice . . . 35
  - 2.4.1 Summary of Methodology Goals . . . 36

3 Definition of the Software Agile Modeling and Engineering Methodology **45**
- 3.1 Purpose of the SAMEM . . . 46
  - 3.1.1 The Secondary SAMEM Purpose is to Enable Innovation . . . 47
  - 3.1.2 Supporting the Engineering Due Diligence Purpose . . . 49
- 3.2 Software Engineering First Principles . . . 50
  - 3.2.1 Solution Conceptual Integrity (CI) SEFP . . . 51
  - 3.2.2 Essential and Accidental Complexity (Essential) SEFP . . . 53
  - 3.2.3 Stability to Variability (S2V) SEFP . . . 55

3.2.4 Symmetry of Action (SoA) SEFP . . . 56
3.2.5 Modules (Modules) SEFP . . . 56
3.2.6 Coupling and Cohesion (C&C) SEFP . . . 57
3.2.7 Patterns (Patterns) SEFP . . . 58
3.2.8 Optimal Performance (OP) SEFP . . . 59
3.2.9 Change Language (CL) SEFP . . . 60
3.2.10 Ready-to-Hand (R2H) SEFP . . . 60
3.2.11 Form Follows Function (FFF) SEFP . . . 61
3.2.12 Summary of Software Engineering First Principles . . . 61
3.3 The SAMEM High-level Structure . . . 62
3.4 The SAMEM Process . . . 65
3.5 The SAMEM Methodology Framework . . . 66
3.5.1 Objective of the Enterprise Viewpoint . . . 69
3.5.2 Objective of the Information Viewpoint . . . 78
3.5.3 Objectives of the Computational (Behavior) Viewpoint . . . 88
3.5.4 Objectives of the Engineering Viewpoint . . . 98
3.5.5 Objective of the Technology Viewpoint . . . 101
3.6 The SAMEM Model Artifacts Component . . . 105
3.7 The SAMEM Tool Component . . . 105
3.8 Non-functional Requirements Handling . . . 105
3.8.1 Look & Feel . . . 106
3.8.2 Usability . . . 106
3.8.3 Maintainability & Portability . . . 107
3.8.4 Performance . . . 107
3.8.5 Reliability . . . 109
3.8.6 Operational . . . 109
3.8.7 Security . . . 110
3.8.8 Legal . . . 113

4 Applying the SAMEM 115
4.1 Case Study Company Descriptions . . . 116
4.1.1 CS-1 Company and Project Descriptions . . . 117
4.1.2 CS-2 Company and Project Descriptions . . . 121
4.2 The SAMEM Introduction in the Consulting Company . . . 124
4.2.1 Medical Device Domain Education . . . 125
4.2.2 UML and Modeling Education . . . 126
4.3 The Experiences in Applying the SAMEM . . . 133
4.3.1 The Controlling Project-level Process . . . 134
4.3.2 Phase 1: Paper Prototype Phase . . . 135
4.3.3 Phase 2: Demo Prototype Phase . . . 136
4.3.4 Phase 3: Conference Room Pilot (CRP) Phase . . . 136
4.3.5 Using RM-ODP . . . 137

4.4 Lessons Learned . . . 147
4.4.1 Case Study Lessons Learned . . . 147
4.4.2 Comparison to Other Lessons Learned Research . . . 157
4.5 Applying the SAMEM Summary . . . 161

5 Methodology and Process 165
5.1 The SAMEM Process Model (SAMEM-PM) Concepts . . . 165
5.1.1 The Core SAMEM Process Model Concepts . . . 165
5.1.2 The SAMEM Process Model Concepts Mapped to Abstract Objects 168
5.1.3 Instance State Guidance . . . 168
5.1.4 Enabling Project Management . . . 170
5.2 Applying the SAMEM-PM: Instance Examples . . . 171
5.2.1 Methodology – Phase – Viewpoint Concept Examples . . . 172
5.2.2 Viewpoint – Iteration Concept Examples . . . 173
5.2.3 Iteration – Artifact Concept Examples . . . 174
5.3 An Activity Model Example for the SAMEM-PM Flow . . . 175
5.4 The SAMEM-PM Concepts Summary . . . 177

6 The Language Used for Information Models in the Projects 179
6.1 The SAMEM-IM Concept Model . . . 179
6.1.1 The SAMEM-IM Artifact Concept Definition . . . 180
6.1.2 The SAMEM-IM Communication Format Concept Definition . . . 181
6.1.3 The SAMEM-IM Properties Concept Definition . . . 183
6.1.4 The SAMEM-IM External Standard Metamodel Concept Definition 183
6.1.5 The SAMEM-IM Status State Concept Definition . . . 183
6.2 Applying the SAMEM Metamodel Composition Concept to UML . . . 184
6.2.1 Composition Extensions for the UML Diagram Definition Specification . . . 185
6.2.2 Composition Extensions for the UML Specification . . . 188
6.3 Information Modeling Summary . . . 194

7 The SAMEM Evaluation 197
7.1 Empirical Evaluation of Case Study Companies . . . 198
7.1.1 Survey Definition Details . . . 199
7.1.2 Answers to the Common Questions Merged . . . 204
7.1.3 Unique Survey Questions Response Values . . . 209
7.1.4 Survey Results and SAMEM Goals Evaluation . . . 211
7.1.5 General Opinions from the Case Study Surveys . . . 237
7.2 Other Applications of the SAMEM . . . 240
7.2.1 Start-up Company for Mobile Application Development . . . 240
7.2.2 Company Specializing in Employee Background Checks . . . 240
7.2.3 Company Creating Safety and Security Solutions for the Oil and Gas Industry . . . 242

7.3 Communication Cognitive Effectiveness Measures of the SAMEM Artifacts 242
7.3.1 Cognitive Effectiveness Principles and Metrics . . . 243
7.3.2 Cognitive Effectiveness Evaluation of the SOD . . . 244
7.4 The SAMEM Mapping to SEMAT and Essence . . . 245
7.4.1 The Essence Kernel Definition . . . 245
7.4.2 The SAMEM and SEMAT Comparisons . . . 255
7.5 Consulting Business Success Measures . . . 260
7.5.1 Team Size . . . 260
7.5.2 Speed to Requirements . . . 261
7.5.3 No Requirements Prioritization . . . 261
7.5.4 Quality of Requirements . . . 262
7.5.5 Customer Satisfaction . . . 262
7.5.6 Consulting Company Business Benefits . . . 263
7.6 Threats to Validity . . . 263
7.6.1 Construct Validity . . . 263
7.6.2 Internal Validity . . . 264
7.6.3 External Validity . . . 264
7.6.4 Reliability . . . 264
7.7 The SAMEM Evaluation Summary . . . 265

8 Chapter 8 Conclusions and Future Work 269
8.1 Contributions . . . 269
8.1.1 Research Questions Addressed . . . 269
8.1.2 Other Contributions . . . 272
8.2 Future Work . . . 273
8.2.1 The SAMEM Improvements . . . 273
8.2.2 Expand the SAMEM into Other Domains . . . 273
8.2.3 UML Extensions Based on SOD Idea . . . 274
8.2.4 The SAMEM and the SEMAT Integration . . . 274
8.2.5 Tool Possibilities . . . 274

Bibliography 277

Glossary 285

A Customer Questionnaire 289

B Developer Questionnaire 305

C Customer Response 1 321

D Customer Response 2 327

E Customer Response 3 333

F Customer Response 4 **337**

G Developer Response 1 **343**

H Developer Response 2 **349**

I Developer Response 3 **355**

J Developer Response 4 **361**

K Curriculum Vitae **367**

List of Figures **371**

List of Tables **375**

Chapter 1

Introduction

The objective of this thesis is the development of a software engineering project methodology based on modeling of requirements and an iterative & incremental process that is an improvement to the more traditional text-based requirements specification using a sequential, phase-based process. This thesis extends Model-driven Engineering (MDE) work, the majority of which has been focused on code generation or Domain Specific Languages (DSL), to a broader approach and applying MDE from the earliest moments in the solution development process. The ideas of agile code development are extended to cover the development of all project artifacts during the complete project lifecycle.

In this thesis *methodology* is defined as a collection of artifacts, tools, and processes related in a structured way with a general, but adaptable choreography of execution which is used to create a product or solution. The name for the new methodology is the *Software Agile Modeling and Engineering Methodology* (SAMEM).

While the SAMEM covers the entire project process from first contact with the stakeholders concerned with the problem to be solved through the deployment of a solution, the focus is on the front-end of the process. The front-end roughly covers establishing the goals of the solution, the business limits, the organizational boundaries, the elicitation of the requirements, and the solution architecture design. As is described in [Brooks10], the expression of the goals, limits, boundaries, and requirements are often revised, as this work entails discovery of information, testing of assumptions, and sometimes the invention of novel engineering approaches. Therefore, the boundary between the front-end and the back-end, which covers design and implementation work, is not hard and precise. The front-end work can be summarized as the specification of *What*, *Who*, *Why*, and *What Not*, while the back-end is summarized by the specification of *How*. The SAMEM is focused primarily on the front-end of the project process, but can be integrated with more prescriptive approaches for the generation of implementations, e.g. code generation tools.

The primary motivation for focusing the SAMEM on the project front-end comes from the costs of fixing errors in the solution. The median cost of fixing an error that was generated in the requirements work and discovered in an in-production solution is 50 times the cost of fixing at the requirement stage [NAS03]. While this thesis does not claim to eliminate errors, the SAMEM does attempt to minimize the probability of an error in the initial project work and enable an improved chance of early detection.

As used in this work, the definition of *model* is a human construct representing some aspects of reality that enables better understanding and communication of a particular

perspective of a problem and/or solution [FM15]. A model allows one to reason about complex situations and potential solutions. The usefulness of a model is in its abilities to describe, suggest, explain, predict, and simulate [LS87], [FM15]. A *visual model* is one consisting of primarily graphical elements annotated with text as opposed to a pure text description.

The first thesis hypothesis is that visual modeling of most of the requirements is possible and superior to a collection of simple text statements. The models are graphical images in form and images communicate many ideas better than text [LS87], [Mat11]. It is recognized that some requirements, such as simple facts, are expressed better as textual statements. For the best possible requirements specification a combination of both is needed. This thesis focuses on understanding the strengths and limits of modeling requirements within the context of a large project. A large project is defined as at least 10 engineering-years of effort and this is not a precise measure as projects vary in complexity. The second thesis hypothesis is that the development of the requirements specification is best performed in an iterative & incremental process. The value of this process approach has been demonstrated often by the SCRUM community [Coh10]. The main process idea is that a small step in developing a project artifact (increment) is taken and immediately followed by verification that the step is correct in direction and content. Then the next small step is taken (iteration) and so on, constantly building up the requirements specification.

Brooks [Bro95] formulates the law of the *Mythical Man-Month*, which states that adding more people to a project only slows it down, The inter-team communication grows as *n(n-1)/2*. Minimizing the team size reduces the effort spent on the team coordination communication. A third thesis hypothesis is that the accidental complexity [Bro95] in the nature of the communication can be reduced through careful structuring of the project process and the process artifacts. If the accidental complexity of the communication can be reduced, then the potential for needing fewer project developers arises as each developer is more effective, thus also minimizing Mythical Man-Month impacts.

The *methodology* described in this thesis is a pragmatic approach to realizing a successful solution to the three hypotheses. The SAMEM was developed in a consulting industry setting and successfully used since 2008 in two major projects with two different customers by two separate development teams and produced a total of 18 separate solutions, which are still in use. For more details on the case study companies and the projects see Sub-chapter 4.1. In addition, ideas from the SAMEM approach have been applied in three other companies, which are described in Sub-chapters 7.2.1, 7.2.2, and 7.2.3.

1.1 Problem Statement

This work is targeted at medium to larger projects, with the lower borderline for medium at about 10 engineering-years of effort. Projects of this size suffer more from Mythical Man-Month effects and the accidental complexity than accompanies smaller projects [Bro95]. This thesis attempts to mitigate the accidental complexity of additional com-

munication arising from the Mythical Man-Month and the greater volume of information to be shared in the team. There are a number of problems that remain poorly addressed.

One of the problems is poor requirements specification, which has two major aspects. One, the requirements are usually expressed in a text format with a one-dimensional organization [LS87]. This results in the choice of which related requirements are listed together. The single dimensional organization of text requirements introduces accidental complexity through the need to jump through the specification in a non-linear manner to gather all the aspects of a requirements concept. Second, the requirements are often too granular because of the expressive limitations of text, which results in a long specification. There is a large semantic and syntactic gap from requirements text to design models.

Another major problem is the transparency of the requirements elicitation process. Traditionally the process has been a sequential, phase-based process, often because it was managed in the Microsoft Project™ tool. There are natural iterations in the requirements elicitation process that are not easily represented in the Microsoft Project™ tool [Bro10], [RR99]. There are often too many review cycles because of poorly understood requirements or inability to remember the whole specification. This results in artificial convergence by management declaring requirements to be complete, while the reality is that there exist gaps, contradictions, and errors in the specification. The flow from requirements elicitation into design is hindered by the regrouping and translation from text statements to design models.

The problems produce a corresponding set of high-level goals for the thesis:

- Better stakeholder communication with respect to the requirements specification and project progress.
- Better development team communication.
- Improved overall requirements quality in a more compact format.
- Faster project process.
- Smaller project team size, especially for the development team.
- Compatibility with agile methods for solution (code) development.

1.2 General Solution Direction

The three thesis hypotheses are general in nature and many solutions can appear. One of the sub-problems is how to proceed through the development project] with effectiveness [FM15], [Bro10]. Effectiveness means that progress is made at a reasonable pace without too many false steps and without getting stuck. The evaluation of effectiveness is a subjective judgment that the customer, the organization paying, and development team managers make. However, a framework for coordinating the artifacts, tools, and process can assist in achieving effectiveness. The tools, software engineering principles,

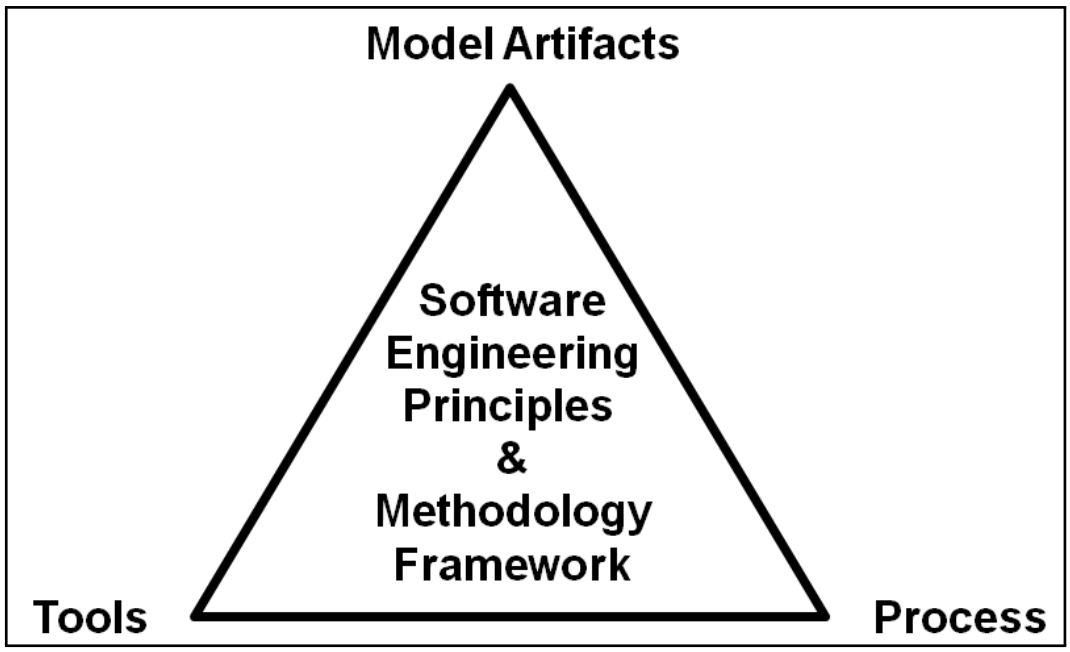


Figure 1.1: SAMEM Components.

framework, and process when combined with artifacts that effectively communicate are the core of the SAMEM. This is illustrated in Figure 1.1.

There are five components in Figure 1.1 that define the SAMEM: 1) Model Artifacts, 2) Process, 3) Tools, 4) Software Engineering Principles, and 5) Methodology Framework.

- **Model Artifacts** – the primarily visual or graphical models for the requirements sometimes with text for clarification, but can also include text artifacts when text is a better medium for expression. The characteristics of communication effectiveness and compactness of form are key evaluation criteria. Many of the model artifacts are based on the Unified Modeling Language (UML) [OMG15b], [BRJ99], [BR05], [RJB99].

- **Process** – the ordered work tasks needed to develop the solution. Processes exist on multiple levels and are nested from high-level guidance to lower-level details.

- **Tools** – refers to any tool used in the development process by any team member to create, update, communicate, and manage the model artifacts or the process itself. The range of tools includes mental tools used by the people, physical tools, and computer-based tools.

- **Software Engineering Principles** – are a type of mental tool as they are an embodiment of best practices that can be applied in many circumstances during the solution development process. They help assure high quality.

- **Methodology Framework** – is a project organization template (framework) that structures and coordinates the work, the use of tools in a logical way, and assures effective progress towards the solution completion.

The body of this thesis explains these components in greater detail along with the rationales for the decisions. The rationales come in part from other research and from lessons learned in multiple case studies. In addition to the definitions, the SAMEM shows how the components interact and work together to achieve the stated goals.

1.3 Scope and Targets of the Thesis

- *Research Question 1 (RQ1)*: Is the visual modeling of the majority of requirements possible?
- *Research Question 2 (RQ2)*: Is an iterative & incremental (agile) process approach effective in the elicitation of requirements?
- *Research Question 3 (RQ3)*: Does there appear to be a reduction in the accidental complexity of team communication with visual models for the requirements and design?
- *Research Question 4 (RQ4)*: Does visual modeling have a positive effect on the size (smaller) and clarity (easier to comprehend) of the requirements specification?
- *Research Question 5 (RQ5)*: Does an iterative & incremental full project process contribute in a positive manner to a faster project process with fewer people?
- *Research Question 6 (RQ6)*: Do iterative & incremental (agile) approaches in the front-end of a project process more effectively feed into agile development processes?

The case studies used throughout this thesis come from actual business solutions in production use in the medical device industry. Details on the case study companies are in Sub-chapter 4.1. The case studies will provide data to be analyzed and provide a base of real data for the development of the model and meta-model definitions. Certain proprietary information under the scope of Non-Disclosure Agreements (NDA) has been removed or modified, but the examples remain true to the reality of the solution.

1.4 Organization of the Thesis

The thesis is organized as follows:

- Chapter 2: Requirements Methodology State of the Art and Practice

  The content of this chapter reviews several other requirements and software engineering methodologies or processes and compares them to the goals for this thesis.

The strengths and limitations of each methodology or process are evaluated. From the evaluations, a set of Goals for the SAMEM is developed which reflect the strengths and address the limitations.

- Chapter 3: Definition of the Software Agile Modeling and Engineering Methodology

  The details of the SAMEM components shown in Figure 1.1 are defined with rationale for the choices and examples. The purposes of the SAMEM, the Software Engineering First Principles, and non-functional requirements handling are covered. The design of the SAMEM is related to the Goals defined in Chapter 2.

- Chapter 4: Applying the SAMEM

  This chapter contains examples of introducing the SAMEM into a company, applying the methodology in various commercial situations, process modeling techniques, and additional lessons learned in the real-world case studies. The organization issues encountered with the introduction and education of the SAMEM process are typical for any new methodology change.

- Chapter 5: Methodology and Process

  A description at a meta-level of a model for the process components of the SAMEM process flows is contained in this chapter. The purpose of the meta-level is to have a foundation for adaptation to other domains and on which to build tools. An example of a project controlling process from the case studies is presented.

- Chapter 6: The Language Used for Information Models in the Projects

  This chapter defines a metamodel for information components in the SAMEM. As with Chapter 5, the metamodel provides a foundation for adaptation and tool development. A set of extensions to the UML is described based on the SAMEM innovative information modeling ideas.

- Chapter 7: The SAMEM Evaluation

  Evaluation evidence for the SAMEM is provided in this chapter. There are five dimensions used to evaluate the SAMEM and the threats to the validity of the work are considered.

- Chapter 8: Conclusions and Future Work

  Summarizes the contributions of this thesis and lists additional research to expand this work to other domains.

- Bibliography

- Glossary

  A glossary of terms and acronyms used in the thesis, although terms and acronyms are also explained on first use within the body of the thesis.

- Appendix A – B

  The questions of the Customer Survey and Developer Survey as screen captures from SurveyMonkey.

- Appendix D – J

  The responses to the Customer Survey and Developer Survey as screen captures from SurveyMonkey.

- Appendix K

  Curriculum Vitae

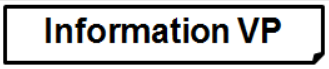
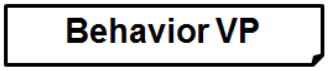
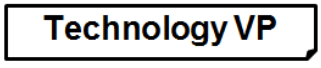
- List of Figures

- List of Tables

1.5 Conventions Used

Within the thesis several classification marks are used to help identify and correlate the information. These can appear in the text or in figures and multiple classification marks can appear together.

Table 1.1: Classification Marks.

Classification Mark	Meaning
	indicates the information or data is from a case study
	is used to indicate a lesson learned from the case studies
	RM-ODP Enterprise Viewpoint component
	RM-ODP Information Viewpoint component
	RM-ODP Computational (Behavior) Viewpoint component
	RM-ODP Engineering Viewpoint component
	RM-ODP Technology Viewpoint component

Chapter 2

Requirements Methodology State of the Art and Practice

There are several well-known and commonly used approaches and methodologies for requirements elicitation and management. Some of the methodologies, such as the Rational Unified Process (RUP) [PK00] and the V-model [V-M16a], [V-M16c], [V-M16b] also continue on through the whole development process.

The agile processes *SCRUM* and *eXtreme Programming* also have requirements elicitation approaches, however, the requirements' needs grew from the coding process. They, like RUP, also have tasks and work that includes development activities. The SAMEM proposed covers the complete development process but with emphasis on the initial project phases.

There are also approaches coming from other research and ideas for improving software and systems engineering. The Object Management Group (OMG) has the System Modeling Language (SysML) [OMG15a], which contains modeling components for requirements. Also within the OMG is the Software Process Engineering Metamodel (SPEM) [OMG08] standard. Growing from an effort to define a software engineering theory, there are two sources, Essence – Kernel and Language for Software Engineering Methods [OMG12b] and Software Engineering Method and Theory (SEMAT) [SEM98].

The last area of existing work evaluated is the form of the requirements artifacts. In this area the value of text-based versus graphical-based artifacts is compared.

This chapter contains an evaluation of these software engineering and requirements methodologies and compares them with the objectives of this thesis.

A *project* is defined as a defined set of work with a start point and end point, which is limited in time, people, money, and other resources, that produces a product or solution. The project definition is similar to that defined in [FM15]. The term project at times might be preceded by a qualifying word producing a term such as development project. A *development project* involves development work of some kind to produce a product or solution.

Product and *solution* are synonyms which indicate the result of a project. Product is used when the discussion focuses on the creation of a result that is intended to be sold multiple times commercially, while solution is used when a project result is built for a single customer. They will be used somewhat interchangeable throughout this work.

A *process* is a set of tasks with a defined choreography of execution that produces a specific result. A development process is therefore the set of work tasks that produces

the solution. A work task in a process could be defined as a process or a sub-process, thus creating a hierarchy of processes. For example, a project process could be partitioned into requirements gathering process, a solution development process, and a solution validation process.

Customer is the generic term for the people or organization paying for the product or solution. This can include the *users* who actually use and interact with the solution. Also included within customer could be the stakeholders. A *stakeholder* is someone who in some way is impacted by or concerned with the solution. A regulatory agency that monitors the business the solution is used in is a type of stakeholder often overlooked. The customer and the users are the most important stakeholders. In general, stakeholder will be used throughout this work unless a more specific term is relevant.

Each of the evaluated methodologies and techniques has strengths and limitations. To ensure that the new methodology is an improvement, the SAMEM goals will be derived from both the strong points and the limitations. Each methodology goal will have a unique identifier. Below is an example of the format for the SAMEM goal:

GOAL-0: Example format for a SAMEM goal.

2.1 Survey of Existing Requirements Methodologies and Processes

Some of the requirements processes have names, such as RUP [PK00] and Volere [RR99]. In other cases a book defines the process, but does not give it a name [FM15], [BPKR09]. There are also open source and standards-based requirements and development methodologies efforts typified by Essence [KM] and Software Process Engineering Metamodel (SPEM) [OMG08]. Many of the existing methodologies were developed to address project problems, both symptoms and root causes. A *symptom* is an observed indicator of a problem, such as excessive rework. A *root cause* is the core deficiency that causes the problem and might be observed through one or more symptoms. For example, excessive rework might have the root cause of poorly formulated requirements which cause misunderstandings.

2.1.1 Rational Unified Process (RUP) Methodology

The Rational Unified Process (RUP) methodology [PK00] is a software engineering process, a process product, and a process framework. The RUP methodology was developed to address the following software development project symptoms and root causes (from [PK00]):

1. Symptom – Inaccurate understanding of end-user needs.
2. Symptom – Inability to deal with changing requirements.
3. Symptom – Late discovery of serious project flaws.

4. Symptom – Team members in each other's way, making it impossible to reconstruct who changed what, when, where, and why.

5. Root cause – Ad-hoc requirements management.

6. Root cause – Ambiguous and imprecise communication.

7. Root cause – Overwhelming complexity.

8. Root cause – Undetected inconsistencies in requirements, design, and implementations.

9. Root cause – Failure to attack risk.

The symptoms and root causes that drove the RUP development are in alignment with the problems cited in Sub-chapter 1.1, although listed at a finer level of detail for RUP. The above list of symptoms and root causes can be accepted for this work. How has the RUP methodology addressed these symptoms and root causes and how well has it addressed them? The symptoms and root causes listed above lead to several SAMEM goals. Symptom 1, root cause 6, and root cause 8 lead to ***GOAL-1***. ***GOAL-2*** is the consequence of addressing symptom 3, root cause 8, and root cause 9. Symptom 2, symptom 4, root cause 5, root cause 7, and root cause 8 drive the need for ***GOAL-3***.

GOAL-1: The methodology should provide accurate communication mechanisms.
GOAL-2: The methodology should provide project process risk mitigation mechanisms.
GOAL-3: The methodology should provide traceable artifact evolution.

2.1.1.1 RUP Definition

The organization of the RUP is two-dimensional, see Figure 2.1. The horizontal axis is time and is divided into four major phases:

1. **Inception** – the start of the project is where the setting of goals and objectives including the initial requirements are collected.

2. **Elaboration** – the gathering, refinement, and verification of the requirements. The design work is started in this phase and much progress is accomplished.

3. **Construction** – the creation of the solution, i.e. the coding. The solution creation can uncover problems that necessitate modifications to the design or corrections to the requirements. During the construction testing at various levels of abstraction occurs.

4. **Transition** – placing the completed solution into production use.

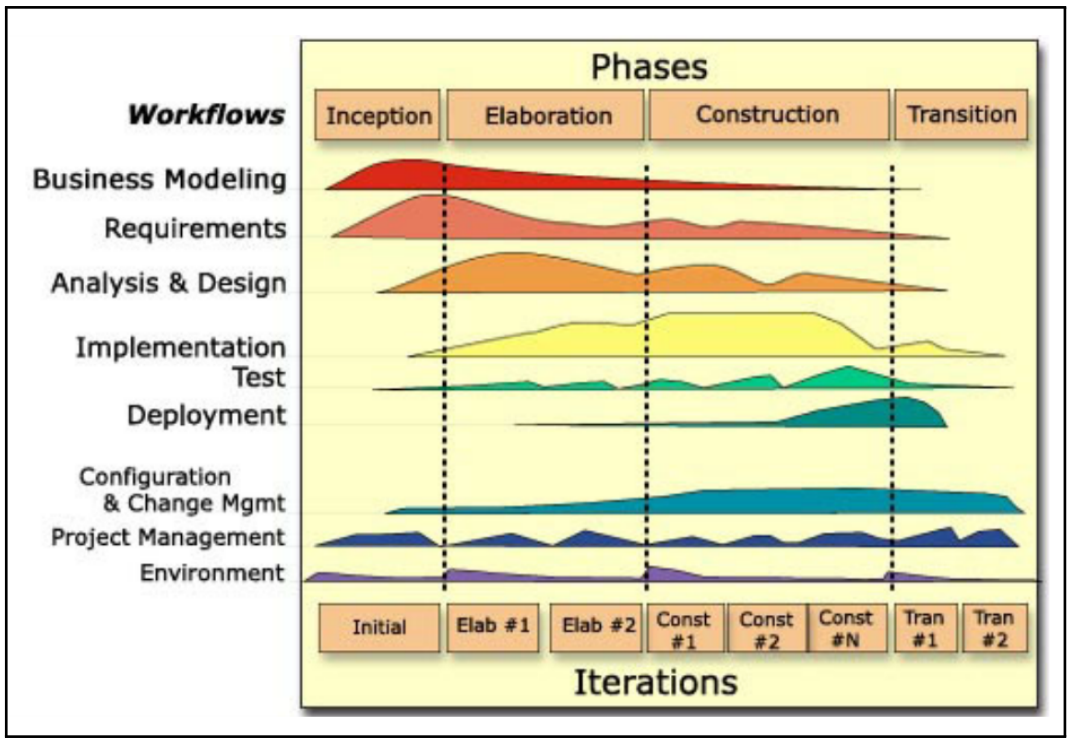


Figure 2.1: RUP Phases and Workflows Organization.

The end of each phase is signified by a milestone. At the milestone evaluations are performed to determine whether it is appropriate to proceed to the next phase. The main evaluation is related to the project risk of proceeding with either insufficient or incorrect information. Within the RUP phase framework, the work can be broken into pieces and carried through the phases. These pieces become iterations that are used to build the full solution in steps. In Figure 2.1, the idea of iterations are shown as blocks with labels such as “Elab #1” or ”Const #2.”

The vertical axis denotes content and is divided into nine workflows for accomplishing work: Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration & Change Management, Project Management, and Environment. Following each workflow is a colored area that spans the RUP phases. The colored area represents relative amount of effort that is expended on that workflow within a particular phase. For example, the effort expended for the Business Modeling workflow is at maximum in the Inception phase, slowly declines during the Elaboration phase, trickles off to near zero in the Construction phase, and disappears entirely in the Transition phase. The other workflows have their corresponding higher and lower levels of effort across the RUP phases. The effort curves in Figure 2.1 reflect general averages and will vary somewhat from project to project.

The purposes of each of the workflows are:

1. **Business Modeling** – this workflow collects the business goals for the solution under consideration.
2. **Requirements** – the solution requirements are gathered and evaluated in this workflow.
3. **Analysis & Design** – the design of the solution and analysis activities determining correctness are performed in this workflow
4. **Implementation** – the solution is implemented in the activities associated here.
5. **Test** – testing of the solution implementation takes place on various levels.
6. **Deployment** – the completed solution is installed into its execution environment and made available to the users.
7. **Configuration & Change Management** – controls the evolution of the artifacts created in the other workflows and provides traceability.
8. **Project Management** – contains the tasks for managing the project work, resources, and ensuring the solution is delivered.
9. **Environment** – in this workflow the creation and maintenance of the solution development environment takes place.

The RUP is based on an architecture-centric process [PK00] that is driven by modeling. The architecture view is the 4+1 view, which consists of the following five views: Use-Case View (Scenarios), Logical View, Implementation View, Process View, and Deployment View. A Use-Case-Driven process is at the core of RUP, Figure 2.2. Use-Cases are the drivers through the workflows and model artifacts.

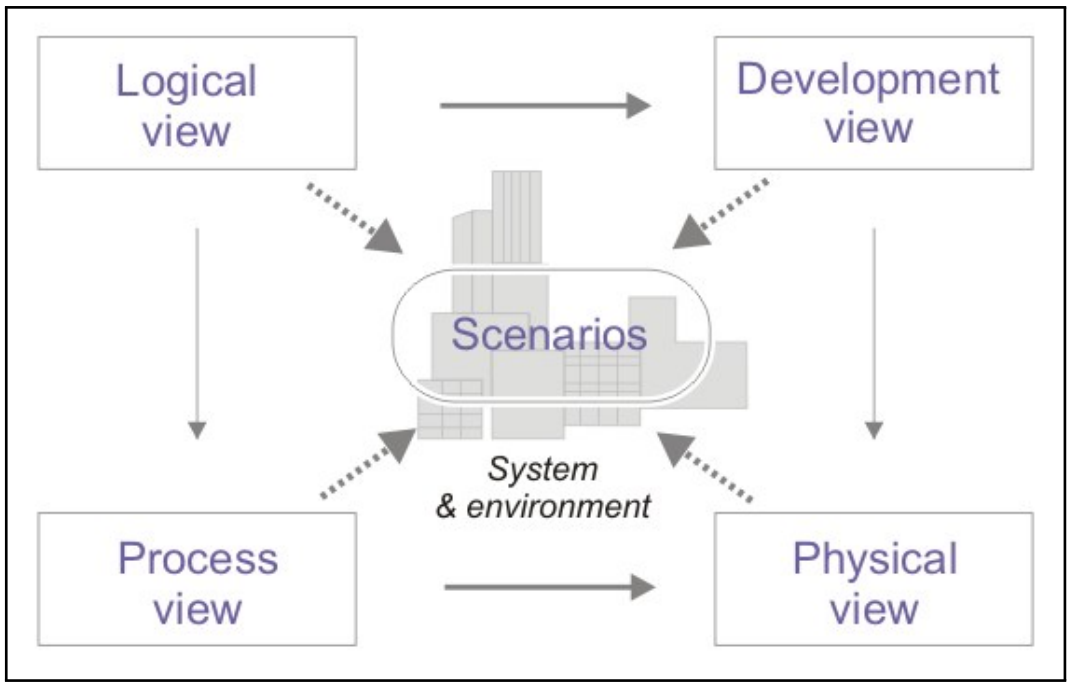


Figure 2.2: RUP 4+1 Architecture Model.

2.1.1.2 Strengths of RUP

The similarities in the problems seen in software engineering projects are listed in Subchapter 1.1. Those problems drive solution choices for the SAMEM approach in terms of artifacts, tools, and processes. Some of those choices are based on current best state-of-the-practice knowledge.

The first strength is an iterative development approach. A single iteration has pieces in each of the elaboration, construction, and transition phases producing an executable release. A collection of iterations is called a cycle. Iterations or cycles are chained together for the evolutionary development of the solution.

The second strength is the use of modeling. The full complement of UML models is used throughout the process. There is a strong emphasis on starting with Use-Case Models in the Business Modeling and refining or transitioning to the other models for the Analysis & Design phases.

A strong advantage is the idea of a flexible framework to guide the development process through the phases and across the workflows. The flexibility enters through the iteration size, the cycle size, and the ordering of modeling activities. These aspects are mostly part of the Project Management workflow.

The management of the requirements is very important. This includes knowing when a requirement was created, who wanted the requirement, understanding if the requirement evolved during the development process, and the ability to trace the requirement to the design and implementation.

The following goals for the SAMEM are generated in order to maintain the above listed strengths of the RUP.

GOAL-4: The methodology and artifacts should be compatible with an iterative & incremental project process.

GOAL-5: The artifacts should maximize the use of visual models for communication and compactness.

GOAL-6: The methodology should be adaptable to individual project process needs.

2.1.1.3 Improvements to RUP

There are several areas in RUP that can be improved. There have been several evaluations and applications of RUP that have uncovered limitations and areas for improvement [AGBA10], [GXX11], [Hir02], [KP02], [KGK07], [NK11], [OCH11], [Tan15], and [ZHG05].

In the SAMEM there are multiple small iterations within each project phase and a verification step is integral to each iteration. The iterations in RUP are very coarse, thereby reducing the verification feedback. This difference also translates to iteration size. With smaller iterations the corresponding increment for verification is smaller. The smaller increment means less to verify by the customer, which makes the verification activity easier to fit into their schedule and a smaller rework effort if the increment has problems.

A lesson learned from the case studies (Sub-chapter 4.4.1.1) showed that the optimal amount of material to present in a verification review was time scoped to about one hour of the stakeholder's time. As the stakeholder gained experience with the review tasks, the models, and process, the amount of information that could be effectively reviewed in an hour grew by as much as 100%. The content density had an influence on the verification review amount and the coherence of the information, but the rule-of-thumb developed was about five information units with associated attribute definitions and state machine or 15 tasks in a business flow. The SAMEM terms *Information Unit* and *Business Flow* are defined in detail in Sub-chapters 3.5.2 and 3.5.1.

In RUP the modeling stays close to the UML, even if it interferes with the customer communication. Alternative models for communication are not proposed. The RUP methodology primarily employs Use-Case Models for behavior at the individual action level, which makes understanding of the overall behavior difficult. In the SAMEM the modeling forms are adjusted to enhance the communication [Mat11]. When the UML did not have the appropriate modeling artifacts, new ones were developed. The modeling details will be described in Chapter 3. The primary goals of the new models are to effectively have a visual record and communication of the important ideas of the solution.

The RUP process assumes that the design will be "right on paper" before proceeding, which contradicts our case studies experience, the experiences described in Brooks [Bro10] and experiences described in other requirements engineering process books [RR99], [BPKR09], [FM15]. The SAMEM looks for stability not completeness in the requirement

artifacts before proceeding to more detailed phases of work such as prototyping.

The following goals for the SAMEM are intended to avoid limitations within the RUP.

GOAL-7: The iterations should be small for project management purposes.

GOAL-8: An iteration should have a verification step of some kind to assure that the work done is correct.

GOAL-9: Flexibility in the visual model artifacts is necessary for communication optimization.

2.1.1.4 RUP Methodology Summary

In summary, with respect to the goals of the SAMEM the RUP methodology has some shortcomings. While it uses iterations similar to agile methods, the iterations are large; therefore the verification feedback loops are longer. Modeling is used, but it is not adjusted to maximize communication either with the stakeholders or within the development team. Also the modeling is not used to the full extent it could be to create a more compact form for the requirements specification. It is unclear how the use of RUP can help to reduce the team size and the resulting Mythical Man-Month impacts.

2.1.2 Volere Requirements Process

The Volere Requirements Process [RR99] focuses on the elicitation and documentation of requirements. The most important contributions are a set of 27 requirements categories to ensure that all types of requirements are gathered, a model of the requirement type inter-relationships, and a documentation format for consistency of representation. The Volere methodology is limited to requirements.

The content of the requirements format is text-based which is placed in a specific and consistent format, the requirement card. A useful field on the requirement card is the *Fit Criteria*, which is a value that can be measured in the solution via testing to verify the requirement is met. While the form of the requirements documentation was not used in the case studies, as it is based on text on 5" x 8" cards, the categories are used. The categories provided the inspiration for the columns in the Action Transformation Matrix behavior pattern (see Sub-chapter 3.5.3).

The process part of Volere is centered on the trawling for requirements. Trawling is the term they use for the elicitation activities of interviewing users and stakeholders. As each requirement is sifted out from the interview process, it is recorded on a card with a unique identification. There is a seven-step process defined as a data flow diagram, which helps to check that all the requirements have been discovered. As part of the trawling process, there are suggested questions for each category to help elicit the requirements.

2.1.2.1 Strengths in the Volere Methodology

There are several important strengths in the Volere methodology. The main strength is the list of 27 major categories to ask about in order to elicit requirements. Within the major categories are sub-categories and sets of example questions to start the elicitation

work. Within the sub-categories are examples of the questioning results. The example questions and answers provide an excellent training approach for new software engineers. Some examples of the questions are shown in Table 2.1. The first column in Table 2.1 lists some of the 27 categories and the right column lists a few of the questions.

Table 2.1: Volere Example Questions.

Volere Requirement Category	**Example Questions**
Functional and Data	1. What data must be received by the product? 2. What data must be produced by the product? 3. What calculations must be made by the product? 4. What decision must be made by the product?
Look and Feel	1. Should the product conform to an existing standard such as Microsoft or Apple? 2. Are there existing paper representations that should be adhered to? 3. Which users should the product interface cater to? 4. What is the domain experience level of the users?
Operational	1. What are the physical conditions of the environment in which the solution will be used? 2. Will the users have limited interaction abilities because of protective gear?
Security	1. Which data is sensitive and needs controlled access? 2. Which function invocations should be controlled? 3. Which data have high integrity needs?

The second strength is a standard format to document not just the requirement, but important related information about the requirement. Some of the related information is the rationale for the requirement, the originator, the fit criterion, the type of the requirement (which of the 27 major categories), customer satisfaction, and customer dissatisfaction. The fit criterion is an objective and testable measure of how well the requirement is satisfied in the solution. The customer satisfaction and dissatisfaction rankings reflect how happy the customer will be if the requirement is in the solution and respectively not in the solution. The related information is helpful in requirement prioritization activities.

An important and useful concept in the Volere methodology is the *Quality Gateway*. The purpose of the quality gateway is to prevent unworthy requirements from becoming part of the specification. There are a series of tests on the requirement to ensure that the requirement is complete, accurate, and is implementable. Most of the tests are performed against the requirement as stated in the standard format.

To preserve the strengths from the Volere methodology, the following goals for the SAMEM are derived.

GOAL-10: The methodology should ensure completeness.

GOAL-11: The methodology should support the incorporation of state-of-the-art software engineering results.

GOAL-12: The visual modeling and textual artifacts need to have consistent presentation to optimize communication.

2.1.2.2 Limitations in the Volere Methodology

While the consistent format and the categories offer help in discovering and communicating requirements, there is the limitation of a short textual expression that constrains the usefulness. A software engineer converting the requirements into a design is still faced with the reading of many individual requirements and the accidental complexity of the mental integration into a full picture. Individual requirements are communicated well, but not the big picture.

The text and small physical format of the cards effectively prohibits any business process overview requirement from being documented. The Volere methodology does not have an effective way to connect the many individual requirements into a larger set, which would communicate a broader solution behavior. Because the cards are small and uniquely identified, they can be duplicated and placed into multiple groups for organization. The card grouping is for some requirement correctness and completeness evaluation activities an improvement over a single text document.

Because the Volere methodology is targeted only at requirements, it tends to support the sequential, phase-based process approach rather than an iterative & incremental project process. It is fairly obvious that the small requirements on the Volere cards could match the creation of SCRUM user stories to start iterations for design and implementation work. On the other hand, the Volere card expression could at times be too fine-grained for effective SCRUM user stories.

The following repeated SAMEM goal is intended to avoid some of the limitations in the Volere methodology.

GOAL-5: The artifacts should maximize the use of visual models for communication and compactness.

2.1.2.3 Volere Methodology Summary

In summary, the Volere methodology has most of the communication disadvantages of a text-based method. A redeeming factor is the additional information collected about the requirement. The additional information improves the quality of the requirement, but does not help produce a more compact form. While the individual cards can be arranged in multiple ways, the complexity of dealing with many physical objects does not scale in a positive manner. There is the possibility that the individual cards can help with the transition to SCRUM user stories, but there are possibilities for granularity mismatch. There are no indications that the Volere methodology helps to speed up a project or reduce team size.

2.1.3 Software & Systems Requirements Engineering in Practice Book

The book *Software & Systems Requirements Engineering in Practice* [BPKR09] (Berenbach Book) is written from experiences on actual industry projects within Siemens. While mostly focused on requirements elicitation and management techniques, there are short forays into the product design activities.

The processes described use modeling in conjunction with requirement expressions, but for analysis and as a transition between the text-based requirements and the design models. The process and the examples are centered on the requirements expressed as text, usually as a single simple sentence. Throughout the book there is a clear separation between functional requirements and nonfunctional requirements.

2.1.3.1 Strengths in the Berenbach Book

There is a match between the intended use of models in the SAMEM and the work in the Berenbach Book. That match can be summed up by a quote from the book:

"When using models as part of an engineering process, one of the objectives is to convey as much information as possible as succinctly as possible."

An interesting contribution is the Requirements Engineering Artifact Model (REAM). The REAM is a metamodel for structuring the requirement model artifacts created during the requirements engineering process. The REAM describes the types of artifacts and the possible relationships between them. The REAM is used during analysis of the requirements artifact model to insure correctness and to generate both the text requirements and some of the high-level design (mostly by heuristic methods).

In Figure 2.3, a REAM example from [BPKR09] is shown. The artifact types are concrete. The actual requirements artifact model can be verified against the REAM for completeness. A complete artifact model will have artifacts of every type defined in the REAM. Verification for consistency is possible by checking the relationships between the actual requirement artifacts. For each project a specific REAM can be constructed to match the domain artifacts. The metamodel for the REAM is employed in their process to verify completeness of requirements and to help communicate the incompleteness to the stakeholders

There are differences between the artifacts in a requirement engineering model and a UML Class Model. A UML Class Model has attributes and operations, while a requirement engineering model only has a name and description. A UML Class Model can have abstract classes, but a requirements artifact model shows real objects which are connected with a simple association.

Much of the modeling and the elicitation work are driven from Use-Cases. They do suggest creating scenario diagrams to chain use cases together for a higher-level process view of the business situation. The scenario diagrams are similar to the UML Sequence models, but with Use-Case connected to Use-Case, rather than object instance method call connected to object instance. Figure 2.4 is an example of a use case scenario diagram from the Berenbach Book [BPKR09]. The scenario technique is good for describing a transaction type of requirement as it shows the ordered and complete set of actions.

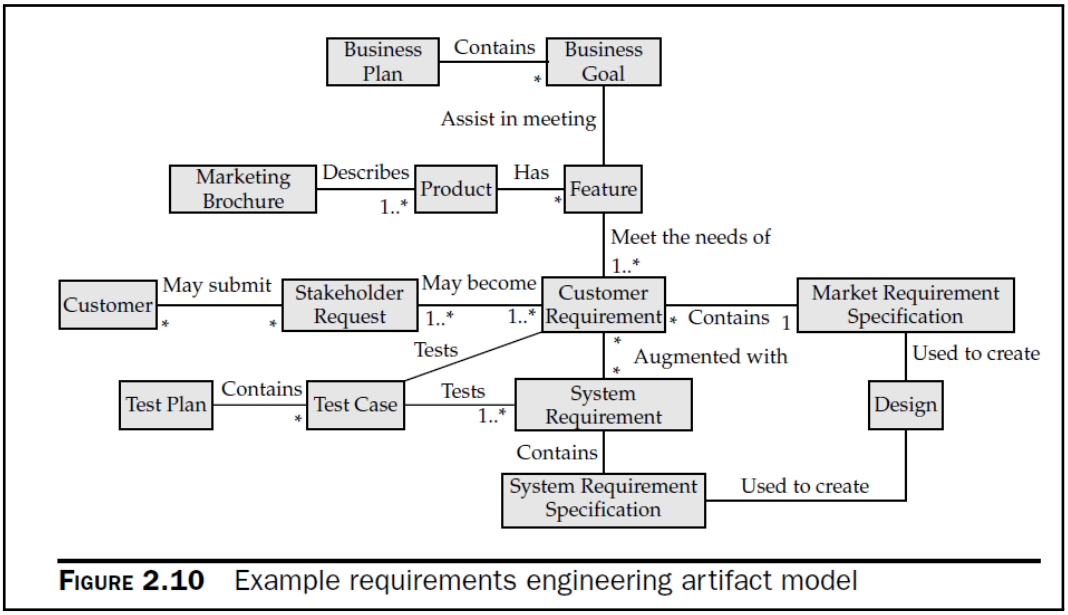


Figure 2.3: REAM Example.

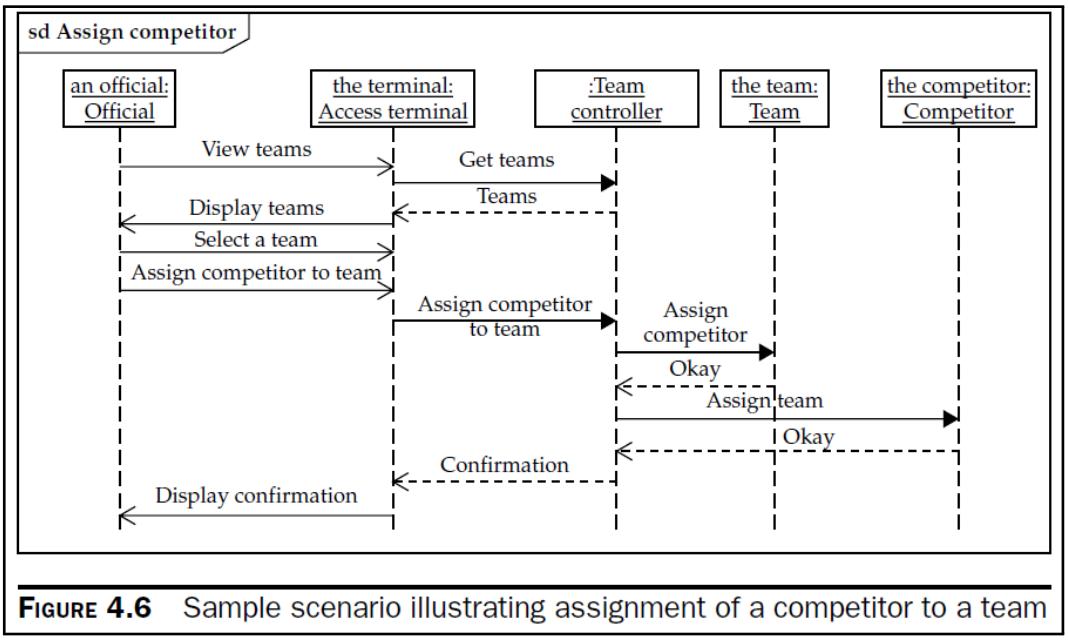


Figure 2.4: Use Case Scenario Example.

There is a strong emphasis on customer communication. Visual techniques such as models, diagrams, and tables are suggested. These are also used in the Berenbach book methodology, but they do not provide a framework to help with decisions on when and how to apply the visual techniques.

The REAM type of model is a goal that should exist in the SAMEM definition.

GOAL-13: A metamodel is needed for the artifacts and process to ensure a rigorous methodology.

2.1.3.2 Limitations in the Berenbach Book

The use of the REAM metamodel and the requirements artifact models unfortunately does not extend the idea of modeling to specifying the requirements as models, but only as text.

The scenario models are a strong part of the process; however, UML Activity Models are not used. In many of the book examples an activity model would communicate the control and data flow in more business detail without the need to make design assumptions about objects. In the scenario diagram in Figure 2.4 there are design assumptions made such as the existence of an access terminal object.

Because of the text-based requirements expression for both functional and nonfunctional requirements, there is a continued discussion of prioritization and ranking of requirements. The prioritization and ranking action are driven by the requirements engineer, but are the decisions of the stakeholders. Because many requirements are the suggestions of the stakeholders without limits on the scope, the two filtering actions of prioritization and ranking must be used to reduce the requirements volume, which take time and are additional accidental complexity.

The process described in the book utilizes brainstorming as an elicitation technique. The downside of brainstorming is a plethora of often marginally correct or relevant requirements. Filtering these requirements out is additional work. Other techniques can be used to avoid generating marginal requirements, which are part of the SAMEM.

While there are aspects in the Berenbach methodology that clearly improve the quality of the requirements, there are many processes and actions that increase the quantity and do not speed up the process. A primary source of increased quantity is the brainstorming elicitation of requirements. As mentioned earlier, requirements of marginal quality are generated which take time to evaluate and either improve or remove. The REAM helps with quality, but is additional modeling work that does not create a more compact requirement specification.

The process as described in the book is clearly more of a sequential, phase-based approach. There is no mention of iterative & incremental build-up and verification of either the analysis model or the text requirements.

The following repeated goal is needed to avoid limitations in the approach described in the Berenbach book.

GOAL-5: The artifacts should maximize the use of visual models for communication and compactness.

2.1.3.3 Berenbach Book Summary

Relative to the goals of this thesis, the proposals from [BPKR09] address only in part better stakeholder and development team communication and requirements quality. The techniques and processes suggested in the book do not help with achieving a more compact requirements specification, reducing the team size, speeding up the project process, or compatibility with agile methods.

Although not applied in the case studies associated with this thesis, the Use-Case scenarios models could be a valuable addition and in some situations provide better communication than a UML Activity Model.

2.1.4 Requirements in Engineering Projects

In [FM15] an engineering project approach defines the context for requirements elicitation and management methodology (Fernandes Process). While there is a chapter on software engineering, the tone of the book casts everything in the light of a more general engineering project approach. The definition of project (defined in the Chapter 2 introduction) in this book independently matches the definition used by the SAMEM.

2.1.4.1 Strengths of the Fernandes Process

[FM15] contains several engineering process principles that are the same as engineering principles used in the design of the SAMEM. The first principle is that process flexibility is needed and results from a feedback mechanism. The assumption in the book is that an engineering project is started to create a new solution. The rationale for process flexibility is that in creating a new solution there are unknowns which must be solved. The task flow and tempo of the project needs to adjust to the discovery of the problems and to the design needs of the solutions.

The second principle is that a visible balance must exist between the decomposition of the problem into smaller parts and a view of the solution as a whole. The decomposition activities explore the nature of the problem. The holistic view of the solution restates the intent and purpose. Exploration of the details and regular assembly of them are an important feedback mechanism [Bro10], [Win96], [Pet96].

The problem of communication is recognized. Domain-specific vocabularies and lack of a translation between them is part of the problem. Establishing a translation is difficult, because at the beginning the requirements engineer or the stakeholders do not know what they do not know. The attempts to describe the solution features and intent often come down to a struggle for the proper words and expressions. There are starting questions suggested for the elicitation work.

The strength of this book is in the collection of various sources of foundational definitions for projects, engineering, and requirements. The term definitions contained within this book match term definitions independently developed for the SAMEM, but in many cases the definitions vocabulary from [FM15] are an improvement.

The strengths in the Fernandes book, which should be maintained, lead to the following goals.

GOAL-14: The methodology should support multiple feedback mechanisms.
GOAL-15: The methodology should allow for multiple requirement artifacts to match the different communication needs of the different levels of abstraction needed during the project process.

2.1.4.2 Limitations of the Fernandes Process

The major limitation in [FM15] is the lack of a process or methodology to connect the definitions into a coherent and workable whole. While there are some small process snippets here and there throughout the book, they are mostly used in helping to define a term. For example, there is a five-step process for requirements elicitation as shown in Figure 2.5. This is the extent of the process depth and is considerably less than provided by the Volere Methodology [RR99].

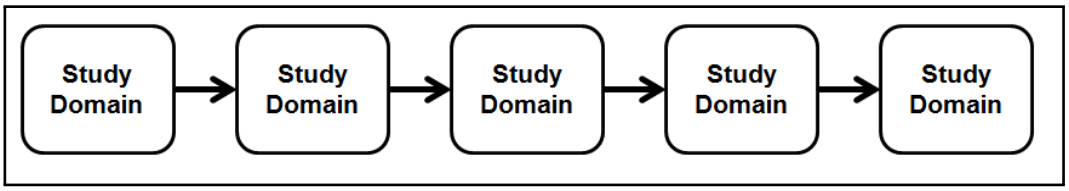


Figure 2.5: Fernandes Process Example.

GOAL-16: The methodology project process must ensure that progress towards a solution is accomplished at a reasonable speed.

2.1.4.3 Summary of the Fernandes Process

In general, this book can be described as a survey book. It brings together various definitions, but there is little on how to turn the definitions into a methodology. There is nothing in this book that directly addresses any of the goals of this thesis; however, the book does have improved phrasing of several principles behind the SAMEM. One founding principle behind the SAMEM was the need to control the level of abstraction during the project work, but was not optimally stated. The phrase from [FM15] that states it better is:

> "being aware of the abstraction level in which the project is developed is crucial to control its complexity."

2.1.5 System Modeling Language

The Object Management Group (OMG) System Modeling Language (SysML) [OMG15a] is a specification for the modeling of systems consisting of mechanical, electrical, and

software components. Part of the specification covers the modeling of requirements. The SysML specification reuses part of the Unified Modeling Language (UML) [OMG15b] specification, modifies definitions, and adds extensions.

2.1.5.1 Definitions from the SysML Specification

The SysML extensions primarily impact UML classes and dependencies. The SysML extension of *View Model* and *Viewpoint* are used to partition the model with respect to a particular set of concerns from some of the stakeholders. A *View Model* or *View* is the model that represents the *Viewpoint*. A *View* conforms to a single *Viewpoint*.

The SysML includes an extension to the UML Comment called a *Rationale*. The *Rationale* is intended to capture in text the design and other decisions within the model. Similar to *Rationale* the SysML extensions have a *Problem* type of comment. The *Problem* is used to capture in text a deficiency, limitation, or failure in one of the model elements to satisfy a requirement. The third extension of the UML Comment is called *ElementGroup*, which is used to group multiple heterogeneous elements by allowing references from the *ElementGroup* to multiple elements.

A major portion of the SysML specification is the extension to deal with modeling requirements. The model supports a text-based requirement. There are additional relationships to support the creation of requirement hierarchies, reuse of requirements across projects, subrequirements, and a master-slave requirement. The requirement hierarchies and subrequirements use the UML namespace containment mechanism. A consequence of the namespace mechanism is that a new modeling mechanism is needed to support reuse of common requirements, like regulations, across disjoint projects. The new mechanism is the *slave* requirement, which is a read-only copy of the *master* requirement.

The refine relationship is used to connect other model elements to a requirement to further refine its meaning. For example, an Activity Model can refine the meaning of a requirement by defining a more complex behavior.

2.1.5.2 Limitations of the SysML Specification

There are two major limitations of the SysML specification for supporting a requirements process. The first is the definition of a requirement as text. As stated in the specification, this was primarily done to provide compatibility with existing requirements management tools. While the requirement can be related to other modeling elements, it is at the core a simple text statement. This means a large number of requirements are needed for a large system [LS87].

The second major limitation is the use of the UML diagramming notation. Other research [MvH08] has shown that the simple visual forms of the UML graphical notation limit communication. There is an extension that shows the text content of several requirements in a table, which is more compact but rather limited in usefulness.

A minor limitation is maintaining the namespace containment mechanism for requirements. By keeping the namespace limitation, another mechanism, a slave requirement, is introduced, which adds accidental complexity.

2.1.5.3 Summary of the SysML Specification

The SysML specification does little to address the goals of this thesis. As a specification it is a definitive document rather than a prescriptive document of how to use the modeling artifacts it defines. There are several examples within the specification that give indications of potential use. The SysML definition can help with the development of a modeling foundation for the SAMEM.

The next methodology goal preserves the valuable idea from SysML of recording design decisions with the modeling artifacts.

GOAL-17: The artifact evolution trace should have possibilities for recording the rationale for the artifact improvement.

The following goal arises from preventing the main limitation seen in the SysML specification of text-based requirements.

GOAL-18: The visual artifacts must not be limited by existing notations such as UML, alt-hough, when appropriate, existing notations should be preferred.

2.1.6 V-Model

The V-Model is a term that covers a range of models [V-M16c], [V-M16a], [V-M16b], Figure 2.6. The purpose of the V-Model is to provide a structure for the planning and execution of a project. The V-Model can be used across a broad range of projects; however, a special variant for software engineering was developed, V-ModelSE [V-M16b].

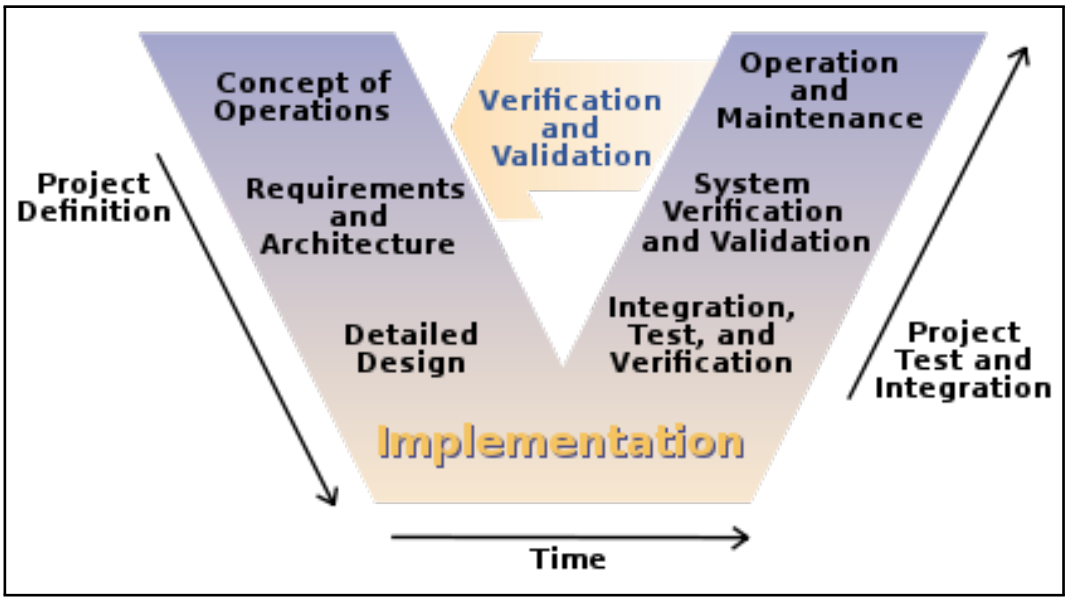


Figure 2.6: V-Model Typical Image.

Many consider the V-Model to be a folded sequential, phase-based project execution plan. The folding highlights the correspondence between design and development activities and their associated testing and verification activities.

2.1.6.1 Summary of V-Model

The V-Model is widely considered a project planning model primarily for project managers. It is rigid and the high level of abstraction defeats meaningful and consistent understanding of project details. It does not address any of the goals for this thesis, especially compatibility with agile development methods, faster project processes, improved requirements quality, and better communication.

2.1.7 Software Engineering Method and Theory

Software Engineering Method and Theory (SEMAT) is a recent (2009) initiative to improve the discipline of software engineering and make it more rigorous [JEJ12], [SEM98], [SEM16]. The SEMAT initiative is divided into four related areas of work: the Practice area, the Education area, the Theory area, and the Community area. The SEMAT work is supported by and has contributed to an OMG specification: Essence – Kernel and Language for Software Engineering Methods [OMG12b]. For this comparison SEMAT and Essence will be looked at as a single effort.

2.1.7.1 Definition of the SEMAT Initiative

The SEMAT initiative is relatively new and therefore it is still developing. For the purposes of this thesis, the Theory and Practice areas are important. The SEMAT Theory area has the objective of creating a general theory of software engineering. As with other theories [JEJ12], the software engineering theory should support good project decision making by expressing clear rationale for the choice and the objective evaluation of project alternatives in a predictive manner. There is little content in this area at this time [SEM98].

The Practice area of SEMAT has the OMG ESSENCE specification [OMG12b] as an initial realization of practices. There is tool support for the ESSENCE specification in the *EssWork Practice Workbench* [Essb]. The ESSENCE specification defines a kernel and language which enables the description of software engineering methods and practices. The definition of methods and practices enables different approaches to be compared, evaluated, adapted, simulated, and measured for both practitioners and researchers. The ESSENCE specification consists of a simple layered architecture as shown in Figure 2.7.

The ESSENCE architecture *Method* is a composition of practices, which describes what is actually done in the project. A *Practice* is a repeatable approach to accomplishing a specific objective. It provides a systematic and verifiable way of addressing the work. The *Essence Kernel* has the essential elements of software engineering methods.

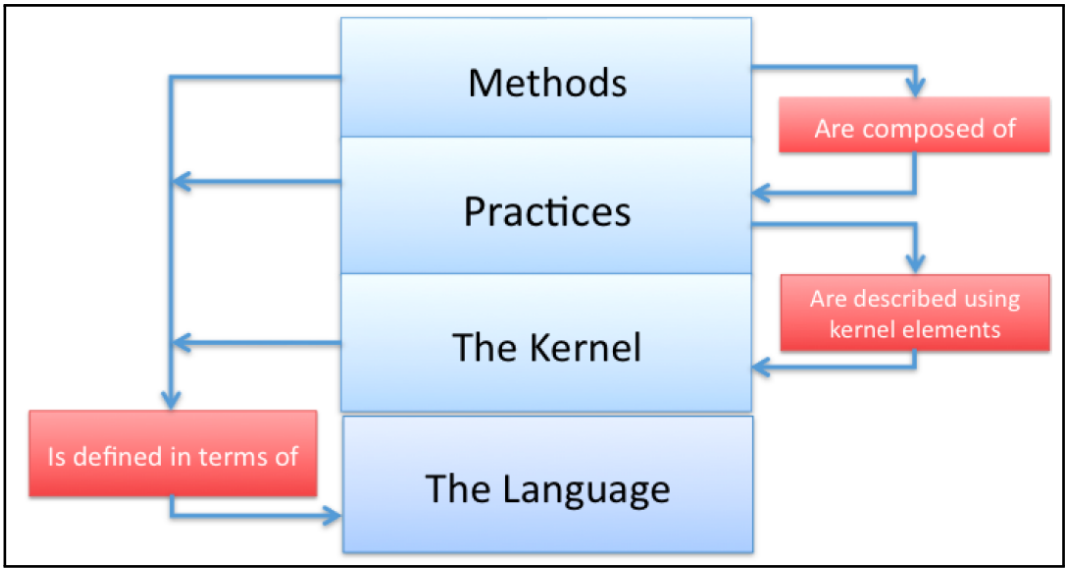


Figure 2.7: ESSENCE Four Layer Architecture.

Through the domain-specific *Language of Essence*, the methods, practices, and kernels are defined.

2.1.7.2 Summary of the SEMAT

The SEMAT initiative does not aim to define any specific methodologies comparable to the SAMEM at this time. It does have definitions and a tool with guidelines on how to define a method or a practice. In the evaluation (Sub-chapter 7.4) the SAMEM will be compared to the SEMAT and ESSENCE approach. This will provide feedback on the abilities of ESSENCE and possible improvements to the SAMEM as a contribution to the SEMAT goals. The goals of the SAMEM are in alignment with the goals of SEMAT.

While the following repeated goal for the methodology is rather generic, it is useful as a reminder that research is ongoing in improving software engineering and, insofar as possible, the SAMEM should not prevent new ideas from being incorporated.

GOAL-11: The methodology should support the incorporation of state-of-the-art software engineering results.

2.1.8 Software Process Engineering Metamodel

The Software Process Engineering Metamodel (SPEM) [OMG08] is an OMG specification for a software and systems process engineering metamodel. The SPEM is a metamodel and a UML 2 profile which reuses many of the other OMG specifications such as UML 2 [OMG15b], the Meta-Object Facility (MOF) [OMG14b], and the UML Diagram Definition Specification [OMG12a].

The purpose of the SPEM is to model a wide variety of processes and to avoid exclusion by having too many features or constraints. The focus is on the modeling of development projects. The SPEM provides additional information structures needed for modeling actual development processes beyond the UML Activity Models and Business Process Model and Notation (BPMN) [BPM16].

2.1.8.1 Summary of the SPEM

The objective of the SPEM specification is to support the creation of metamodels and models of processes that are beyond the capabilities of the UML Activity Models and BPMN. The SAMEM in its current form does not need any of the capabilities of the SPEM specification. The process modeling abilities of the UML Activity Models or BPMN are sufficient for all the SAMEM needs. With the expansion of the SAMEM to other domains, it is conceivable that the SPEM capabilities could be needed. However, an evaluation between using SPEM or SEMAT to extend the SAMEM is required before making that decision.

2.2 Survey of Existing Agile Methodologies and Processes

In the following evaluations and comparisons the focus will be on the requirements portion of the agile methodologies and processes. Some small consideration will be given to the transition from requirements elicitation to design work.

2.2.1 Agile Modeling

Scott Ambler [Amb02] describes approaches to Agile Modeling (AM) in the development of software solutions. He defines AM as:

> "... a chaordic, practice-based methodology for effective modeling and documentation of software-based systems."

The term chaordic comes from Hock [Hoc99] and is a combination of the chaos of simple modeling practices and the order inherent in software modeling artifacts. Ambler claims that there are two primary reasons to model: 1) to understand what should be built and 2) to aid the communication with the development team and the stakeholders. His modeling purpose rationale is echoed by [FM15], [BPKR09], [Pet96].

2.2.1.1 Agile Modeling Methodology Description

AM aligns with the Agile Software Development Alliance (www.agilealliance.org) manifesto. The Agile Alliance (AA) values are:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.

- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

There are another 12 principles that the AA promotes and AM accepts but will not be detailed here. The models that AM creates are "good enough" models. Good enough models exhibit the following traits:

- Agile models fulfill their purpose – whether the purpose is better understanding or better communication, that purpose is achieved.
- Agile models are understandable – by the intended audience, so language, form, and organization all play a role.
- Agile models are sufficiently accurate – the models do not have to be 100% accurate or complete as long as they are accurate enough for the purpose.
- Agile models are sufficiently consistent – missing details or different words, for example synonyms, do not necessarily mean crippling inconsistency.
- Agile models are sufficiently detailed – the models have enough detail to meet the purpose. An application of information hiding according to Parnas [Par72].
- Agile models provide positive value – the benefit of having the model should outweigh the cost of producing it.
- Agile models are as simple as possible – this drives toward limiting the detail and using a clean effective notation.

Ambler in [Amb02] contains many practical examples of applying the AM values, tips to be effective at agile modeling, and shows a variety of good models. There are several tips that Ambler mentions that also appear in the SAMEM. Use modeling standards, such as UML, but do not let following the standard have a negative impact on communication. Apply patterns gently, which means that there might be an opportunity to use a pattern, but go slowly as new information can invalidate the pattern.

There are several good ideas from the agile approach that the SAMEM should preserve and if possible strengthen. The goals immediately listed below are intended to preserve the good agile ideas.

GOAL-19: The visual models must be good enough to achieve the purpose within the current context or step of the project process.

GOAL-20: The methodology should have a set of principles that support adaptation to various domains and provide a checklist for rigor.

GOAL-21: The models used for artifacts in the methodology should support multiple small iterations.

2.2.1.2 Summary of Agile Modeling

Agile Modeling (AM) supports the same goals as the SAMEM, especially the goals of better communication, better requirements quality in a more compact form, faster project process, and compatibility with agile solution (code) development processes. Both AM and the SAMEM have a set of adaptable principles used to ensure quality of work, although there are some differences. The SAMEM principles are explained in Sub-chapter 3.2.

The main difference is that in Agile Modeling a looser, free-form process approach is taken. AM strives to be non-procedural. In that approach some guidance is lost. Since the SAMEM is targeted at medium to large projects, a rough roadmap of how to proceed is provided. A lesson learned from the case studies shows that the project process roadmap helps with establishing good customer expectations [TFR05].

While Ambler recognizes that models are viable forms for requirements, they can be taken further than he discusses in his book [Amb02]. Like RUP, AM starts with UML Use-Case Models as the primary requirements model. The meaning of a UML Use-Case Model is inherently shallow and needs additional modeling artifacts and/or text to provide real understanding.

2.2.2 eXtreme Programming (XP)

The agile software development practice called eXtreme Programming (XP) [Bec00], [XP] takes observed good programming practices to extremes. The origins of XP date back to the 1990s. XP advocates frequent releases and short development cycles, which are intended to improve productivity and introduce checkpoints for the customer verification of the product. XP is an iterative & incremental development process.

2.2.2.1 XP Definition

XP is described as a software-development discipline that organizes people to be more productive. For XP the *code* is the important solution development product. The code can start simple to communicate the essence of the problem, and then evolve to the full functionality. Continuous testing is part of the XP discipline, from unit tests that should be written before the code to frequent system-wide integration tests. Along with the simple code, a simple design structure that limits dependencies is advocated.

Frequent verbal communication with the customer and common metaphors rather than documentation is preferred. XP encourages starting with the simplest solution, then gradually extending it according to the customer's reviews. The goal is to avoid building something that is not needed or incorrect in the customer's eyes. Since code reading is a type of testing, XP advocates programming in pairs to have the code reading test happen while the code is being created.

The requirements for XP are expressed as user stories. The user stories form the basis for the iteration planning activities. A critical aspect of the XP approach is the demand

that a customer representative is available at all times to clarify a user story and to verify that the implementation is functionally correct.

2.2.2.2 Summary of XP

One of the biggest impacts XP has had on software development is the questioning of the sequential, phase-based project approach. The success of XP and several of its principles, especially to make frequent small releases, that the project is divided into iterations, simplicity, that no functionality is added early, and acceptance tests are run often, were influences on and rationale for the design of the SAMEM.

There are several daily work practices from XP that the SAMEM should support or allow. The notion of automatic testing as much as possible is important to the goal of always having working code prototypes. The automatic testing of the requirement artifacts and design artifacts against the meta-definitions should be possible. The idea of pair programming, a second set of eyes during the creation activities, should be compatible with the development of any of the artifacts, not just code. The objective of pair programming is to prevent errors or mistakes at the earliest stages in the creation process. Within the budget constraints of the project, these good working practices should be used.

There are several weaknesses in XP, especially as the project size scales up. The user story as the requirement cannot handle non-functional requirements easily. In a larger project there can be several customers, each with their own slightly different views, and there is often the problem of continuous customer availability. Without continuous customer availability a means for asynchronous communication is needed, which the SAMEM provides via models. There is an upper limit to the team size because of the emphasis on verbal communication and the effects of the Mythical Man-Month [Bro95].

XP does support to a certain extent the thesis goals of a faster project process, compatibility with agile code development methods, and smaller team size. It does not contribute to improved requirements quality in a more compact form, better team communication, and better stakeholder communication.

GOAL-22: The methodology should support frequent small releases of project deliverables to support verification of progress.

2.2.3 SCRUM

SCRUM is an iterative & incremental agile software development process [Coh10]. It uses the idea of a sprint, which normally is one week to one month long, to deliver a potentially shippable increment of the solution. A potentially shippable increment is integrated, fully tested, end-user documented, and could be shipped to the customer.

2.2.3.1 SCRUM Process Definition

SCRUM uses a sequence of sprints to deliver ever increasing completeness of the solution. Each sprint is limited by time, usually a minimum of one week to a maximum of one

month in duration. The sprint implements some of the items (requirements) in the backlog. An item is how the requirements for the solution are collected and documented. The backlog is the term for the items yet to be implemented. As the customer reviews the completed sprint deliverable, new items can be added to the backlog, removed from the backlog, modified, or their priority changed.

The SCRUM process depends on face-to-face communications with minimal written documentation. Before a new sprint starts, a *sprint planning event* happens with all the team members, which takes items from the product backlog and places them in the *sprint backlog* of work to be done. The *product owner* role in SCRUM represents the stakeholders and writes items, prioritizes them, and adds them to the product backlog. The developers evaluate the items suggested for the sprint and give estimates of the duration involved to complete each item. Based on the total effort involved, requirements can be added to or removed from a sprint backlog. Each day the team holds a daily *SCRUM* (stand-up) meeting to evaluate the progress and possible impediments to meeting the sprint goal. The *scrum master* keeps track of the issues, progress, and is responsible for enabling the developers to complete their work.

The product backlog is the ordered list of items (requirements), which consists of features, defect fixes, knowledge acquisition, nonfunctional requirements, and anything else that is needed for a viable product. The requirements are normally in an ordered story format and define what will be delivered. In many cases the items are user stories, but can be use cases or anything else that clearly expresses the customer needs. SCRUM is neutral on requirement techniques and allows multiple expressions of requirements to be used.

2.2.3.2 Summary of the SCRUM Process

The SCRUM approach does not insist on any particular requirements form or process. It is requirements neutral. SCRUM is an operation process intended to control and manage the development steps. It can be used in conjunction with the SAMEM as the two are disjoint in their areas of focus. The requirements modeling of the SAMEM can feed into the project backlog items of SCRUM, which fulfills one of the thesis goals.

GOAL-23: The requirements elicitation portion of the methodology should be consistent with modern implementation best practices.

2.3 Limitations of Text-based Requirement Formats

2.3.1 A Diagram is (Sometimes) Worth 10,000 Words

In the paper by Larkin and Simon [LS87], the cognitive value of a diagram or drawing over text is explained. The text-based representations are sentential, meaning the expressions form a sequence corresponding to the sentences in a natural language. In a diagrammatic representation the expressions correspond to information that is stored at a particular locus in the diagram including information about related information.

The definition of *better* for a comparison is defined in terms of the informational and computational equivalence of the representations. Informational equivalence is defined as all the information in one representation is inferable from the other. Computational equivalence is defined as informationally equivalent and the inferences drawn from one representation can be easily drawn from the other. *Easily* is not a precise term as there are many dependencies on the forms and organization within the representations, but it can be assumed for these arguments that the best possible sentential and diagrammatic representations are available for the computational equivalence comparison.

The human who is operating on a representation to understand it, communicate it, or refine it invokes three basic components of information processing: search, recognition, and inference. Searching a sentential structure (text) is done linearly over the list of items. The search action on a diagram is finding the correct two-dimensional location. Often searching is not just for one item but several related items. The computational cost includes finding the additional items. The ease of recognizing the information, both explicit and implicit, is directly related to the form. When the external form of the information matches known or internal forms (mental models) that a person has, then the information is easily recognizable. Once the searching has found the information and it is recognized, the process of inference can begin. The inference is drawing conclusions from or producing new information and this work is largely independent of the representation.

The conclusion that Larkin and Simon come to is that the major advantage of a diagrammatic representation is in the recognition effectiveness. This is especially obvious when the diagram is used both as a mental model and as external memory, i.e. a communication mechanism between people. There is also speculation in the paper that mental images play a role in problem solving.

The reasoning and explanations within [LS87] are the source for an important rationale for the maximum use of visual models by the SAMEM. The need to have the models developed jointly with the customer to establish a common visual vocabulary is supported by this thesis.

2.3.2 Cognitive Effectiveness of Visual Notations

Work by Moody and others [Moo09], [HK99] establishes some of the cognitive effectiveness criteria for visual notations. In [MvH08] an evaluation of the UML visual notation is performed and in [MHM10] the visual notation of *i**, the goal modeling notation is done. *i** is a goal modeling notation [Yu97], [Yu09], [Yu95].

There are several cognitive measures which allow for the evaluation of the communication effectiveness of visual notations. These measures are based in cognitive psychology and are listed in [Moo09]. In the evaluation Sub-chapter 7.3, these measures will be used to determine the cognitive effectiveness of the pragmatic Solution Overview Drawing (SOD) (defined in Sub-chapter 3.5.1) developed in the various case studies. The results of the effectiveness evaluation will be drawn on to suggest improvements to the SAMEM visual notations.

As two of the thesis goals are better communication with the stakeholders and development team, the cognitive effectiveness measures provide an objective evaluation. As

pointed out in [LS87], diagrammatic notations provide a more compact form of communication. The use of cognitive measures to evaluate the models will assure an effective compact requirements specification form.

GOAL-24: The methodology artifact representations should be consistent with the best practices of communication as measured through cognitive effectiveness.

2.3.3 User Requirements Notation

User Requirements Notation (URN) [URN12], [AM11] includes and is an extension to the *i** approach. The URN combines a Goal-Oriented Requirements Language (GRL) and Use-Case Map (UCM) which uses scenario paths for causal relationships among responsibilities. There are modeling concepts and graphical notations defined in the specification. Goal-oriented modeling places the focus on the *who* and the *why* for requirements specification.

The objective of the URN is to specify and perform early evaluation of telecommunication protocols and services. The specification is done at a level that abstracts out the details of the messages and the component architectures, which allows a simplified description of services.

Both URN and *i** have visual artifacts for some of the needs of requirements modeling [URN12], [AM11]. However, these notations also have cognitive shortcomings [MHM10]. The conversion from URN to a UML design model is done by hand through a heuristic algorithm. URN can help with communication goals for some of the requirements and potentially with a more compact form.

2.3.4 Proposals for Requirements Models

2.3.4.1 Requirements Visual Notation Improvements

In [CGHM13] the results of an empirical study on designing new symbols for the *i** notation are reported. The symbols were designed by novices in the discipline of requirements engineering rather than experts. The cognitive effectiveness measures of [Moo09] were used to evaluate the visual notations. There are improved symbols described in the paper that can be used instead of the standard notation of *i** or URN. The paper makes the claim that the novice users did a better job of creating a more comprehensible set of visual notations in 25 minutes than the requirements community has in 25 years. The novices were business and economic students with no knowledge of goal modeling or *i**.

The design of the new symbols used the sign production technique developed by Howell and Fuchs [HF68] to design military intelligence symbols. This technique uses some members of the target population of end users to design the visual symbols. The assumption is that the cognitive profile is the same, so that correct interpretation is high.

The comprehensibility of the new notation is measured through semantic transparency. Semantic transparency is defined as the meaning of a symbol being apparent from its

visual appearance alone. One of the keys to the improvement in visual communication was avoiding the curse of knowledge [HH08]. The curse of knowledge means that an expert has difficulty in thinking as a novice because of their greater understanding.

One of the important lessons learned in the case studies (see Sub-chapter 4.4.1.5) was having flexibility in changing the notation when stakeholder communication suffered. The refinement of much of the visual notation in the SAMEM is in cooperation with the stakeholders, who are novice requirement specification users.

GOAL-25: The methodology should allow for the ad-hoc creation of artifacts for domain adaption and communication improvements.

2.3.4.2 Requirements Modeling Language

The Requirements Modeling Language (RML) [BC12] is a collection of 22 industry best practices that have been used in an ad-hoc fashion over many years to visually model requirements. The focus of the modeling techniques in RML is the capture of business value and representing the solution from the end-user viewpoint. For the RML approach, the models are a means to reach the end stage of a list of requirement statements. The RML models are organized into four categories: objectives models, people models, systems models, and data models. The RML also includes processes for using the various models and linking them together.

In addition to the visual models created in the RML methodology, several other important solution relevant definitions are necessary. The vision of the *Product Concept* is needed to form a central pillar for the models. RML recommends compiling a list of *Guiding Principles*, which are important characteristics of the solution that should be maintained. For example, adhering to specific government regulations is a guiding principle. Success Metrics should be defined for the solution to verify the models and the final solution to meet the stakeholder's expectations.

While there are several interesting models and modeling processes described in [BC12], the book fails to mention and draw contrast to other well-known methodologies, such as RUP or Volere. The UML is mentioned as not being useful, but at least eight of the models have direct UML representations. Although not part of the work in this thesis, the visual models are simple graphical forms like rectangles and circles, which would have a low cognitive effectiveness score.

2.4 Summary of State of the Practice

There are several well-known software engineering methodologies with requirements elicitation and evaluation as part of the complete development approach. Multiple requirements engineering approaches only focus on elicitation through talking with the stakeholders. However, while each supports some subset of the goals of this thesis to one degree or another, no existing methodology supports them all.

By combining the strengths of effective visual modeling when possible, agile principles, and filling in the gaps, the thesis goals can be achieved. The largest gap is the lack of a

guiding process that helps progress the project from very abstract work in the beginning to more concrete work in design and implementation. It is almost universally expressed in all the other methodologies and approaches that requirements are elicited by talking with the stakeholders.

But the big question that is left unanswered is: "How do you talk with the stakeholders to get the requirements?" The core of the SAMEM is directed towards providing a pragmatic answer to that question.

At the 2013 European Software Engineering Conference a question was asked whether there was an empirical result that determined if text or visual models were more effective [GCH13]. The response described in [GCH13] is about a challenge at the 2009 Requirement Engineering Conference to have different groups use their modeling techniques to respond to a fictitious problem. The techniques of *i**, URN, text, UML, rich pictures, information FLOW modeling, and formal methods were tried. The audience observed how the teams worked, responded to changes, and then voted for the best technique. The conclusion is that no single modeling technique is best and that questions about which approaches to use, for what purposes, and how to combine the models remain. The SAMEM attempts to answer some of these questions.

2.4.1 Summary of Methodology Goals

In Sub-chapter 1.1, the discussion of the software engineering project problems produces a corresponding set of high-level goals for the thesis. The high-level goals are supported by the goals defined in this sub-chapter and are restated below for convenience with unique identifiers.

HL-GOAL-1: Better stakeholder communication with respect to the requirements specification and project progress.

HL-GOAL-2: Better development team communication.

HL-GOAL-3: Improved overall requirements quality in a more compact format.

HL-GOAL-4: Faster project process.

HL-GOAL-5: Smaller project team size, especially for the development team.

HL-GOAL-6: Compatibility with agile methods for solution (code) development.

In the evaluation of the related work and state of the practice, a number of goals are identified from either the strengths or the limitations. The methodology goals will be evaluated against the SAMEM definition in Sub-chapter 7.1.4. The methodology goals are gathered here.

GOAL-1: The methodology should provide accurate communication mechanisms.

GOAL-2: The methodology should provide project process risk mitigation mechanisms.

GOAL-3: The methodology should provide traceable artifact evolution.

GOAL-4: The methodology and artifacts should be compatible with an iterative & incremental project process.

GOAL-5: The artifacts should maximize the use of visual models for communication and compactness.

GOAL-6: The methodology should be adaptable to individual project process needs.

GOAL-7: The iterations should be small for project management purposes.

GOAL-8: An iteration should have a verification step of some kind to assure that the work done is correct.

GOAL-9: Flexibility in the visual model artifacts is necessary for communication optimization.

GOAL-10: The methodology should ensure completeness.

GOAL-11: The methodology should support the incorporation of state-of-the-art software engineering results.

GOAL-12: The visual modeling and textual artifacts need to have consistent presentation to optimize communication.

GOAL-13: A metamodel is needed for the artifacts and process to ensure a rigorous methodology.

GOAL-14: The methodology should support multiple feedback mechanisms.

GOAL-15: The methodology should allow for multiple requirement artifacts to match the different communication needs of the different levels of abstraction needed during the project process.

GOAL-16: The methodology project process must ensure that progress towards a solution is accomplished at a reasonable speed.

GOAL-17: The artifact evolution trace should have possibilities for recording the rationale for the artifact improvement.

GOAL-18: The visual artifacts must not be limited by existing notations such as UML, alt-hough, when appropriate, existing notations should be preferred.

GOAL-19: The visual models must be good enough to achieve the purpose within the current context or step of the project process.

GOAL-20: The methodology should have a set of principles that support adaptation to various domains and provide a checklist for rigor.

GOAL-21: The models used for artifacts in the methodology should support multiple small iterations.

GOAL-22: The methodology should support frequent small releases of project deliverables to support verification of progress.

GOAL-23: The requirements elicitation portion of the methodology should be consistent with modern implementation best practices.

GOAL-24: The methodology artifact representations should be consistent with the best practices of communication as measured through cognitive effectiveness.

GOAL-25: The methodology should allow for the ad-hoc creation of artifacts for domain adaption and communication improvements.

The relationships between the high-level goals and the goals can be represented graphically using the URN notation [URN12]. The models are specifically the portion of URN that is derived from i* for modeling of goals and their relationships. In URN terms a SAMEM goal is a Softgoal, which means that there are not any clear-cut or quantitative criteria for determining that they are achieved. Whether a Softgoal is achieved is an interpretation of the modeler. URN offers several types of relationships between its modeling components. For the URN models below only the Contribution relationship is used with the qualitative contribution value of some positive. Since all the contribution relationships have the same value, it is not shown in order to keep the images more readable. Figure 2.8 through Figure 2.13 show the graphical relationship of which goals support which high-level thesis goals. A goal can contribute to fulfilling more than one high-level goal. To conserve space, only the goal identifiers are used in the URN images with keyword descriptions.

2.4.1.1 HL-GOAL-1 Relationships to Goals as URN Model

Figure 2.8 shows the goals that at least in part contribute to fulfilling the ***HL-GOAL-1***, which is the high-level goal of better stakeholder communication of the requirements specification and the project progress.

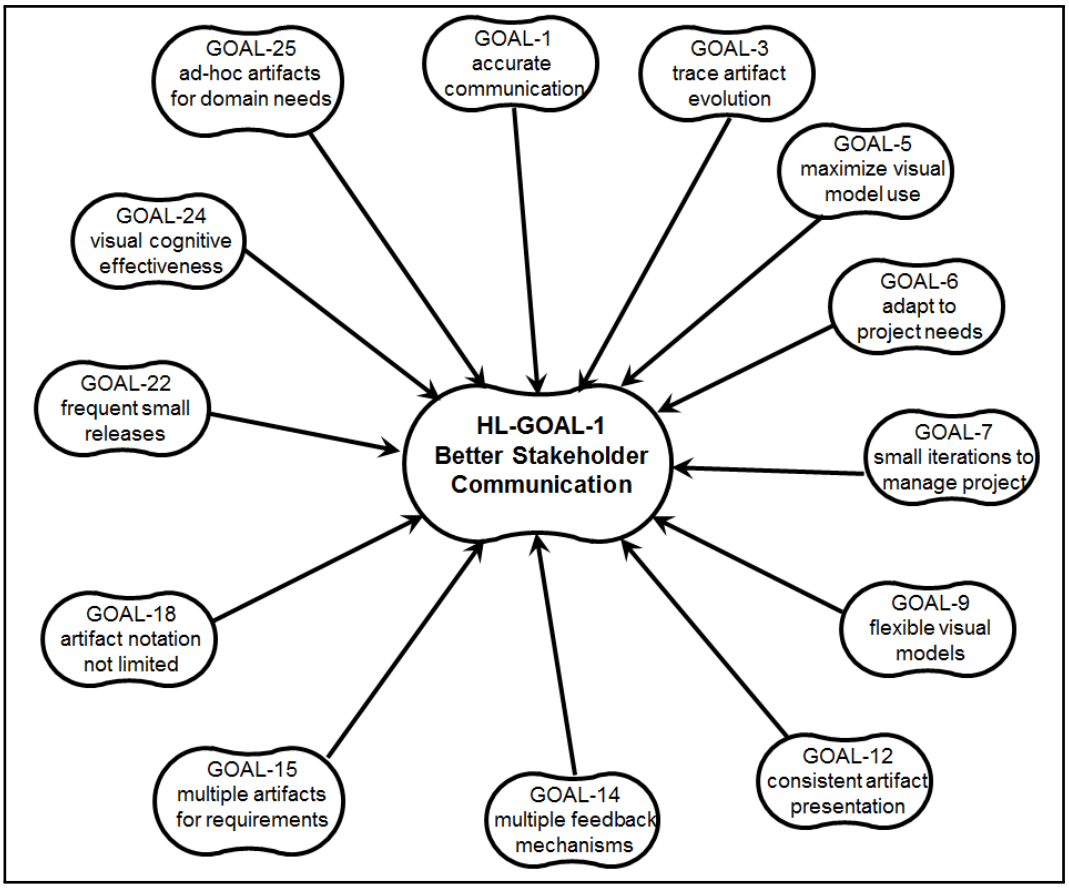


Figure 2.8: URN Goal Model for HL-GOAL-1.

2.4.1.2 HL-GOAL-2 Relationships to Goals as URN Model

In Figure 2.9, the goals that contribute to the achievement of ***HL-GOAL-2***, which is better development team communication, are shown. The goals needed for achieving ***HL-GOAL-2*** are a subset of the goals needed for ***HL-GOAL-1***. This is not surprising as both are focused on communication.

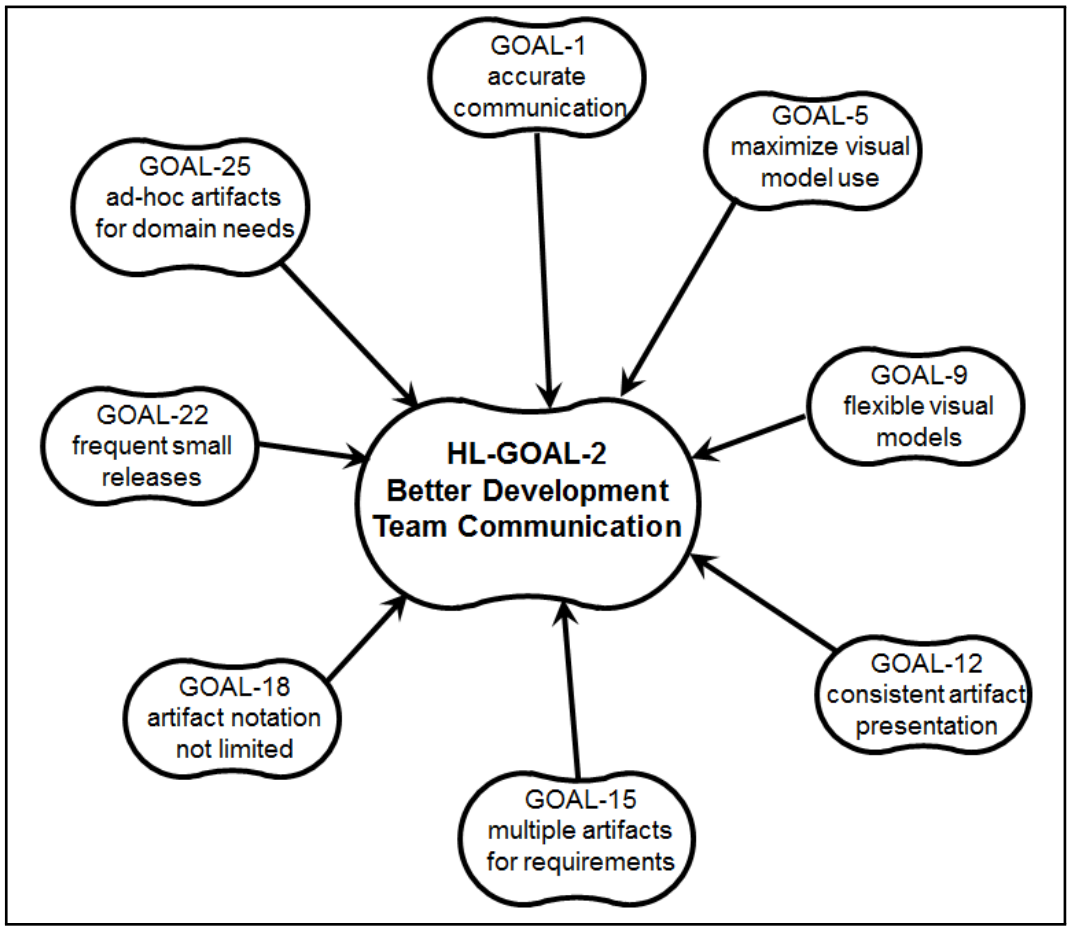


Figure 2.9: URN Goal Model for HL-GOAL-2.

2.4.1.3 HL-GOAL-3 Relationships to Goals as URN Model

Figure 2.10 lists the goals that contribute to fulfilling the ***HL-GOAL-3***. This high-level goal is concerned with improving the overall quality of the requirements specification.

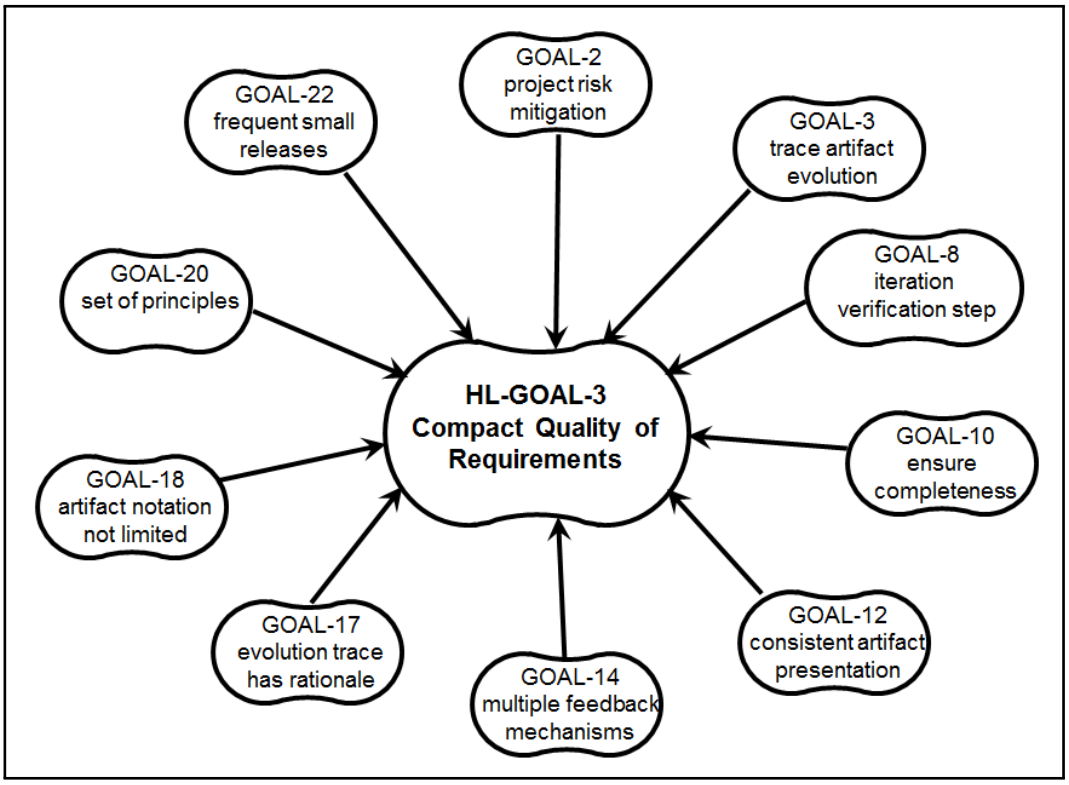


Figure 2.10: URN Goal Model for HL-GOAL-3.

2.4.1.4 HL-GOAL-4 Relationships to Goals as URN Model

The content of Figure 2.11 shows the contributing relationships of the goals to the ***HL-GOAL-4***, which is the achieving of a faster project process.

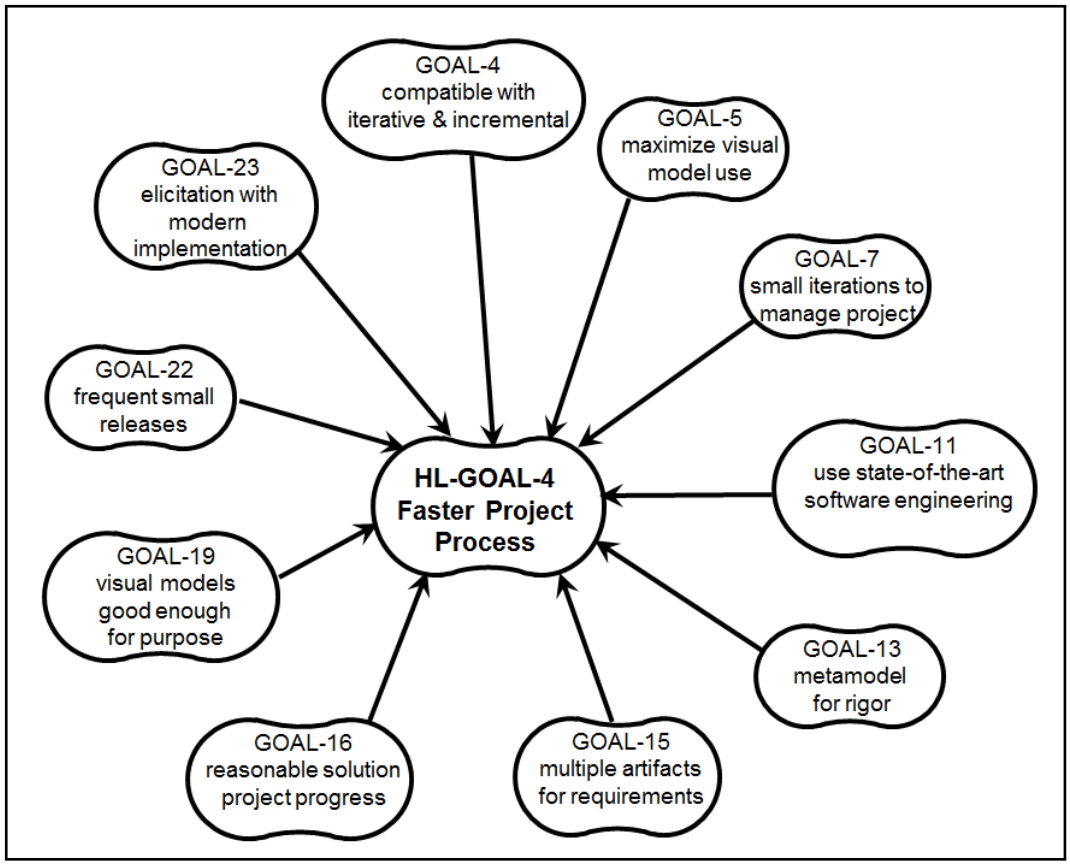


Figure 2.11: URN Goal Model for HL-GOAL-4.

2.4.1.5 HL-GOAL-5 Relationships to Goals as URN Model

In Figure 2.12, the goals that impact in a positive manner the high-level goal, ***HL-GOAL-5***, of a smaller development team size are shown.

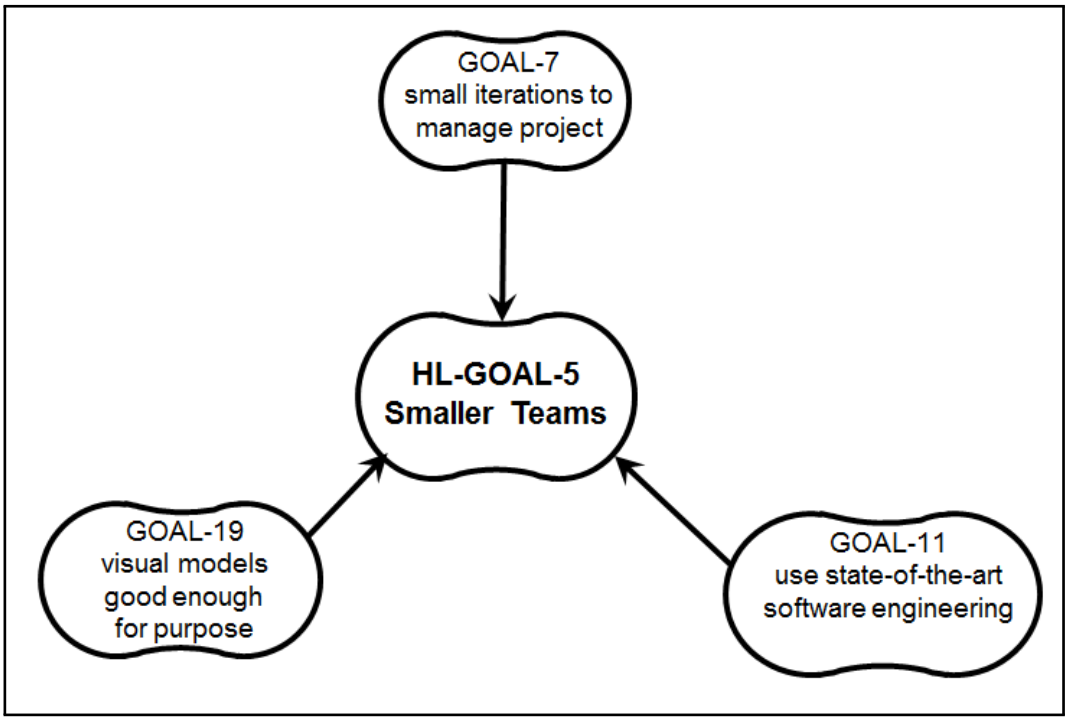


Figure 2.12: URN Goal Model for HL-GOAL-5.

2.4.1.6 HL-GOAL-6 Relationships to Goals as URN Model

The goals that contribute to ***HL-GOAL-6***, compatibility with agile methods for solution development, are shown in Figure 2.13.

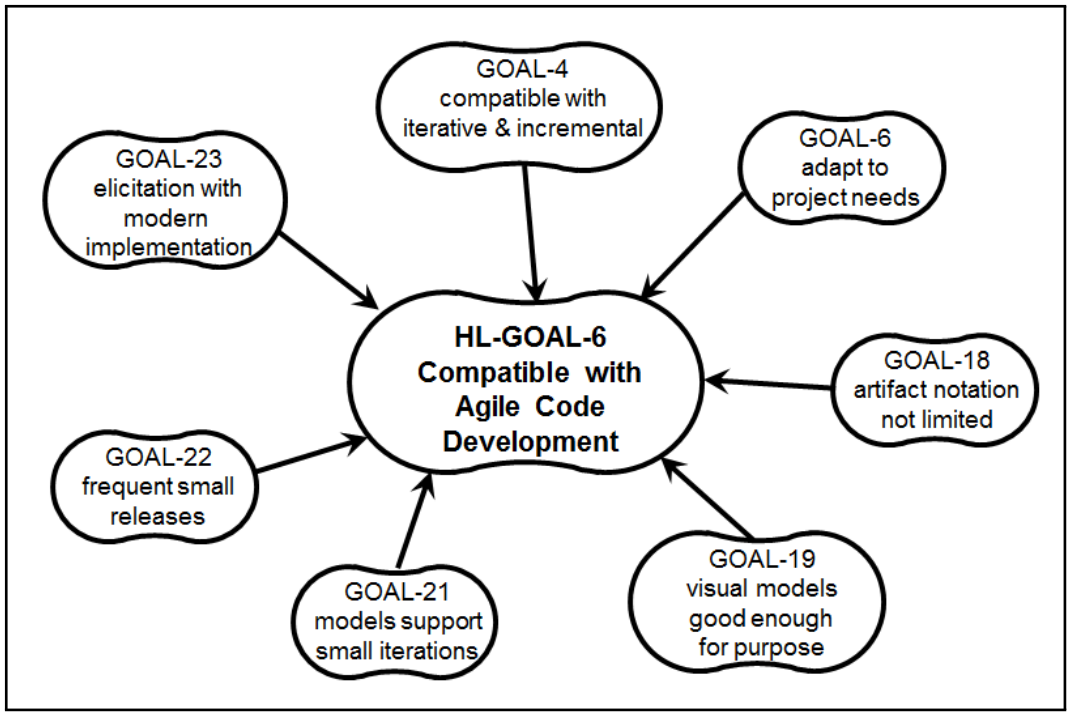


Figure 2.13: URN Goal Model for HL-GOAL-6.

Chapter 3

Definition of the Software Agile Modeling and Engineering Methodology

This chapter presents the definition of the Software Agile Modeling and Engineering Methodology (SAMEM). Methodology as defined by the Oxford English Dictionary: *a system of methods used in a particular area of study or activity.* According to Wikipedia, a methodology consists of the following concepts: paradigm, theoretical model, phases, and qualitative or quantitative techniques. A methodology is a guide to achieving a goal such as the development of a software solution without specifying a specific set of actions. The SAMEM is a methodology in the accordance with the above definitions.

The organization of this chapter for the description of the SAMEM is as follows:

- 3.1 Purpose of the SAMEM – defines the goals and objectives of the SAMEM methodology, including some motivation for the work.
- 3.2 Software Engineering First Principles – describes a set of first principles for reasoning about the software solution development process to assure high quality. A first principle is a truism that can act as a metric for evaluation of the solution artifacts or project process.
- 3.3 The SAMEM High-level Structure – covers the fundamental concepts providing the foundation for the SAMEM and relationships between the concepts.
- 3.4 The SAMEM Process – is detailed by following the organization of the RM-ODP viewpoints, which leads from the highest levels of abstraction of the solution to the lowest levels. Each viewpoint level is accompanied by examples from one of the case studies.
- 3.5 Non-functional Requirements Handling – describes the relationship of non-functional requirements to the SAMEM approach, which is focused on the functional requirements, the solution design, and the realization.

There are many examples in this chapter that come from the industry projects where an early version of the SAMEM was employed. The SAMEM was used for two different medical device companies, called Case Study 1 (CS-1) and Case Study 2 (CS-2). The case study companies develop, manufacture, and sell two different sets of products. Where

appropriate, the designator *CS-1* or *CS-2* will be used to label examples to indicate the industry source.

Paradigm as defined in the Oxford English Dictionary: *a typical example or pattern of something.* Another definition of paradigm is a distinct set of concepts, thought patterns, and standards. The recurring idea in the above definitions is the use of patterns. In the SAMEM, several important paradigms or patterns are used to provide guidance. The main pattern of guidance is provided by the Open Distributed Processing – Reference Model (RM-ODP) [ISO98] for systematically progressing from abstract definitions of the problem to a concrete solution realization. The use of RM-ODP is defined in Sub-chapter 3.4.1 and practical examples of its use are in Sub-chapter 4.3.5. A second pattern used throughout the SAMEM is of iterative & incremental or agile work cycles [Amb02]. The application of iterative & incremental techniques is mentioned throughout the thesis, but the primary description of the practical use is in Sub-chapter 4.3.1.

There are three parts to the theoretical model behind the SAMEM. The first part is a set of Software Engineering First Principles (SEFP). A SEFP is a rule of truth outside the methodology against which the aspects of the methodology can be tested to determine the goodness of the definitions. Examples of the SEFPs used are in Sub-chapter 3.2. The second and third parts of the theoretical model are respectfully meta-definitions for the process and artifacts in Chapter 5 and Chapter 6.

The phase aspect of the SAMEM adherence to the definition of a methodology appears in the project process phases. The project process phases are described in Sub-chapter 4.3. The phases provide a different perspective on the process of moving the project from the beginning to the end. In the SAMEM, the phases are coarser than the RM-ODP pattern in that several RM-ODP viewpoints fit into one project phase. The project phases assist in coordinating the iterative & incremental cycles and providing business decisions milestones for evaluating further investment in the project against likely benefits.

The bulk of this chapter describes the techniques used within the SAMEM. The organization of this chapter follows the normal use of the RM-ODP paradigm from the most abstract level of specification in the Enterprise Viewpoint to the most concrete level in the Technology Viewpoint. Each RM-ODP Viewpoint has its own set of techniques, sub-processes, and artifacts for expressing the requirements or design information. The techniques are described and multiple examples from the case studies are given to illustrate concrete usage of the techniques. The final major part, Sub-chapter 3.8, describes how the SAMEM handles important non-functional requirements.

3.1 Purpose of the SAMEM

The primary purpose of the SAMEM is to as effectively as possible achieve success in creating a software-based solution to a business problem. A key to achieving this purpose is in the following statement from Brooks [Bro95],

> "I will contend that conceptual integrity is the most important consideration in system design."

With a clear concept statement of the problem to be solved, albeit at a high level of abstraction, as the starting point and a disciplined refinement to more concrete expressions of the solution, adherence of the solution to the concept can be achieved and demonstrated. However, in practice, the concept can become misinterpreted and misunderstood by the team members due to the accumulation of accidental complexity during the requirements elicitation and design activities [Bro95], [Bro10], [Win96], [BPKR09], [FM15].

All elements of the SAMEM are intended to mitigate the risk of developing the wrong or sub-optimal solution and against an ineffective project process. A wrong solution does not fulfill the stakeholders' goals. The wrong evaluation is not black and white as the solution could fulfill some of the stakeholders' goals, but partially fulfill or miss others which results in a sub-optimal solution. Through the processes and communication techniques, the SAMEM should clearly express which goals will be completely realized and which goals might not be fully realized because of technical reasons or because of project budget limitations. The management of stakeholder expectations is of critical importance to the perceived success of the project and is reflected in the high-level thesis goal ***HL-GOAL-1***.

The SAMEM is constructed as a pragmatic balance among the components of an iterative & incremental process, light-weight artifacts (visual models) supporting effective communication, and tools for both project management and artifact creation. The pragmatism is based on 35 years of software solution development experience by the author. The author has experience with successful solution development projects and unsuccessful projects. The key motivating unsuccessful project was the first professional project the author was involved in, which was the development of a computer-based drafting product for mechanical engineers. Many instances of the product were sold, but almost every customer returned the product and demanded their money back. This experience committed the author to learning, developing, and employing software engineering techniques to mitigate against future such failures. The confirmation of the pragmatic success of the SAMEM is via the empirical evidence collected in the surveys of the participants in the case studies (see Sub-chapter 7.1).

The SAMEM uses certain first principles that provide a framework to evaluate the methodology design choices and allow generalization to other domains. The foundation of SEFPs helped to make the choices needed to fulfill the definition of a methodology as given above. Alternatives for the paradigm, processes, and techniques were evaluated with the principles with the best choices selected. A further benefit to using the SEFPs is in explaining to the project team members why the SAMEM does things in a certain manner. See Sub-chapter 3.2 for the SEFP definitions.

3.1.1 The Secondary SAMEM Purpose is to Enable Innovation

A secondary purpose of the SAMEM is not to create methodology barriers to innovation, but to enable innovation possibilities. The author's experience with other methodologies during his years of product development revealed a general lack of support and consideration for innovation. Much of this sub-chapter is from the author's paper "A Proposal

of Practices, Processes and Models that Enable Innovation Potential" [Mat17] with additions beyond the scope of that paper. Innovation is neither predictable nor guaranteed within a solution development project. Novel ideas can appear during any level of work, whether as insight during requirements gathering, in high-level design activities, or in developing code algorithms. As innovation is a creative action, the SAMEM proposes a pragmatic collaboration of related practices, processes, and modeling ideas that enable innovation to potentially happen or at least remove some blocking factors.

3.1.1.1 Innovation Practices Rationale

The practices, as embodied in the SEFPs, are mostly mental techniques for a designer to use which help to maintain an open mind to solution possibilities. The process approaches are based on and support progress within a flexible, iterative & incremental methodology that can respond to new ideas. The modeling technique ideas strive to support the clear communication of understanding and the evolution of the design through multiple alternatives.

There are many factors that contribute to establishing an environment where innovation flourishes. In [Pet96] the following statement about engineering is made: "And though engineering is the art of rearranging the materials and forces of nature, the immutable laws of nature are forever constraining the engineer as to how those rearrangements can or cannot be made." Software engineering differs because it is, at least outside of the hardware, independent of the physical laws of nature. However, both software and physical materials engineers have the same goal of creating a good solution for their fellow humans.

Are there corresponding "laws of nature" for software engineering design that enable good design and minimize poor design? There are multiple essays and examples in [Win96] about bringing design to software and in [Bro10] the idea is repeated that design is a messy creative process. Messy is a situation that the business manager of the project tries to avoid, rather predictable project execution is desired. A contradictory project situation is created from the two competing goals.

Practices are mental or thinking aids to maintaining the thought flexibility and discipline needed to allow for innovation. A practice is also called a Software Engineering First Principle (SEFP). The analogy is to physics first principles such as the idea of inertia or the absolute speed of light. The benefits of the SEFPs are to have an evaluation framework outside of the project. Through an application of an SEFP to the design alternative, the designer can perhaps judge if the alternative can be improved or a novel alternative is possible. A discussion of SEFPs or practices is covered in Sub-chapter 3.2.

3.1.1.2 Innovation Processes Rationale

The process used to control and manage the project can either enable or suppress innovation. What are the characteristics that could enable innovation during the project? An iterative & incremental or agile approach has several features that can enable innovation to occur. The primary feature is the short cycles of work, which allows for change

of direction. Not just during an iteration, but especially at the end there is an ideal opportunity to review the work using the SEFPs. In [Bro10] the cycle of evaluation of design alternatives and back to earlier decisions is clearly explained.

When the increments of work are kept small, then the design momentum is kept small. The main component to the momentum is the personal investment by the team members in their work. Design momentum interferes with innovation, because people get invested in a direction and resist admission that the earlier choices were poor [Coh10].

A good project process will provide guidance from an abstract solution design through a concrete implementation. The Open Distributed Processing – Reference Model, ISO-10746, (RM-ODP) [ISO98] is one example of a framework that provides abstraction guidance. The desired guidance feature is an approach that helps to order the questions to be addressed by importance and overall impact on the solution possibilities. The RM-ODP standard uses viewpoints to order the abstraction level and therefore the question importance. The path through the viewpoints, from the Enterprise Viewpoint (highest abstraction level) to the Technology Viewpoint (lowest abstraction level), is not a straight line but can be done in an iterative & incremental manner. More detail about the SAMEM processes are given in Sub-chapters 3.4, 4.3, and 4.4.1.

3.1.1.3 Innovation Modeling Rationale

The modeling of requirements and designs is standard engineering practice. Mechanical engineers, civil engineers, and building architects use graphical models or drawings as the primary documentation artifact for their solutions. At times, the engineers will create physical mock-up models to communicate the ideas. The main reason for this is the communication density and clarity over text [LS87], [Moo09]. The stories of sketching the new idea on a napkin during a lunch with colleagues abound.

Creating graphical models of the requirements and the design alternatives to those requirements is an application of change language SEFP. The language change is from text to graphics. The graphical expression allows for different perspectives to be generated. The graphical models are often faster to generate, especially in a sketching mode. Fast exploration of new ideas aids innovation. It is rare that an innovative idea is fully formed on its first expression. Rather, the innovation happens in multiple refinement steps as the idea is evaluated against the solution goals, discussed with colleagues, and compared to alternatives. Equally important is keeping a record of the failures or weaknesses so that the effort is not duplicated.

3.1.2 Supporting the Engineering Due Diligence Purpose

Sub-chapter 3.1.1.2 describes how the RM-ODP gives guidance in moving through the project abstraction hierarchy levels from the initial vague ideas to the concrete solution. The additional engineering benefit of **Due Diligence** is achieved by employing the RM-ODP viewpoints. Due diligence is the professional behavior of investigating and honestly evaluating alternatives to achieve a successful solution. In the academic world, one appearance of due diligence is in the extent and quality of the references.

Due diligence in the goals and requirement elicitation work takes many forms. One form is the interviewing of an appropriate number and variety of stakeholders. The awareness of legal limitations and requirements impacting the solution is also part of due diligence. In making a decision to move forward with a project, the knowledge and understanding of major competitive and complementary products is essential.

In making the design decisions of the RM-ODP Engineering Viewpoint, due diligence is needed to make effective build or buy business choices. Creating a solution by writing code from scratch is not always the best business decision. Some of the other choices that must be evaluated with due diligence are buying an existing product, buying a product that can be adapted within its range of options or preferences, looking for an open source product, using a product or products that fulfill part of the solution and creating the missing capabilities, or creating a new solution from scratch. The due diligence work involves looking for alternatives and evaluating them with respect to fulfilling the goals and requirements, the costs versus the benefits, and impacts on the organization.

Due diligence within the RM-ODP Technology Viewpoint is making the best possible technology decisions, within current limitations and constraints. Some of the decision areas are programming languages, hardware infrastructure, user interaction devices, networking protocols, and development tools. In making the technology decisions with due diligence, the factors like initial cost, people education, maintenance costs, speed, reliability, and productivity are evaluated against the alternatives.

Each RM-ODP Viewpoint has an associated set of decisions. Some decisions have large impacts on the solution and some have small impacts. Engineering professionalism demands that the decisions with the large impacts be made with well executed due diligence. The earlier a decision is made in the project or in other words at a higher level RM-ODP Viewpoint or level of abstraction, the more likely it will have a larger impact. The choice between Python or Java as a programming language is an example of a large impact decision. An example of a small impact decision is the name of a variable inside a Java method or using a particular coding standard.

3.2 Software Engineering First Principles

An important goal of the SAMEM is its ability to adapt to various domains. The SAMEM definition needs to be flexible so that it can evolve, but within a set of guiding principles that assure a quality solution is created through a reliable process. In addition, the SAMEM should not block insight into the essence of the problem; rather, it should facilitate innovation in the design of the solution [Bro95], [Bro10], [Win96]. The SAMEM is not a blind cookie-cutter approach [Amb02], [Pet96]. The Software Engineering First Principle (SEFP) idea provides a thought mechanism that allows for flexibility while providing a touchstone to maintain rigor. The SEFPs are designed to manage the balance between rigor for quality and completeness, and the flexibility to adapt to unknown situations. The SEFPs will provide some of the rationale for the SAMEM definition choices. The intent of the SEFPs is similar to the principles of the Agile Alliance [Bec00], [Amb02].

The SEFPs described here are the practices or principles which either existed beforehand and were adopted or were formulated from lessons learned during the SAMEM development. Many of the SAMEM SEFPs relate to the SEMAT theory area and are a possibility for future work (examined in more detail in Sub-chapter 8.2).

3.2.1 Solution Conceptual Integrity (CI) SEFP

The Solution **Conceptual Integrity** (**CI**) SEFP is adapted from ideas in the book *The Mythical Man-Month* by Frederick Brooks [Bro95] and describes the difficulties in building large complex solutions. There are two points he makes that are critical to having a successful solution:

> "The hardest single part of building a software system is deciding precisely what to build."

> "I will contend that conceptual integrity is the most important consideration in system design."

The *solution concept* encodes the essence of what should be built and *conceptual integrity* means to be true to the solution concept. The encoding could be visual or text or a combination of both, but has the property that all the stakeholders understand the value that will be created by the solution and agree to the concept. The concept encoding should be compact and easily represented in order to meet ***HL-GOAL-1***, ***HL-GOAL-2***, and ***HL-GOAL-3***.

Communicating and recording "what to build" and the solution concept are the primary purposes of the requirements specification. This SEFP states that it is critical to define and establish a concise and clear solution concept. The solution concept is the highest-level summary of the *What To Do* and the *Why To Do It*. The solution concept is the touchstone of correctness used during the Development Process to ensure the solution stays on the correct track. It does not cover project management objectives or other business objectives targets. To support the modeling artifact approach, the solution concept is a statement of the goals to be achieved in a combination of both graphical images and text where necessary.

The conceptual integrity of the solution is set in the business goals and requirements. An important part of the requirements that is often forgotten as people hurry to list functional requirements is the *Goal* of the solution. The *Goal* of the solution is the most basic statement of what value the solution is trying to satisfy; therefore, it becomes the touchstone for maintaining the conceptual integrity, but the expansion to more detail must be consistent with the *Goal*. The *Goal* can be a single statement or a small set of *Goal* statements, but too many *Goal* statements lead to confusion of purpose. In many cases, the Goal is expressed in terms of the business value the solution is intended to deliver [Amb02], [BPKR09], [FM15]. For simplicity, this thesis will use *Goal* in a singular form.

This is also echoed in the Volere Requirements Management Methodology [RR99] in the first category "The Purpose of the Product." The SAMEM emphasizes separating

the world of the solution from the NOT the solution. The *Goal* needs to be expanded to more accurately and precisely define the solution.

Establishing the concept or *Goal* for the solution is the first step in the SAMEM and as the first step it is the fundamental SEFP. It provides a metric against which the work can be validated. After each iteration, at a minimum, but also during the iteration in the execution of the SAMEM the following questions should be asked to detect any drifting away from the *Goal*:

- Does this requirement adhere to the *Goal*?
- How does this requirement help fulfill the *Goal*?
- As the nature of the business problem and the solution direction are better understood, should the *Goal* definition be updated?
- Does this solution design proposal adhere to the intent of the *Goal* or does it go beyond?

3.2.1.1 Visual Model Example of the Solution Concept or Goal

The *Solution Concept* or *Goal* can be expressed either in text or through an image. Expressing it both ways has advantages, as it helps with clear communication to the development team and for verification with the stakeholders. In Figure 3.1, the solution concept from a case study is expressed in text combined with a picture, and the business value concept is included. Each project will have a different solution concept, although within a business domain there will be similarities.

The specific solution represented in Figure 3.1 (from Case Study company 2) is designed to support the electronic management of information created and updated during the design of a new medical device. At the start of the project, the bulk of the product definition and design information was managed by hand on paper. The management of the information created during the product design is mandated by the US Food and Drug Administration (FDA) through regulation 21 CFR §820.30 (Quality System Regulation – Design Controls).

There are three major components to the solution concept expressed in Figure 3.1. First (label ①), the title phrase "Product Definition enables Product Leadership" at the top of Figure 3.1 expresses the solution *Goal* of producing the leading product in their business area. Ideally, the title summarizes the business value according to the Value-Discipline Model [TW95]. Included in the business *Goal* is the belief from the stakeholders and expressed in their words that managing the product definition is key to creating leading products. Second (label ②), in the box at the bottom of the figure the phrase "File it, Find it, Reuse it AND Don't REDO it!" expresses the business value objective. The business value is created by not redoing work, i.e. do it *right* the first time and reuse it as often as possible. Third (label ③), includes the arc with the text "Product Definition Scope" and the jigsaw puzzle image in the middle of Figure 3.1 shows at a high level of abstraction that a product definition consists of many related

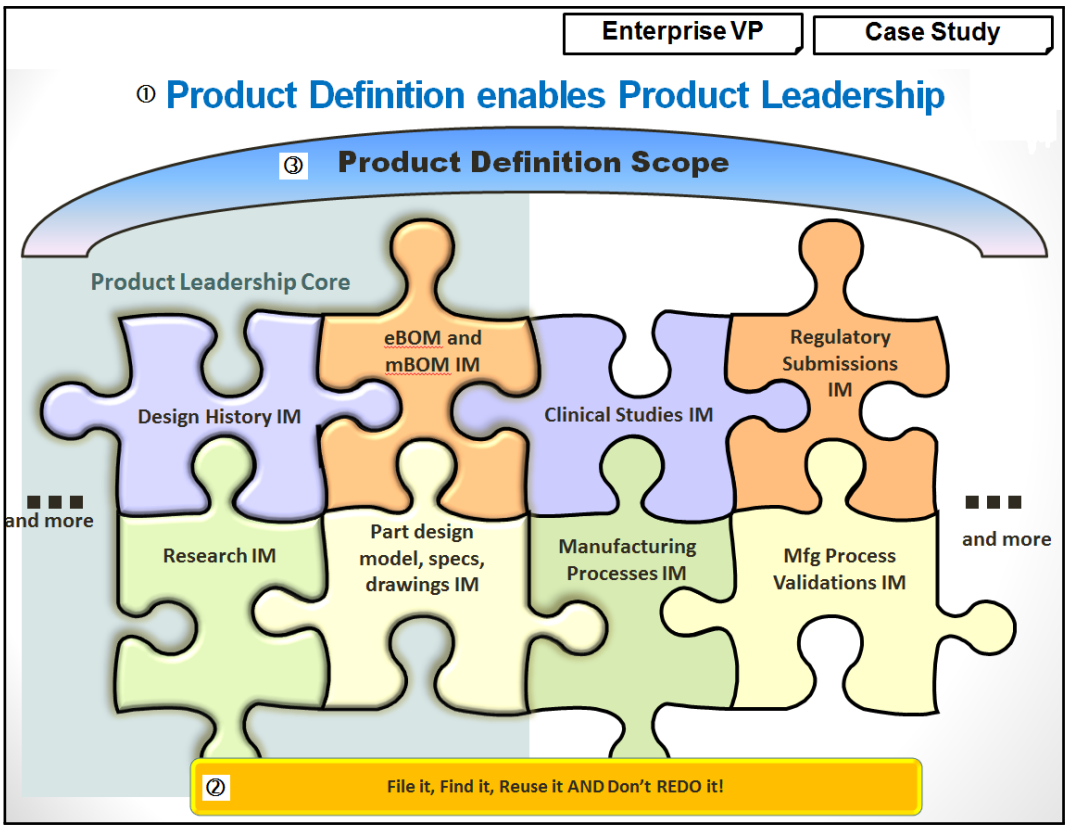


Figure 3.1: Solution Concept Example.

Information Model (IM) pieces. The jigsaw pieces in the central visual image form only a part of the total product definition. The limited number of jigsaw pieces was selected with stakeholder feedback as being both representative and within the project budget.

While there are other ways to represent the idea of solution components that are different, but equivalent, the jigsaw image or metaphor arose in an early meeting with the stakeholders from CS-2. Because the jigsaw metaphor resonated with all meeting participants, it was accepted as the core image. Images such as the jigsaw puzzle cannot be strictly planned for, but normally arise organically from the initial interactions with the stakeholders. A visual model such as Figure 3.1 fulfills the thesis goals of a *compact format* (***HL-GOAL-3***) and *better communication* (***HL-GOAL-1***).

3.2.2 Essential and Accidental Complexity (Essential) SEFP

The "No Silver Bullets" article by Frederick Brooks [Bro95] introduces the concepts of the essential complexity and accidental complexity of a solution.

> "Second, to see what rate of progress we can expect in software technology, let us examine its difficulties. Following Aristotle, I divide them into essence

> – the difficulties inherent in the nature of software – and accidents – those difficulties that today attend its production but are not inherent."

The purpose of the adopted **Essential** SEFP is to instill in the design of the SAMEM and the practice of the SAMEM the habit of asking the following questions often:

- Does this work contribute to the realization of the solution?
  - A **No** answer indicates possible accidental complexity or drift from the solution concept.
- Is this work done in the most effective manner possible or is there a simpler manner?
  - A **No** answer raises possible accidental complexity factors.
- Is the additional overhead of using the tool less than the amount of work without the tool?
  - A **Yes** answer indicates a minimum of accidental complexity issues.
- What forms of artifacts minimize the effort to create and maintain while maximizing the communication effectiveness?
  - Not a **Yes** or **No** answer, but a regular question used to drive continuous process improvement. Best asked at the end of an iteration.
- Are things done out of habit that can be eliminated?
  - A **Yes** answer raises the possibility for elimination of accidental complexity and process improvement.
- Are things accomplished through the use of outdated technology that can be improved?
  - A **Yes** answer raises the possibility for process improvement.

The problems that need to be tackled are often very complex. The complexity comes from many sources, such as a problem previously not tackled, a very large problem, a geographically distributed team, the use of new technologies, or poor project processes. Some of the complexity issues cannot be reduced or eliminated, but some can.

In the context of the SAMEM, the focus is on minimizing the accidental complexity to generate a requirements specification. The clearer the requirements are stated, the cleaner the solution design and implementation. The specification of the business requirements of the solution should be a very pure form of the essential complexity of the business problem; however, the form itself introduces accidental complexity. Establishing the solution concept is part of gaining an understanding of the essential complexity of the problem. There are other tools, mental and computer-based, in the SAMEM that help with establishing the essence of the problem and minimizing accidental complexity in its expression.

This principle helps limit the false and non-essential requirements from the solution specification. The application of this principle can improve the SAMEM and help when it is adapted to other domains. The Agile Alliance principle of *simplicity* is very similar.

3.2.3 Stability to Variability (S2V) SEFP

The SEFP of working from **Stability to Variability** (**S2V**) helps to identify starting points for the requirements gathering. *Stability* is the identification of the one, two, or small number of aspects about the problem domain that are least likely to change with respect to a time period much longer than the scope of the project. If a too large number of aspects are seen as stable, then constraints on innovation can be artificially created. The identification should be at a higher level of abstraction. For example, in Figure 3.1, the most stable part of the problem is the information about the product definition. Of course, there will be variations in the information for different products, but product information emerged as the most stable through the requirements elicitation process. With the identification of a stable aspect, it must be understood why that aspect is more stable than other aspects of the problem domain.

The opposite of stability is *variability*. The more something is likely to change with respect to the time period of the project, the more variable it is. The understanding of why and relative placement of the problem aspects on a scale of stable to variable provides an indicator of the likely optimal order in which to proceed with the project. The activities of identifying the most stable and most variable aspects of a problem domain are an important step in understanding the problem domain.

Product lines or variant products are an example of explicitly applying this SEFP. The core of the product in a product line set of products is relatively *stable*, while the differences are the *variability*. For example, the stability in a mobile phone line of products is the capability to make calls, while the variability can be features such as the amount of memory or the camera resolution.

Progress is shown and confidence in the project is developed when the stable aspects are addressed first. It is easier to elicit requirements, do design, and develop a prototype when there is little chance of change. There is a side benefit when the problem is attacked from stable to variable aspects of creating a perception of reliable and reasonable project progress.

The solution concept and project Goal will often give a good indication of the more stable business aspects. A good solution concept will reflect a very stable part of the business problem, as well as opportunities for improvement. One approach that helps in identifying a stable aspect is widespread agreement among the stakeholders on that aspect. The reasons why the information models are stable in Figure 3.1 is that there are two forces that create the stability: 1) the products themselves using the fact that most new products are evolutionary, not revolutionary and 2) the FDA or ISO regulations that specify the information that must be delivered for product approval. Identifying why an aspect is stable or variable is the task of identifying the business forces that promote stability for the smooth running of the business or the forces that disrupt for business responsiveness or innovation.

In other domains other aspects will be more stable. For example, for a company that produces safety and security solutions for the oil and gas industry, the business process is the point of greatest stability. The business process consists of four high-level steps: 1) monitor, 2) detect incidents, 3) assess risk, and 4) suggest response. The more variable

parts are the monitor data that flow through the process and patterns of failure to be identified.

3.2.4 Symmetry of Action (SoA) SEFP

The **Symmetry of Action** (**SoA**) SEFP is especially useful in both requirements elicitation and during design work. Symmetry of Action arose from applying the physics principle of "for every action there is an equal and opposite reaction." In a solution, there are often pairs of actions: *do* and *undo*. For example, if a stakeholder talks about *creating* information during requirements elicitation, then the symmetric action is *deleting* the information. At the design level, if there is a create function, then a delete function should be considered. Of course, there are domains where information or data is not or cannot be deleted, but is transformed from a *can be used* status to a *do not use* status. The symmetric action to *create* becomes *obsolete* or *superseded*.

The value of the Symmetry of Action SEFP is in achieving ***GOAL-10*** which is ensuring completeness. Applying this SEFP is a means to check on the completeness of the requirements specification that can be used by the developers and explained to the stakeholder without explicit questioning. The **SoA** check is used in design review meetings to evaluate the functional completeness or user interface actions. Iteration & Increment planning can use **SoA** to schedule pairs of implementation work and testing so a more complete prototype is presented to the stakeholder.

3.2.5 Modules (Modules) SEFP

David Parnas [Par72] makes the point that a decomposition of the solution based on information or decision hiding often provides the better level of development management, product flexibility, and comprehensibility. The decompositions are called Modules and are adopted as the **Modules** SEFP. This is one of the earliest statements of abstraction and the power of abstraction. Models can be an abstraction of the solution requirements and the design of the solution.

One of the goals he values is that modules should be capable of being assembled, reassembled, and replaced within the completed system. His technique for this is expressed in the following quote,

> "...module is considered to be a responsibility assignment."

From a requirements perspective, this translates into specifying "what" should happen or is needed from a business view, while delaying the specification of the "how" it is accomplished until the design phase. The word object can be substituted for module in most places.

The idea of modules can be applied to organizing the requirements in terms of the RM-ODP viewpoints. In the gathering of the data requirements of the solution, specific implementation possibilities are avoided. For example, in specifying the requirement for unique identifiers for each data item, a particular format or size definition is postponed

until the engineering design work is started. When modules are used in the sense Parnas has defined, they can help with design work and implementation planning through separating the engineering design decision for a relational database and the technology choice of Oracle™. If the definition of the module is clear enough (***Goal-19***), the work of either further detailed definition (design refinement) or the implementation can be given to another person with minimal communication overhead.

3.2.6 Coupling and Cohesion (C&C) SEFP

When attempting to satisfy the goals of defining modules, the design of how things are connected and the consequences of the connections are important [Mye75]. The term cohesion expresses the singularity of purpose of the module. A module with the best cohesion stands on its own and does one thing, which fulfills the definition from Parnas. Coupling is used to indicate the closeness of interaction between two modules with no interaction the best. The leveraged ideas of **Coupling & Cohesion** form the **C&C** SEFP.

Coupling and cohesion are metrics by which the quality of modules can be measured. A good module will have high cohesion meaning that it stands on its own and clearly does one thing. A good module will have low coupling meaning that it refers to a relationship in which one module interacts with another module through a stable interface and does not need to be concerned with the other module's internal implementation.

The types for coupling from best (low) to worst (high) are:

- No Coupling – the modules do not communicate with one another.
- Message – the modules are not dependent on each other, instead they use a public interface to exchange parameter-less messages or events.
- Data – the modules share data through, for example, parameters. Each datum is an elementary piece, and these are the only data which are shared (e.g. passing an integer to a function which computes a square root).
- Stamp – the modules share a structure and only use part of it, possibly a different part (e.g. passing a whole record to a function which only needs one field of it).
- Control – one module controlling the logic of another, by passing it information on what to do (e.g. passing a what-to-do flag).
- External – occurs when two modules share an externally imposed data format, communication protocol, or device interface.
- Common – when two modules share the same global data (e.g. a global variable). Changing the shared resource implies changing all the modules using it.
- Content – one module modifies or relies on the internal working of another module (e.g. accessing local data of another module).

The types for cohesion from best (high) to worst (low) are:

- Functional – all parts of the module or class contribute to a single well-defined task.
- Sequential – the parts of the module are grouped because the output from one part is the input to another part (e.g. a function which reads data from a file and processes the data).
- Communicational – parts of the module are grouped because they operate on the same data (e.g. a module which operates on the same record of information).
- Procedural – parts are grouped because they always follow a certain sequence of execution (e.g. a function which checks file permissions and then opens the file).
- Temporal – the parts are grouped because they are all processed at a particular time of program execution (e.g. a function which is called after catching an exception which closes open files, creates an error log, and notifies the user).
- Logical – the parts are categorized as doing the same thing, even if they are different by nature (e.g. grouping all I/O handling routines).
- Coincidental – the parts are grouped arbitrarily and have no significant relationship (e.g. a module of frequently used mathematical functions).

While functional cohesion is considered the most desirable type of cohesion for a software module, it may not be achievable. There are cases where communicational cohesion is the highest level of cohesion that can be attained under the design or programming language circumstances.

During the requirements gathering process, the information pieces and business process actions are discovered in a disjoint manner and must be assembled into a coherent specification. The concepts of cohesion and coupling guide that assembly process to a quality and coherent specification. This also applies to the design of the solution from the requirements. Different combinations of assembling the modules correspond to different design alternatives and coupling and cohesion metrics are one dimension that can be used to evaluate the alternatives.

3.2.7 Patterns (Patterns) SEFP

Patterns of structure and action are hallmarks of a professional approach to doing work. The use of patterns, both architectural and transformational, ensures a completeness and correctness to the establishment of the requirements, design, and ultimately the solution code. The **Patterns** SEFP is intended to keep the use and evaluation of good engineering patterns at the forefront of the solution work.

Patterns can be used to assist in achieving the necessary completeness while deferring details and as such they can be used to maintain the appropriate level of abstraction.

Patterns work well with module design. The lesson for innovation from [AIS+77] is that there are levels of patterns or patterns within patterns that can be used in designing a solution. For software, there are collections of design patterns available [Fow03], [GHJV95]. In addition, there are the patterns conveyed by the acronyms, such as CRUD (Create, Read, Update, Delete) and ACID (Atomicity, Consistency, Isolation, Durability) which can be used to check the completeness of requirements and evaluate designs.

While there are lists of well-known patterns, often an aspect of completeness and innovation is the discovery of new or domain-specific patterns. Design innovation can be stimulated by the following questions:

- Can an existing pattern simplify or improve the design?
- Is there a new pattern emerging from the design?

Patterns can be over-used which results in accidental complexity [Bro95]. Just because a pattern could be applied does not mean it should be. The use of a pattern in the design should be combined with a rationale that explains the benefit of the pattern and the context of its use. The rationale is especially useful for future maintenance activities, which are often done by a person other than the original author and for the training of less experienced colleagues.

3.2.8 Optimal Performance (OP) SEFP

The SEFP for **Optimal Performance** (**OP**) is formulated as *the fewest operations on the fewest pieces of data.* This approach to thinking applies from the gathering of requirements to the design of code methods. It helps in discovering the essence. Applying **Optimal Performance** SEFP to requirements gathering, the need for collecting traditional performance requirements is eliminated. In evaluating requirements, the **OP** SEFP is used to check whether the requirement is necessary to achieve the solution goal and is consistent with the solution concept.

In practice, some of the optimal performance SEFP questions are:

- Is this feature or this information essential to achieving the solution concept or not?
- Is the design of the feature or data as efficient as possible?
- Does the collection of features work in an optimal manner or are transformations/translations involved?
- Has the design been translated into good code?
- Are the best coding practices being used in relation to the life of the solution?

3.2.9 Change Language (CL) SEFP

An idea that evolved from the author's experience that helps to discover the essence of a problem or express the solution concept is stated as *change your words to change your perception to open innovation opportunities*. This is the **Change Language** (**CL**) SEFP. Almost every domain has its own set of vocabulary. In some cases, the domain vocabulary must be used to ensure clear communication. However, when a solution developer uses alternative expressions and synonyms, opportunities are opened to think in new directions. This is the mental equivalent of picking an object up and turning it around in your hands to view all sides.

A co-approach to new words is the communication richness provided by the graphical expression of the design idea or concept [LS87]. It is possible to articulate the solution concept graphically [Mat11]. User Requirements Notation (URN) [URN12] and *i** [Yu09] can be used to express the essence of the business value and the goals of the business organization as an alternative to text. Many aspects of a software design alternative can be conveyed graphically in Unified Modeling Language (UML) [OMG15b] or a modified sub-set of UML.

The **CL** SEFP also stimulates innovation. Through deliberate attempts to express the solution concept, the requirement, the architecture design, or the technology design via different words or images, alternatives are revealed.

An example of the **CL** SEFP is visible in Figure 3.1. The jigsaw puzzle pieces used to communicate that the solution can be seen as a set of interconnected semi-independent information units is a simple visual statement. Another example of the **CL** SEFP is that the solution in one case study was called the R&D Knowledge Management System (R&D KMS), rather than using the name of the off-the-shelf product it was implemented on. The vision of managing the R&D knowledge was more powerful and less limiting than referring to the Siemens Teamcenter Product Lifecycle Management product, which was the technology of implementation.

3.2.10 Ready-to-Hand (R2H) SEFP

The term **Ready-to-Hand** comes from Martin Heidegger in his book *Being and Time* which refers to the natural usability of a tool such as a hammer. When there is some kind of breaking down in using a tool, an *unreadiness-to-hand* occurs. The **R2H** SEFP idea acts to evaluate whether the requirements reflect the concept, the design reflects the requirements, and the implementation reflects the design. When there is a breaking down, then the solution fails to be **Ready-to-Hand**, this is revealed through the unnatural use of the solution.

A deeper explanation of Heidegger's ideas to the design of computers and software is found in [WF86]. For this thesis, the essence of **Ready-to-Hand** (**R2H**) is that it offers an external criterion for the evaluation of the design of a solution. In this case, design is used in a very general sense referring to everything from the statement of the solution concept through the realization. A breakdown is often caused by the presence of accidental complexity or an unclear expression of the solution essence at some point in

the design with its propagation throughout the solution. For example, the inclusion of a "cool" feature such as pop-up windows that obscure valuable information, but one that does not help achieve the solution concept. At any point along the process, a breakdown can occur. The **R2H** SEFP can generate questions to discover something about the nature of the breakdown and possible remedies, such as:

- What is unnatural about the current solution design?
- Upon gaining a deeper understanding of the business problem and possible solution alternatives, is the solution concept still valid?
- Are there requirements that do not support the solution concept?
- Is there a lack of fidelity to the solution concept?
- Are there missing features?
- Does the process and task navigation match the natural business thinking?
- Is the user interface too complex and off-putting?

3.2.11 Form Follows Function (FFF) SEFP

The American architect Louis Sullivan coined the phrase *form ever follows function* [Wikipedia]. The phrase is commonly shortened to **Form Follows Function** (**FFF**). The application of the **FFF** SEFP is as an external check on the artifacts produced by the SAMEM. As an external check, the features, functions, and capabilities proposed for the solution in the requirements can be evaluated against achieving the solution concept. The **FFF** SEFP can be applied to the engineering design to minimize over-engineering often caused by addressing marginal error situations that are very unlikely to arise. In the technology design and implementation, the **FFF** SEFP is used to evaluate which code patterns are most useful. Distinguishing between essential and accidental complexity is helped through the application of the **FFF** SEFP.

The **FFF** SEFP is used in the design of the SAMEM. The functions of the SAMEM are expressed in the high-level goals (***HL-GOAL***) as listed in Sub-chapter 1.2. The *function* of the communication goals ***HL-GOAL-1***, ***HL-GOAL-2***, and ***HL-GOAL-3*** are achieved in large part through the *form* of visual modeling techniques. The *form* of the iterative & incremental process approach helps achieve the *function* stated in ***HL-GOAL-4***.

3.2.12 Summary of Software Engineering First Principles

The Software Engineering First Principles are independent of any particular methodology, engineering approach, or technology. They are also independent of solution domain. This independence places them outside the problem and solution worlds. Since the SEFPs are external to the work, they can act as measures of the quality, in the broadest sense, of the solution and all its components. Below is a list of the SEFPs:

- Conceptual Integrity (CI)
- Essential versus Accidental Complexity Awareness (Essential)
- Stability to Variability (S2V)
- Symmetry of Action (SoA)
- Modules as defined by Parnas (Modules)
- Coupling and Cohesion Metrics (C&C)
- Patterns (Patterns)
- Optimal Performance (OP)
- Change Language (CL)
- Ready-to-Hand (R2H)
- Form Follows Function (FFF)

The SAMEM is designed to work in conjunction with the SEFPs to meet its design goals and to produce the highest quality solutions with quality being fitness for use as judged by the stakeholders.

3.3 The SAMEM High-level Structure

The SAMEM is introduced by a simple view of its architecture. Five major components make up the essence of the SAMEM: the Processes, the Model Artifacts, the Tools, the Software Engineering Principles, and the Methodology Framework (see Figure 3.2, same as Figure 1.1 but repeated for convenience). For both the development of the SAMEM and the education of the development team and the solution stakeholders, it is useful to start from a simple model of the SAMEM and build up the definition in logical steps with the rationale for each step.

Iterative & incremental is the **Process** style in Figure 3.2. *Incremental* is defined as progressing towards a solution in small, controlled, and well-defined steps. An increment could be eliciting requirements from a specific stakeholder, designing a user interface screen, writing a set of tests, or implementing a Java class. An *iteration* is one cycle of a series where each cycle consists of the same general steps of work and produces a result. An example of an iteration is: talk with a stakeholder to elicit some requirements, model the requirements, verify that the requirements are correct, repeat until no more requirements are found.

An iteration includes a **verify** step so that each block of work done in an iteration is correct as far as can be known at that point in the project process before proceeding with the next block of work. The iterative & incremental process paradigm also applies to the development of the solution engineering design and the realization of the design.

The primary rationale for choosing the iterative & incremental process approach with a verification step is to control project risk. Another rationale for choosing an iterative & incremental process style is in the proven success at the programming level as seen in eXtreme Programming [Bec00], [Amb02], SCRUM techniques [Coh10], and general good engineering practices [Amb02], [Bro10], [FM15], [Pet96], [Win96].

The SEFPs are an essential part of the verification action at the end of an iteration. Especially early in the project as understanding is developing, the SEFPs of **CI**, **Essential**, **S2V**, **CL**, and **FFF** are useful in focusing the work. The focus is achieved by project team agreement on the artifacts and agreement on the boundaries between what is in the project and what is outside the project scope.

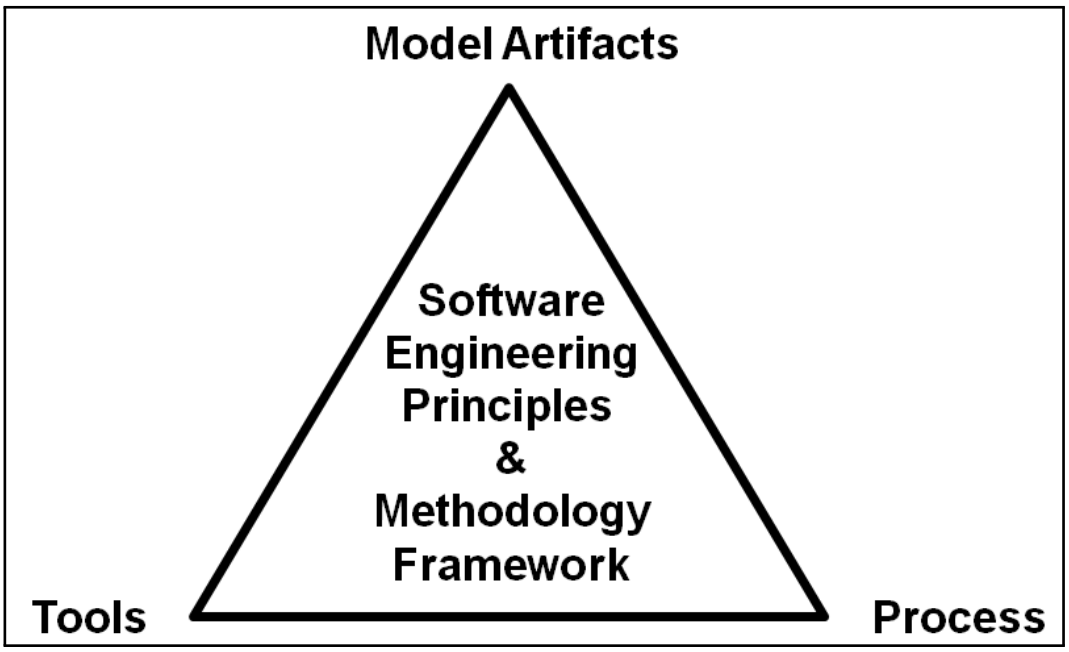


Figure 3.2: SAMEM Components.

The iterative & incremental process paradigm strongly contributes to meeting the thesis goal of ***HL-GOAL-1***, *better stakeholder communication* by providing small amounts of bounded information for review on a frequent basis. The improvement in stakeholder communication carries over to the goal of ***HL-GOAL-2***, *development team communication improvement* for the same reasons. When the increments are of a reasonable size, reasonable varies from project to project and with the experience of the people involved, then achieving the goal of ***HL-GOAL-4***, *faster project processes*, is possible. The main speedup is due to the elimination of long wait times while either the stakeholder reviews a large body of work or the development team produces something for review. Iterative & incremental supports the thesis high-level goal of ***HL-GOAL-6***, *compatibility with agile methods for development.* While there are natural iterations due to people finding a common understanding [Bro10], [Pet96], [Win96], a consequence of small steps is the

risk reduction of major rework because something incorrect relative to the stakeholder desires is built.

The choice of an iterative & incremental process paradigm contributes to the fulfillment of the following goals: ***GOAL-2***, ***GOAL-3***, ***GOAL-4***, ***GOAL-7***, ***GOAL-14***, and ***GOAL-22***. The reduction in project risk of ***GOAL-2*** comes about through the small iteration steps which enable frequent checks on the correctness of the work and facilitates meeting ***GOAL-7*** for project management overview and control purposes. The ***GOAL-4*** is a design goal based on the successes of SCRUM and Agile techniques. The iterative & incremental process paradigm supports the related goals, ***GOAL-14*** and ***GOAL-22***, which look for multiple feedback possibilities and frequent small releases. ***GOAL-3*** is indirectly fulfilled through the capability to create traceability information on the artifact increments from iteration to iteration.

The requirements specification consists of many **Model Artifacts**, as shown in Figure 3.2, both visual and text. The ideas of Model-driven Engineering (MDE) [OMG14a], [Mat02], [Fra03], [FR07], [KWB03], [Whi15] and the power of graphical communication [LS87], [Moo09] are the rationale for specifying as many requirement artifacts as possible as visual models. The UML [OMG15b] is chosen as the modelling language starting point because of its widespread use as a design modeling language. However, the UML is not intended to be the only visual modeling language. The SAMEM supports the freedom to invent new and more appropriate models as needed [Amb02], [CGHM13], [BC12], [Mat09].

The high-level thesis goals strongly related to the **Modeling Artifacts** are ***HL-GOAL-1***, ***HL-GOAL-2***, and ***HL-GOAL-3***. The value of ***HL-GOAL-1*** and ***HL-GOAL-2*** are directly related to the power of visual communication [LS87]. The overall quality of the requirements specification, ***HL-GOAL-3***, is related to how compact it is, which means that it is more readily reviewed and is a result of the higher communication band width of visual models [LS87], [Moo09].

The visual **Modeling Artifacts** are connected to the following goals: ***GOAL-1***, ***GOAL-5***, ***GOAL-9***, ***GOAL-12***, ***GOAL-15***, ***GOAL-18***, ***GOAL-19***, ***GOAL-21***, ***GOAL-24***, and ***GOAL-25***. The accuracy of visual models for certain specifications is well known [LS87] which supports achieving ***GOAL-1***. The goals expressed in ***GOAL-5***, ***GOAL-9***, ***GOAL-12***, and ***GOAL-24*** all focus on communication effectiveness, which visual models excel at for many specification needs. ***GOAL-15***, ***GOAL-18***, ***GOAL-19***, and ***GOAL-25*** emphasize different aspects of communication flexibility and there are visual modeling techniques to choose from, including text when it is a better communication medium. In general, it is faster to create a visual model and get stakeholder feedback, which supports the faster iterations of ***GOAL-21***.

The **Tool** category in Figure 3.2 is broad. It covers computer-based tools used to create and manage the model artifacts, mental tools to guide the work during an iteration, and project management tools. The SAMEM does not depend on any particular set of computer-based tools. Mental tools, practices, and SEFPs used for investigation, design, and analysis are also an important part of the SAMEM. Of particular importance are the mental tools of patterns. Patterns happen at all levels of abstraction and help to

bridge abstraction levels by hiding details as Parnas proposes [Par72]. The SAMEM does not employ any specific project management tools, but a general project process supporting the SAMEM and used in the case studies will be discussed in Chapter 4.

The **Tool** category is most closely related to the high-level thesis goals of ***HL-GOAL-3***, ***HL-GOAL-4***, and ***HL-GOAL-5***. A good set of tools can speed up the project process as expressed in ***HL-GOAL-4***. Through the use of good tools, the high-level goal of ***HL-GOAL-5*** can be achieved as each development team member is more effective in their work. Good tools help in creating, reviewing, and communicating the more compact form of the visual models, which help in achieving ***HL-GOAL-3***.

The specific goals that have a high affinity to the **Tool** category are ***GOAL-10***, ***GOAL-11***, ***GOAL-13***, ***GOAL-16***, ***GOAL-17***, ***GOAL-20***, and ***GOAL-23***. Ensuring completeness as expressed in ***GOAL-10*** can be assisted by using tools to check the artifacts. Meeting ***GOAL-11*** and ***GOAL-23*** keeps the SAMEM up-to-date with the latest tool and software engineering improvements. ***GOAL-13*** is a design goal that provides the basis for creating and extending tools as the SAMEM evolves over time. The use of tools can help achieve the ***GOAL-16*** objective of making reasonable progress by limiting slower manual efforts. ***GOAL-17*** is a requirement on the tools to have the capabilities to track evolutionary improvements in the artifacts. Meeting ***GOAL-20*** requires that the tools have a range of adaptations or configurations to support a spectrum of domains.

The **Software Engineering Principles & Methodology Framework** in the center of the triangle in Figure 3.2 provides the coordination and integration between the **Process**, **Tools**, and **Model Artifacts**. While the coordination and integration aspects can be viewed as a type of accidental complexity, scaling up to a large project requires the synchronizing of work to avoid chaos.

In the following sub-chapters, the SAMEM details will be explained by following a typical development project as it proceeds from the higher abstraction levels to finer details. The details for the rationale, first principles, and lessons learned from the application on 18 industry projects will be interwoven. Chapter 4 contains more details about the companies involved in the case studies and the practical application of the SAMEM.

3.4 The SAMEM Process

The process approach is iterative & incremental throughout the whole project lifecycle. This means taking a small step forward, whether with requirements, design, or realization, and verifying that the small step is correct as soon as possible. The objective is to minimize the risk of investing time and effort in false work. By ensuring that the project stays on the correct path toward the solution Goal, the time and effort in completing the solution is also minimized.

Of the three SAMEM components in Figure 3.2, the **Process** component is primary. The **Model Artifacts** component will be adjusted to support the **Process** component and the **Tools** component will be adjusted to support the **Model Artifacts** and the

Process components.

For large projects, a project coordination process is needed to support business due diligence manage the return-on-investment, and support fiscal responsibility. The SAMEM recognizes the need, but does not mandate any specific approach. In Sub-chapter 4.3.1 an example of the project coordination process used in the case studies is described.

3.5 The SAMEM Methodology Framework

The initial cycles of gathering the requirements and understanding the domain often seem chaotic and random [Win96], [Amb02]. Others have written about the indirectness of the design process and the impact of design attempts on requirements [Bro10]. In order to help tame the chaos and provide some guidance on proceeding, some assistance is needed. In practice, the question is: *How does one do the iterations and make the incremental steps be forward progress?*

A process and mental framework is used to help guide the overall project process. A *framework* is defined as a structure for organizing the model artifacts, set of recommendations for the next tasks, and a roadmap for systematically advancing from a simple high-level statement of the problem to a realization acceptable to the stakeholders. The primary feature for a requirements methodology framework is the ability to manage the level of abstraction appropriately during the current stage in the development process.

In Figure 3.2, the SAMEM **Methodology Framework** component is the Open Distributed Processing – Reference Model (RM-ODP) [ISO98]. The reason for the selection of RM-ODP is the systematic progression from abstract expressions of the problem and solution domain to concrete expressions of the solution realization. It is the systematic progression that will provide guidance and control to setting the context of work for the iterations. The RM-ODP is also an ISO standard which implies validity to its ideas through the ISO standard review process.

In addition to the systematic progression from vague initial idea to complete solution, the RM-ODP helps achieve due diligence. As discussed in Sub-chapter 3.1.2, due diligence is a characteristic of professional engineering that implies thoroughness and quality of work. Not only are the decisions and their associated rationale part of due diligence, but also an evaluation of the consequences of the decision is included. The RM-ODP assists in achieving due diligence by the ordering of the work so that the more important decisions are taken first.

The RM-ODP framework consists of five viewpoints. The viewpoints correspond to the modularization guidelines from Parnas [Par72]. Each of the viewpoints provides a different focus for looking at the solution for a specific area of essential understanding. The five viewpoints are:

- **Enterprise Viewpoint**: A viewpoint on a system and its environment that focuses on the purpose, scope, and policies for that system.
- **Information Viewpoint**: A viewpoint on a system and its environment that focuses on the semantics of information and information processing.

- **Computational (Behavior) Viewpoint**: A viewpoint on a system and its environment which enables distribution through functional decomposition of the system into objects which interact at interfaces.
- **Engineering Viewpoint**: A viewpoint on a system and its environment that focuses on the mechanisms and functions required to support distributed interaction between objects in the system.
- **Technology Viewpoint**: A viewpoint on a system and its environment that focuses on the choice of technology in that system.

The RM-ODP standard is not blindly followed. Adaptations are made to provide the flexibility needed for an iterative & incremental approach and a pragmatic project process. For example, the Computational Viewpoint definition in the standard is relaxed to cover the behavior requirements of the solution. At the time of requirements elicitation, the objects of the system, as referred to in the RM-ODP standard, are unknown. The spirit of different levels of abstraction described in RM-ODP is used. The viewpoints help keep the people involved in the project focused on the task at hand. By controlling the progress from abstract to concrete, the options for innovative solutions are left open. Jumping to early conclusions or decisions is hindered.

A visual relationship of the viewpoints is shown in Figure 3.3. A similar graphic, but with customer-specific terminology, was developed to explain RM-ODP concepts with their usage to both the development team members and to the stakeholders. The SAMEM organizes the RM-ODP viewpoints into two categories: 1) Requirements in Business Vocabulary, the Enterprise, Information, and Computational (Behavior) Viewpoints, and 2) Design Models, the Engineering and Technology Viewpoints. The RM-ODP categories are used to help control and focus the work at the appropriate level of abstraction. For example, if a project team member offers an opinion on a specific database technology and the project is working on behavior requirements, then it is easy to table the suggestion for later consideration with the rationale that it is inappropriate at the current time.

As the project moves from the Enterprise Viewpoint through the Technology Viewpoint, increasingly detailed decisions are made. Each decision is an evolution step from the abstract to the more concrete and restricts the options at the next level of detail. For example, while eliciting business information requirements in the Information Viewpoint, design work of the Engineering Viewpoint is discouraged. However, as Brooks [Bro10] points out, sometimes a quick jump to more specifics can provide indications of realization risk or assurances of no realization risk.

The RM-ODP abstraction structure helps guide work through the separation of concepts and their language. The artifacts and communication at the Enterprise, Information, and Computational (Behavior) Viewpoints use business concepts and vocabulary. At the Engineering Viewpoint level, systems architecture concepts are used, such as the Model-View-Controller pattern, a database for persistent storage, or the Web for the user interface. The Technology Viewpoint uses the language of specific technologies, such as Java, HTML5, CSS, etc.

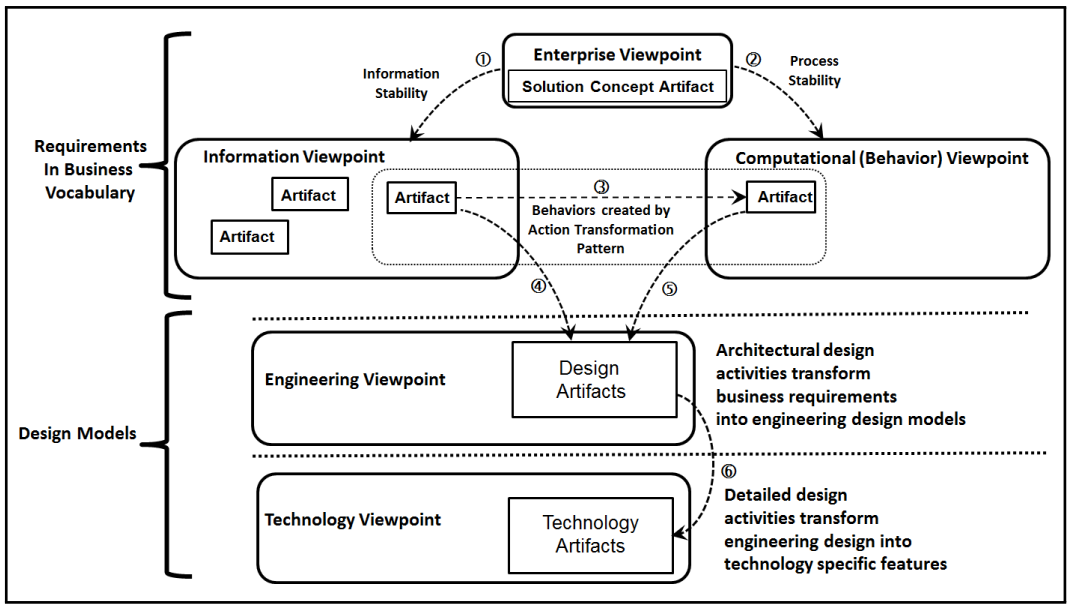


Figure 3.3: RM-ODP Viewpoint Relationships.

Besides the categorization of the viewpoints and the viewpoint structuring according to level of abstraction, there are important transition paths involved: ①, ②, ③, ④, ⑤, and ⑥. The transition paths shown in Figure 3.3 are the default paths for a normal project because of the abstraction flow. When the most stable aspect of the solution concept is information, then path ① to the Information Viewpoint makes the most sense. If the process in the solution concept is the most stable aspect, then path ② to the work in the Computational (Behavior) Viewpoint is probably a more optimal first refinement step. Transition path ③ involves applying a pattern to create behavior on the information artifacts. This pattern is described in more detail in Sub-chapter 3.5.3.

As Figure 3.3 shows, moving from the business concepts to the engineering concepts through paths ④ or ⑤ involves design choices about which engineering approaches best satisfy the requirements. Good engineering work should produce multiple design alternatives [Bro10], [Pet96], [Win96]. Refining a high-level engineering design to a specific set of realization technologies, traversing path ⑥, produces artifacts in the Technology Viewpoint. In the Technology Viewpoint another set of concepts and vocabulary exists. For example, the engineering design specifies a database (engineering concept) to meet the information model requirement of persistence (requirements concept), which leads to the technology design choice for the PostgreSQL product (technology concept), after considering other products and the business constraints.

The RM-ODP framework helps achieve the thesis goals of *better communication* by providing a context for the work being done and the purpose of the associated model artifacts. While in other methodologies, such as RUP [PK00], Volere [RR99], and Agile Modeling [Amb02], the communication is focused on the requirements artifacts, the

communication about the project status is just as important. The SEMAT approach has specific components to communicate the project status. Besides the communication of the artifacts, the abstraction transition shines a light on the rationale for the design options and choices which enables them to be documented as well. The design rationale and documentation support the demonstration of due diligence.

The thesis goal of faster project process (***HL-Goal-4***) is supported by the systematic but flexible progression through the different types of work involved in the project. The systematic progression guided by the RM-ODP framework helps to minimize the time spent in redoing work because a decision was made out of its proper sequence.

3.5.1 Objective of the Enterprise Viewpoint

The objective of the RM-ODP Enterprise Viewpoint is to specify the business purpose and benefit of the solution to all the people involved. This matches with the idea of specifying the business *Goal* and *solution concept* discussed earlier. The Enterprise Viewpoint is used at the beginning of the project to keep the discussion focused on the proper level of abstraction. The proper abstraction level of the questions that are "What?" and "Why?", not "How?".

While the specification of the solution concept and the mantra, as shown in Figure 3.1, is an essential part of the Enterprise Viewpoint, it is not enough. The project experiences uncovered a communication gap of an overall picture of the solution which led to the development of the Solution Overview Drawing (SOD). An example of a SOD is shown in Figure 3.5. The SOD communicates in more detail than the solution concept the scope of the project and is a visual summary of the requirements. In brief, the SOD is an assembly of models created during the requirements elicitation activities of the RM-ODP Information and Computational (Behavior) Viewpoints. The SOD contains at a high level of abstraction and a somewhat more symbolic representation the essence of the solution process (behavior) together with the solution information model. The SOD is incrementally built up through the iterations for requirements elicitation, so that it provides a *conceptual integrity* feedback mechanism from the more concrete work of the Information and Computational (Behavior) Viewpoint to the Enterprise Viewpoint.

In practice, the SOD is used at the start of almost every meeting as the basis for setting the meeting context and for verification review. Colloquially it was called the "1-pager" with the stakeholders as it fit on a single 11x17 inch or A3 size paper. In the industry projects, the SOD became part of the requirements specification. The SOD examples were reviewed by an FDA auditor [1] and were found to be more helpful in understanding the intent of the product than corresponding text explanations.

The SOD is an example of the SAMEM fulfilling the following thesis goals through a novel approach:

- ***HL-GOAL-1***: better stakeholder communication.

[1] An FDA auditor is an employee of the FDA that checks a business, usually with respect to a particular product, for compliance to the regulations.

- ***HL-GOAL-3***: requirements in more compact format.
- ***GOAL-5***: maximize the use of visual models for communication.
- ***GOAL-9***: flexibility in visual models for communication optimization.
- ***GOAL-1***: consistent presentation to optimize communication.
- ***GOAL-18***: visual artifacts must not be limited by existing notations.
- ***GOAL-19***: visual models must be good enough for the purpose.

3.5.1.1 Examples of Enterprise Viewpoint Model Artifacts

In the case studies, there are two primary visual model types used to document and communicate the Enterprise Viewpoint. The first model type is for the solution concept. It embodies the solution concept at the highest level of abstraction. All modeling refinement to greater detail starts with this model. The second model type is an assembly of more specific models from the Information and Computational (Behavior) Viewpoints, called the Solution Overview Drawing (SOD). In assembling the SOD, more abstract representations of information and behavior are included, so that the SOD fits on one page of size A3 or US 11x17 inch.

Figure 3.1 shows an example of the solution concept model. One important communication characteristic of this model is that it should fit on one page. The puzzle communicates that the Product Definition is assembled from multiple interconnected groups of Information Models (IM). The value of this particular image is that it can be refined while keeping the same format. For example, the Design History IM puzzle piece from Figure 3.1 shows the next level of refinement as several puzzle pieces with more detailed information models in Figure 3.4. While keeping an image style through refinement enhances the communication, at some point the image loses its communication effectiveness. The images and text details, in Figure 3.4, such as case study specific jargon of "CRM Library" and "Tox. Library," are not optimally represented using the jigsaw puzzle piece metaphor when going to the next level of detail. A more appropriate format based on the UML for these details is the information unit and is addressed in Sub-chapter 3.5.3. When a communication style transition takes place, it is often an indication of an abstraction transition to a more concrete form.

Figure 3.1 provides the first point of agreement with the customer. The objective during the project is to stay true to the agreement (CI SEFP). When there are questions or concerns, the refinements can be traced back to this image for confirmation of direction.

The Solution Overview Drawing (SOD), Research Note example in Figure 3.5, is an assembly drawing of the primary requirement models done for each Information Model indicated either in the solution concept model (Figure 3.1) or in a direct refinement (Figure 3.4). In a large project, there can be multiple SODs. The number of SODs is dependent on clear communication and a pragmatic division of work. For the complete solution, there is a set of SODs representing all of the IMs. This is an application of the

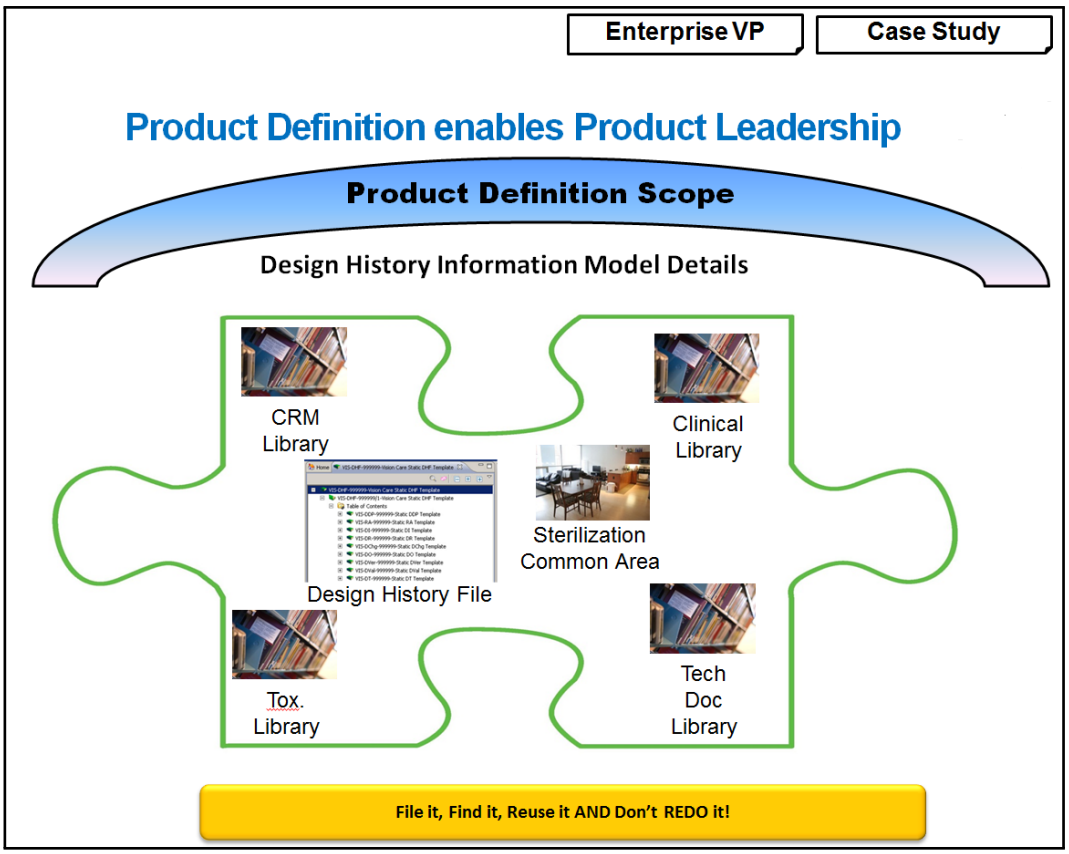


Figure 3.4: Design History Puzzle Piece Refinement.

modularization ideas from Parnas [Par72] as embodied in the Module SEFP. The SOD displayed in Figure 3.5 defines the responsibility scope for the module of the Information Model (IM) of Research Notes as shown in one of the jigsaw puzzle pieces of Figure 3.1.

While the SOD is considered to be part of the Enterprise Viewpoint, it is developed over time as the Information and Computational Viewpoint models are discovered and refined. The rationale for considering the SOD as part of the Enterprise Viewpoint is that it is viewed as an abstract summary of the major requirements.

From stakeholder communication experience, the SOD was designed for a horizontal or landscape 11x17 inch or A3 sized piece of paper and intended to be printed, it was referred to as a "1-pager" in the case studies. In Figure 3.5, the SOD image shows the standard layout that was developed which has been rotated counterclockwise to maximize resolution, although the reduction in size will make some text illegible. Higher resolution images of the components of Figure 3.5 are shown in Figure 3.6, Figure 3.7, Figure 3.8, and Figure 3.9. Having some open space on the SOD is very useful for stakeholder review comments. Each SOD has four major components, which are described in detail and there is a standardized format for each major component to support communication consistency:

1. (label ①)In the upper right corner is an abstraction of the information units and structures of the Information Model. Across multiple SODs the same colors and graphical forms are used for communication consistency. The objective is to balance the limited space with enough detail to communicate the essence of the information units and structure. Figure 3.6 shows this component in more detail.

2. (label ②) The upper middle section holds the state machine or machines for the information units. The state machine describes the transitions from new and unreliable information to approved and trusted information. Figure 3.7 is a close-up of the state machines.

3. (label ③)The upper left corner contains text notes with significant requirements that are best expressed in text or known constraints. Figure 3.8 shows the notes in greater detail.

4. (label ④)The majority of the space on the SOD, the bottom half of the drawing or more, is for the Business Flow. The details of the Business Flow are explained in Sub-chapter 3.5.3. If the Business Flow is large, then it is abstracted until it fits or as a communication alternative several sub-SODs are produced with greater detail. Figure 3.9 shows the Business Flow at greater resolution.

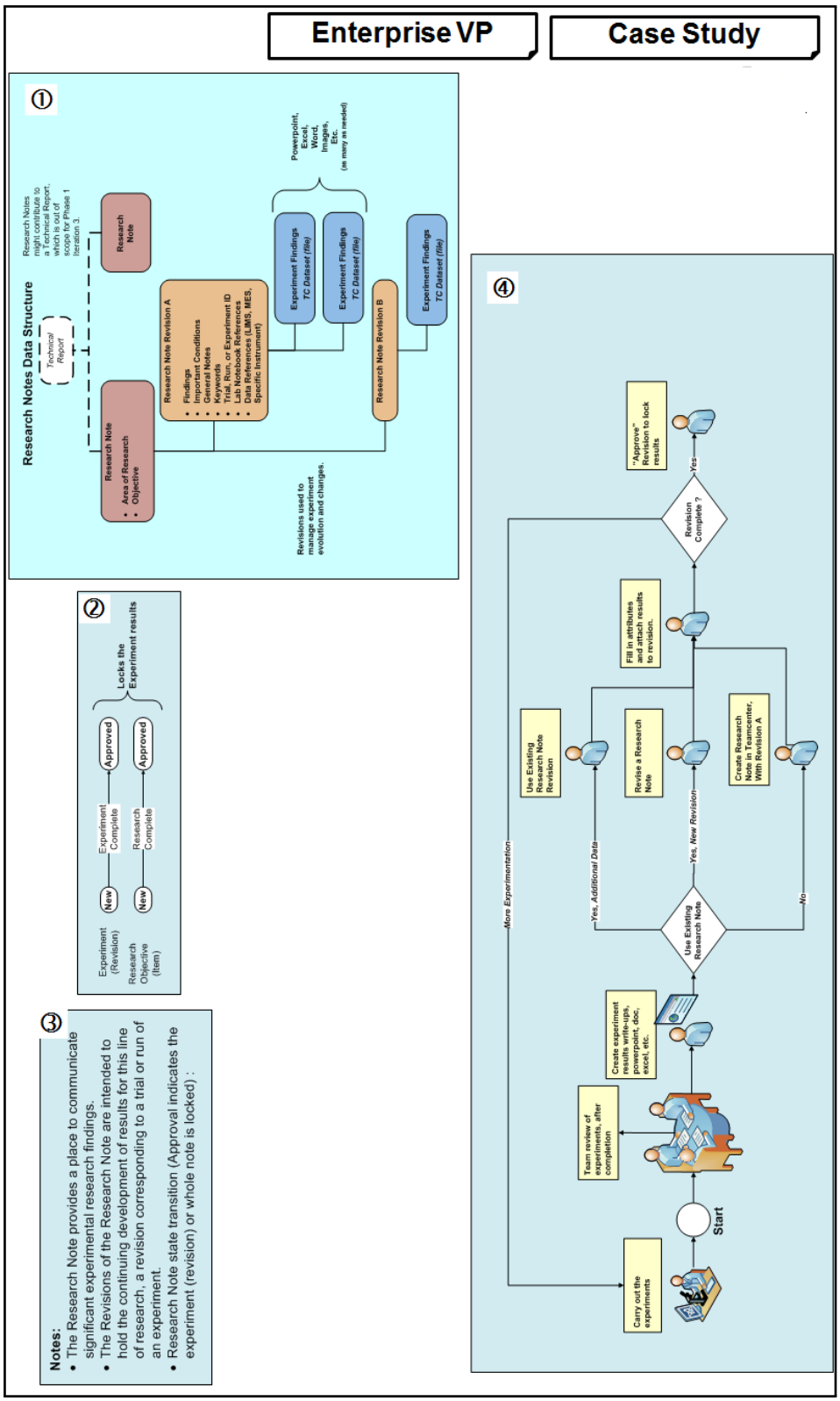


Figure 3.5: SOD Standard Layout Overview Example.

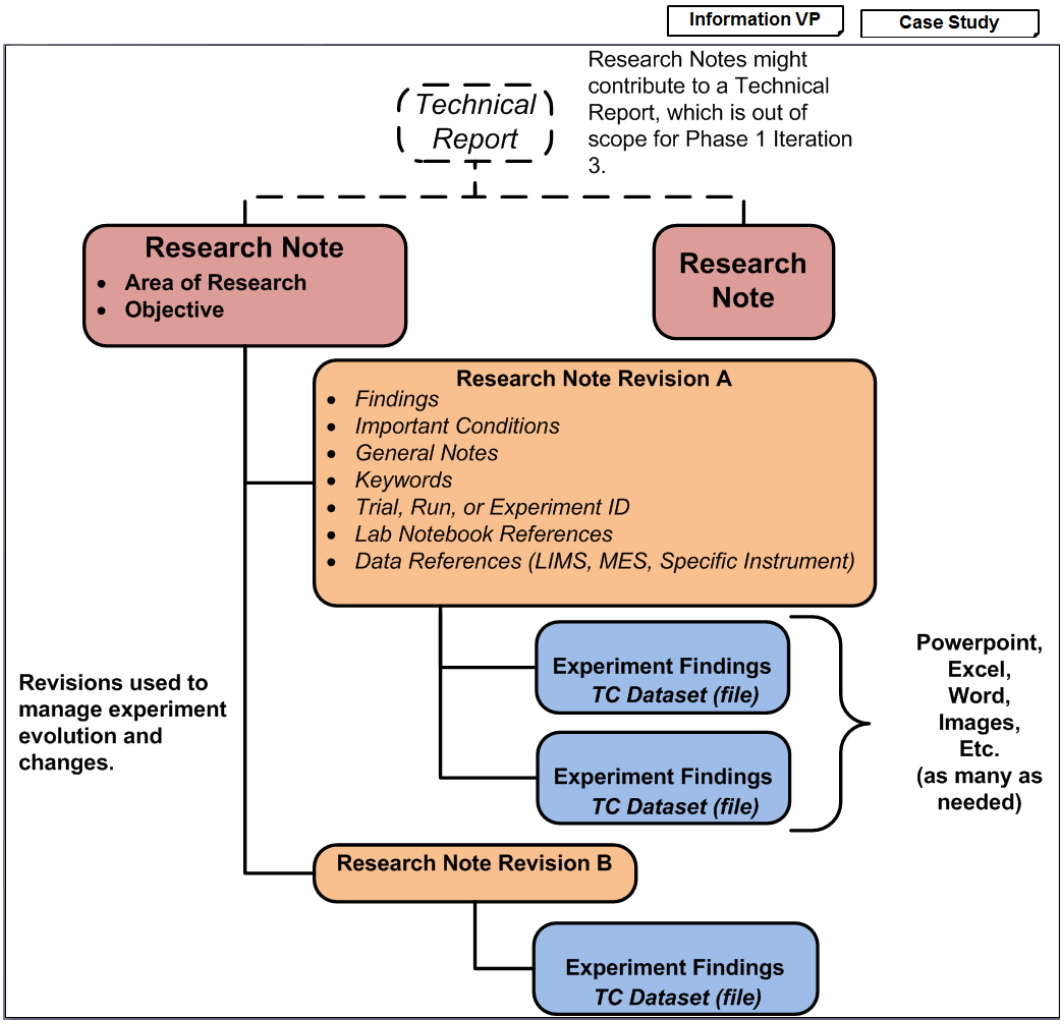


Figure 3.6: SOD Information Model Section Detail.

Figure 3.6 shows an abstract version of the full information model for the Research Notes IM, which is used in the SOD. It contains two Research Note Items with one expanded to illustrate the use of revisioning within the Research Note Item, Research Note Revision A, and Research Note Revision B. The expanded Research Note Revision A box also shows the significant business properties, the ones most used for finding a Research Note, but not all of the business properties. The complete definition of business properties for an information unit is specified in other artifacts, see Sub-chapter 3.5.2.1 for examples. The Technical Report object and dashed lines connecting it to the Research Notes indicate that multiple Research Notes might be used as input to create a Technical Report, which is the subject of a separate IM and a dashed line visual notation is used to indicate that those items are separate from this model but related. This case study example illustrates a lesson learned, which is to include additional clarifying text to

enhance stakeholder communication. In this example, the large curly bracket character by the two Experimental Findings boxes attached to Research Note Revision A indicates that the data could be in a variety of forms such as MS PowerPoint™, MS Excel™, MS Word™, or some other format. The data format is appropriate to the nature of the experimental data.

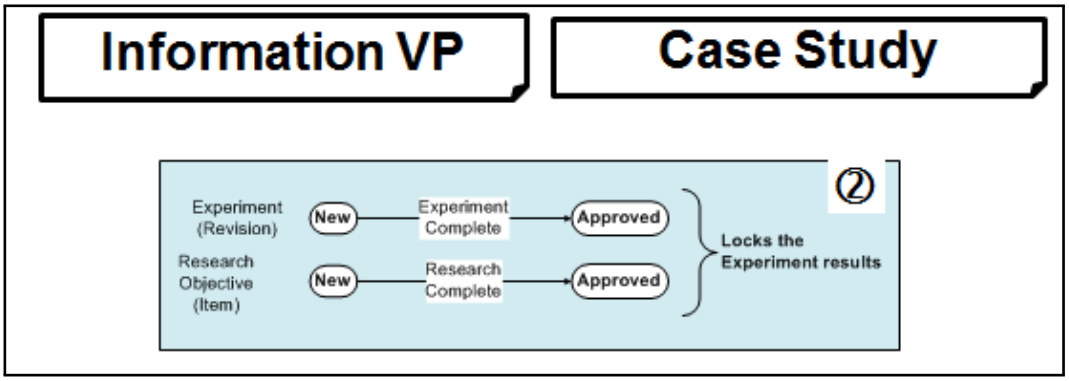


Figure 3.7: SOD State Machine Section Detail.

The two state machines shown in Figure 3.7 are part of the information model, even though state machines are usually considered behavior. The reason for considering the state machines as part of the information model is because they show the possible values for the *Status* property, or attribute of the Research Note Item, or one of its Revisions. There is a state machine that changes the status of the Research Note Item to *Approved* and a separate state machine for the Research Note Revision.

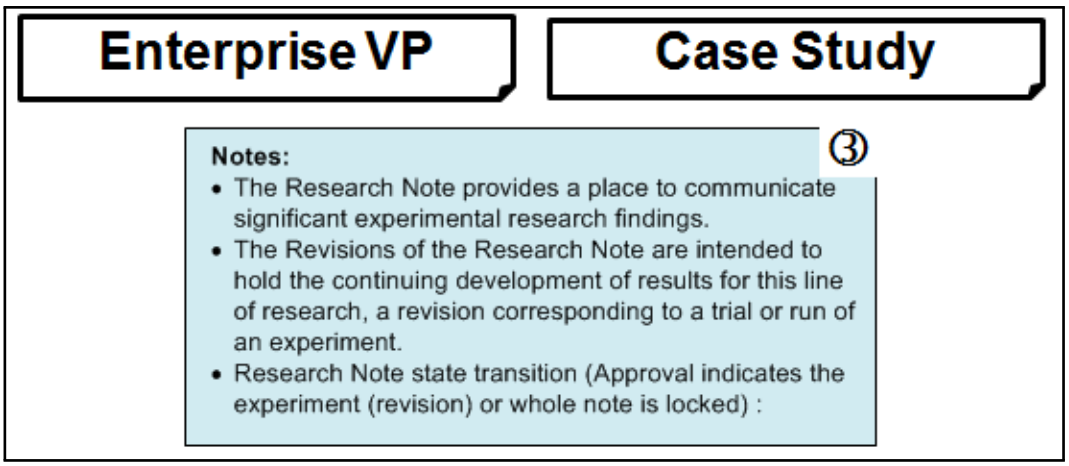


Figure 3.8: SOD Notes Section Detail.

The Notes section of the SOD, Figure 3.8, normally has a few points in text that clarify requirements or list restrictions best expressed in text. In some cases, the notes section will contain questions arising from the review tasks. The contents can vary from SOD

to SOD instance and will evolve within an SOD as progress is made on the information models or behavior models.

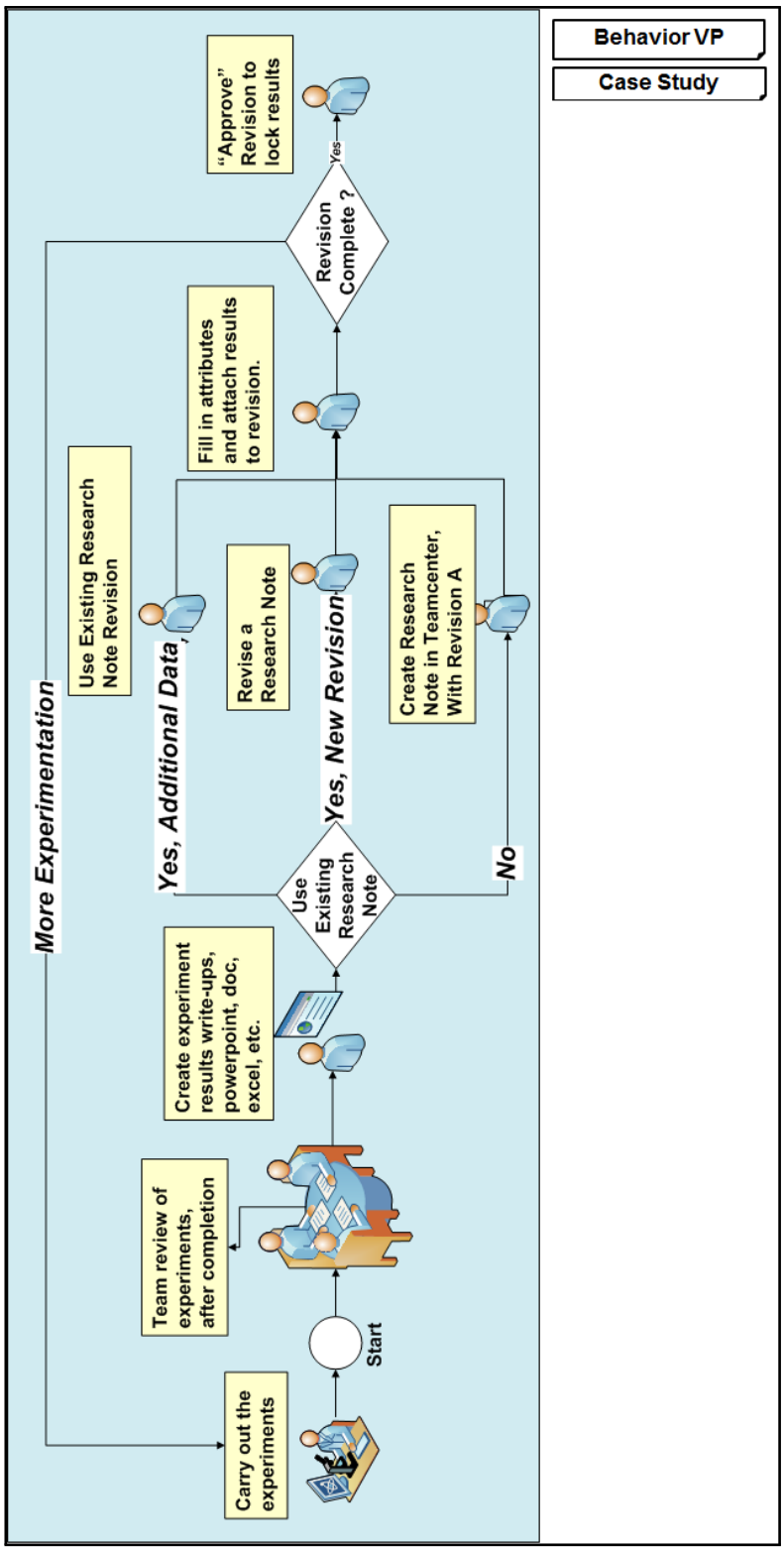


Figure 3.9: SOD Business Flow Section Detail.

The main Computational (Behavior) Viewpoint component of the SOD is the Business Flow as shown in Figure 3.9. The Business Flow is a business process model, similar to or compatible with the UML Activity Model or Business Process Modeling Notation (BPMN). It describes the lifecycle process for the Research Note information model, from its creation, the *New* status, to its finish, the *Approved* status. In general and relative to the information model, the Business Flow in the SOD specifies how the business works. Some of the tasks in the Business Flow are automated in the solution, but some tasks are not, so that the Business Flow is the complete picture of the tasks, not just the parts of the automated solution. A complete explanation of the Business Flow Model can be found in Sub-chapter 3.5.3.

3.5.2 Objective of the Information Viewpoint

The RM-ODP Information Viewpoint holds at a business level the information definitions for the solution Information Models (IM). In the solution concept model of Figure 3.1, there are eight different top-level IMs assembled to make up the product definition. There are actually more than eight for the whole business, but in this case study the scope of the project was limited to the eight listed in Figure 3.1. Figure 3.4 holds the refinement of one top-level IM and this is the subject of an Information Viewpoint increment. The benefit of this viewpoint is in keeping the discussions at the business level and avoiding computer science data structure like vocabulary during the requirements elicitation work. Computer science terms such as object, class, tree, linked list, and array were explicitly avoided. Neutral terms of "information unit" and "information structure" were used in the case studies, this is a lesson learned on effective communication with non-computer literate stakeholders. In Sub-chapter 4.1 more detail about the case study companies and the stakeholder backgrounds is discussed.

UML Class Models provided the initial starting point for the information artifacts. However, a lesson learned was that the stakeholder communication did not work well with standard UML artifacts [Moo09], [Mat11]. In brief, the lesson learned was the visual simplicity of form, line figures, and the associated complexities needed for code development interfered with the communication. More about this lesson learned is discussed in Sub-chapter 4.4.1.3. While keeping with the intent of the UML Class Model paradigm, several of the rules and behaviors are relaxed to achieve effective artifacts. A simplified Information Unit Model is the replacement for the UML Class Model. Other researchers have also reached a similar conclusion on UML communication limitations for non-programmers [Amb02], [BC12].

The Information Unit Model for the SAMEM separates the data aspects from the behavior on the data, which is the domain of the Computational (Behavior) Viewpoint. The basic characteristics are attributes for data values, relationships to build parent-child structures, and relationships between units for other business structures. The Information Unit Model also includes an associated State Machine Model which describes the completeness and trustworthiness of the Information Unit Model. In practice, a newly created Information Unit Model is not complete or fully defined when first created. The entry of attribute data values or the completing of a structure can take an extended

amount of time (days) and involve multiple people.

Stakeholder communication and comprehension is the major impact of creating the Information Unit Model, which impact the thesis goals of *better stakeholder and development communication*. The second major factor in the design of the Information Unit Model is artifacts that support the thesis goals of *faster projects* and a *more compact form* for the requirement specification. The specific thesis goals that are supported by explicitly separating the solution information requirements and representing them as graphical models are:

- ***HL-GOAL-1***, better stakeholder communication.
- ***HL-GOAL-3***, improved requirements quality in more compact format.
- ***GOAL-1***, accurate communication mechanisms.
- ***GOAL-3***, traceable artifact evolution.
- ***GOAL-5***, maximize visual models for communication.
- ***GOAL-9***, flexibility in visual models for communication.
- ***GOAL-18***, visual artifacts must not be limited by existing notations.
- ***GOAL-19***, visual models must be good enough for the purpose.

Rather than specialized tools, Microsoft Office™tools are employed for the initial creation and updating of the requirements information models. This is a benefit to communications because stakeholders are familiar with the tools and it impacts the project speed because the artifacts can be exchanged with the stakeholders via email to facilitate review.

A more detailed description with a corresponding metamodel is the subject of Chapter 6. For tracking and traceability purposes, each information unit requirement gets a unique identifier.

3.5.2.1 Information Viewpoint Requirements Model Artifact Examples

The first application of the SAMEM was for manufactured products. The SEFP of "Stability to Variability" (**S2V**) directed the team to look at the existing product definition information as the starting point for developing the Informational Viewpoint artifacts. This is information in R&D collected during the product design process, information in manufacturing for product production, and information in end-customer support of the product. Often the information is scattered over multiple documents in a somewhat disorganized manner. The FDA regulations on required information for product approval were the second helpful source.

The information unit and structure models are assembled over multiple meetings (increments) and each change verified with the stakeholders (iteration). The verification is

focused on the contribution of the additional information units added during the iteration to achieve the mantra and the consistency with the solution concept. This process eliminated information previously thought necessary by the stakeholders from the new product definition.

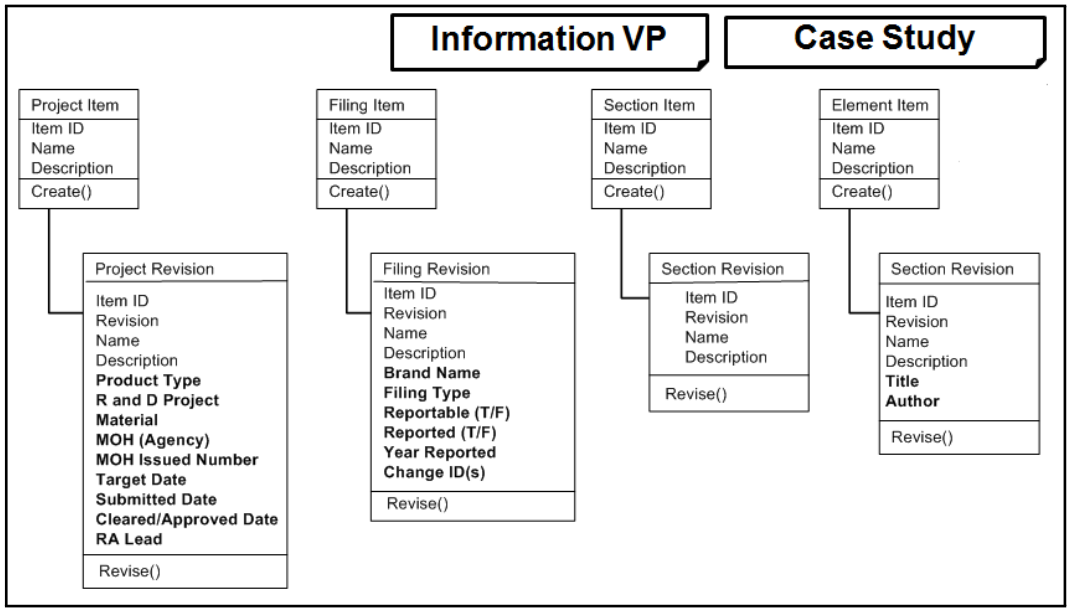


Figure 3.10: Information Model Example of Basic Information Units.

Figure 3.10 shows an example of the basic information units for a FDA regulatory submission solution. While there is a great similarity to a UML Class Model, this version is simpler. There are eight information units: a Project Item (a structure unit), a Project Revision, a Filing Item (a structure unit), a Filing Revision, a Section Item (a structure unit), a Section Revision, an Element Item (a leaf unit), and an Element Revision. The term *structure unit* indicates that this item can be involved in a parent-child structure as the ultimate parent or an intermediate parent; an example can be seen in Figure 3.13. The term leaf unit means that the item is only involved in a structure as a final child element. The terms Project, Filing, Section, and Element are customer vocabulary.

The *Items* and the *Revisions* form pairs of complete information units. Each *Item* can have multiple revisions which track the evolution of that information over time and approval cycles (see Sub-chapter 3.5.3). The Revisions are ordered in time and are owned by only one *Item*. The purpose of the *Item* is to uniquely identify each separate business information unit instance, such as a specific FDA regulatory submission project. Each *Item* unit has an associated set of revisions to track the evolution of the information unit. The *Revision* unit isolates the data that can change multiple times over the course of the project. The *Item* and *Revision* units are examples of the application of modules from Parnas [Par72]. Within each unit, the major attributes are listed and in this example a

method for the major action, although that is not always necessary. These units follow a data structuring pattern from the OMG Product Data Management Enablers [OMG00] standard.

In Figure 3.11, an image of the attribute table associated with the Filing item information unit of Figure 3.10 is shown. Placing this information in a separate table is a response to a lesson learned that improved the customer communication, which is a thesis goal. The splitting of the artifact representations, which is contrary to the UML Class idea, helps the communication by enhancing focus. In Figure 3.10, the focus is on the number of basic units of information, their differentiation, and any possible structures. With the table shown in Figure 3.11, the focus is on the attributes or properties of a single information unit.

Information VP | Case Study

Table 1: Regulatory Filing Item Properties

Name	Type	Values	Description	Origin	Standard Search	Required
Item ID	*Text*	*REGFLG-nnnnnn Starting at 000001*	*Unique Regulatory filing identifier*	*S*	*No*	*CR, Sys Gen*
Name	*Text (Limit 240 characters)*	*Any text*	*Also referred to as a 'nick name'*	*U*	*No*	*CR*
Description	*Text (Limit 256 characters)*	*Any text*	*Simple description*	*U*	*No*	*No*
MOH Issued Number	**Text (Limit 256 characters)**	**Any text**	**The filing number issued by the agency. To be populated after the submission is made and before the 'Mark as Complete' workflow process is executed. This property is displayed in the Filing Revision as a compound property.**	**U**	**Yes**	**CM**
Submitted Date	**Date**	**Date**	**The date the filing was submitted to the MOH office. To be populated after the submission is made and before the 'Mark as Complete' workflow process is executed. This property is displayed in the Filing Revision as a compound property.**	**U**	**Yes**	**CM**
Cleared/ Approved Date	**Date**	**Date**	**The date on which the filing was cleared or approved. To be populated after the submission is made and before the 'Mark as Complete' workflow process is executed. This property is displayed in the Filing Revision as a compound property.**	**U**	**No**	**CM**

Figure 3.11: Information Model Example of Attribute Table.

The example in Figure 3.11 does not list all the attribute rows from the case study, but does show the columns with the characteristics of the attribute. Some of the attributes (actual case study names displayed) are user defined for these information units (MOH Issued Number, Submitted Date, Cleared/Approved Date), and some are application built-in attributes that match the solution needs (Item ID, Name, Description). The columns represent the attribute characteristics important for the requirements level. They are:

- **Name** – to enable effective searching for information, each attribute within an information unit has a unique name. When an attribute has the same business

purpose across multiple information units, the same name value is used for user interface consistency.

- **Type** – the data type of the attribute is specified, such as Text, Date, Integer, etc.
- **Values** – the legal values for the attribute are specified in this column. In some cases, an attribute will have an initial value assigned on the creation of a new information unit.
- **Description** – this contains a business rationale of why the attribute is important. This column is extremely valuable in minimizing the number of attributes. It prevented the "wouldn't it be nice to have" accumulation of low value attributes.
- **Origin** – this column is started in the requirements work and completed during the design activities, usually in the Technology Viewpoint when the out-of-the-box attributes of the COTS system were examined for possible use. An *S* indicates that the system controls the value, while a *U* indicates that the user enters the value.
- **Standard Search** – whether the attribute is part of the standard search or is in the full search is indicated by a *No* or *Yes* entry. The standard search has a small number of the most used attributes and is designed to fit on a single screen. The full search has all the attributes and often is multiple screens in size, therefore requiring the user to scroll. This is a usability criterion that was discovered in the Demo Prototype Phase feedback and was an improvement feedback to the requirements elicitation work [Bro10].
- **Required** – indicates whether this attribute must have a value set in the information unit creation step. The goal is to minimize the number of attributes that need to be set at creation. This is both a usability benefit and a realization that some attribute values might take months to discover.

In the following figures, three levels of information structure models for an equipment engineering solution are presented as examples. There is a simple specification in Figure 3.12, a simple project in Figure 3.13, and a large project in Figure 3.14. The figures show how a basic information building block can be assembled into a more complex structure. The simple project in Figure 3.13 introduces the information units of *Engineering Project* and *Engineering Review*, which have a parent-child relationship. In Figure 3.14, an example of sub-projects within a larger project, i.e. structuring of projects is laid out, but the basic project information unit is the same. The project hierarchy is built up to match the equipment engineering project complexity.

Figure 3.12 shows an example of an *Engineering Spec Item* with the most important four attributes and its first *Engineering Spec Revision* with its four most important attributes. This case study example in comparison to the case study example in Figure 3.10 shows the need of the visual notations to be flexible enough to optimize communication with the different stakeholders. A lesson learned is that color in the artifact

models varies from group to group. For example, the engineering group found color is helpful in the communication versus the monochrome visual images in the regulatory group as shown in Figure 3.10.

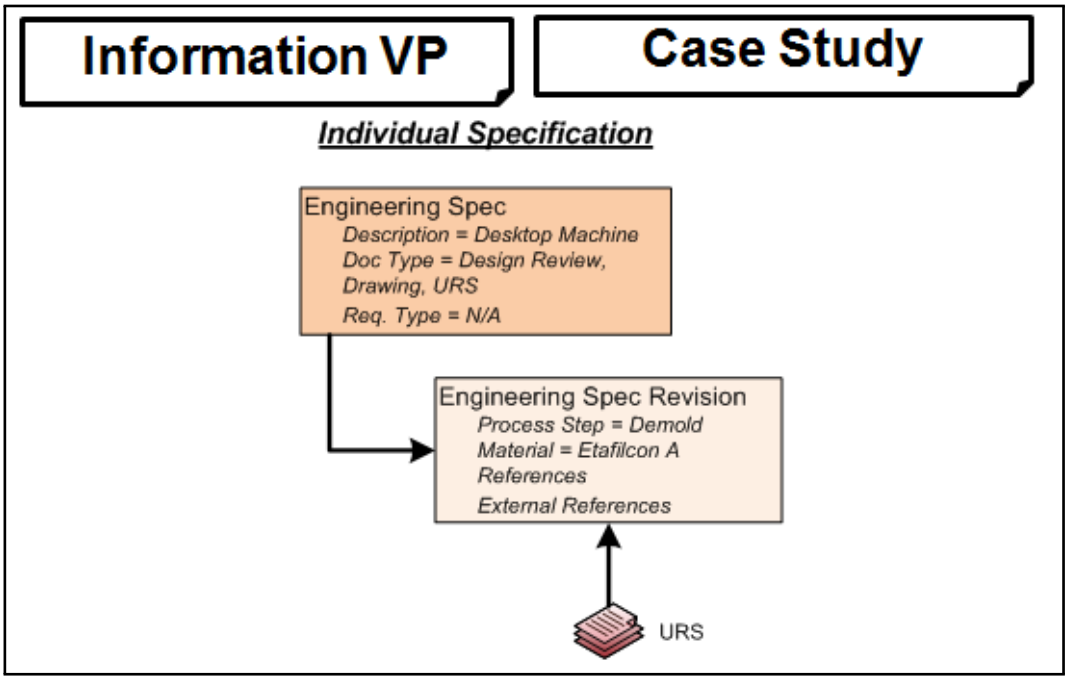


Figure 3.12: Information Model for a Single Specification.

Figure 3.13 is an example of a simple *Engineering Project* structure. There are three child elements of the *Engineering Project Revision*, two *Engineering Spec* items, and one *Engineering Review* item with respective revisions. The structure is stored under a revision which enables *Engineering Project* revisions to capture the changes in structure. Figure 3.13 shows the inclusion of useful text explanations with the visual notations.

A more complicated structure from a case study is illustrated in Figure 3.14. In this example, a common solution is shown where the structural information units that are used to build the parent-child relations are defined so that they can be either a parent or a child. The example shows a *Main Project* with three *Sub Projects* in the structure. Each *Sub Project* is structured similar to the example in Figure 3.13. The *Section Items* of Figure 3.10 are also defined in this manner. When presented in the correct way to the stakeholders, through visual notations, they understood the flexible building block approach.

Figure 3.15 shows an example of a report produced by the solution. The figure has been rotated to maximize the size on the page. While some text is barely legible, the intent is to show how requirements can be gathered and specified in other visual formats. The requirement source for this report was an existing report. However, in the past the report was created by hand in a spreadsheet program by reading many documents, emails, and

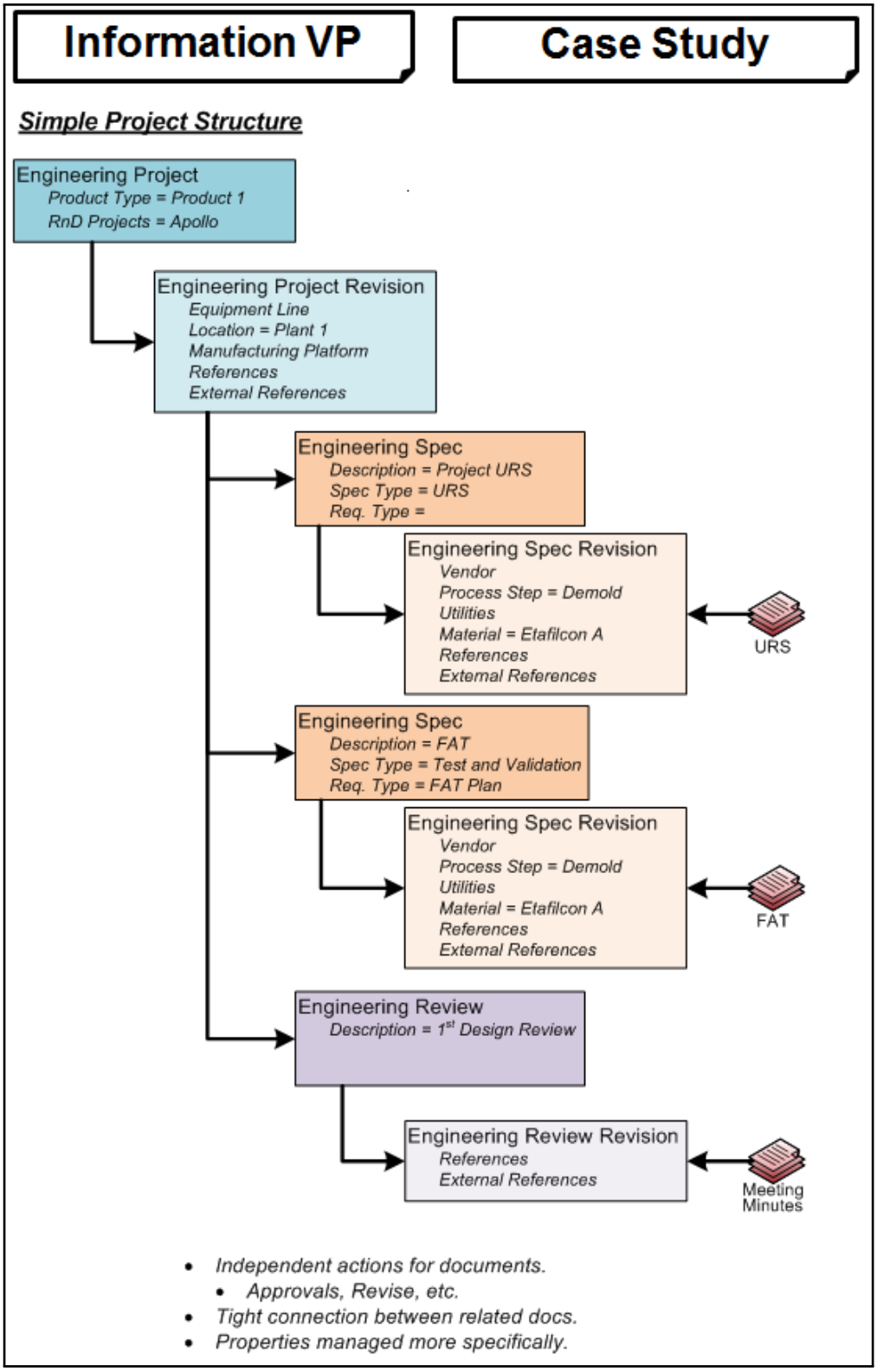


Figure 3.13: Information Model Example of a Simple Project Structure.

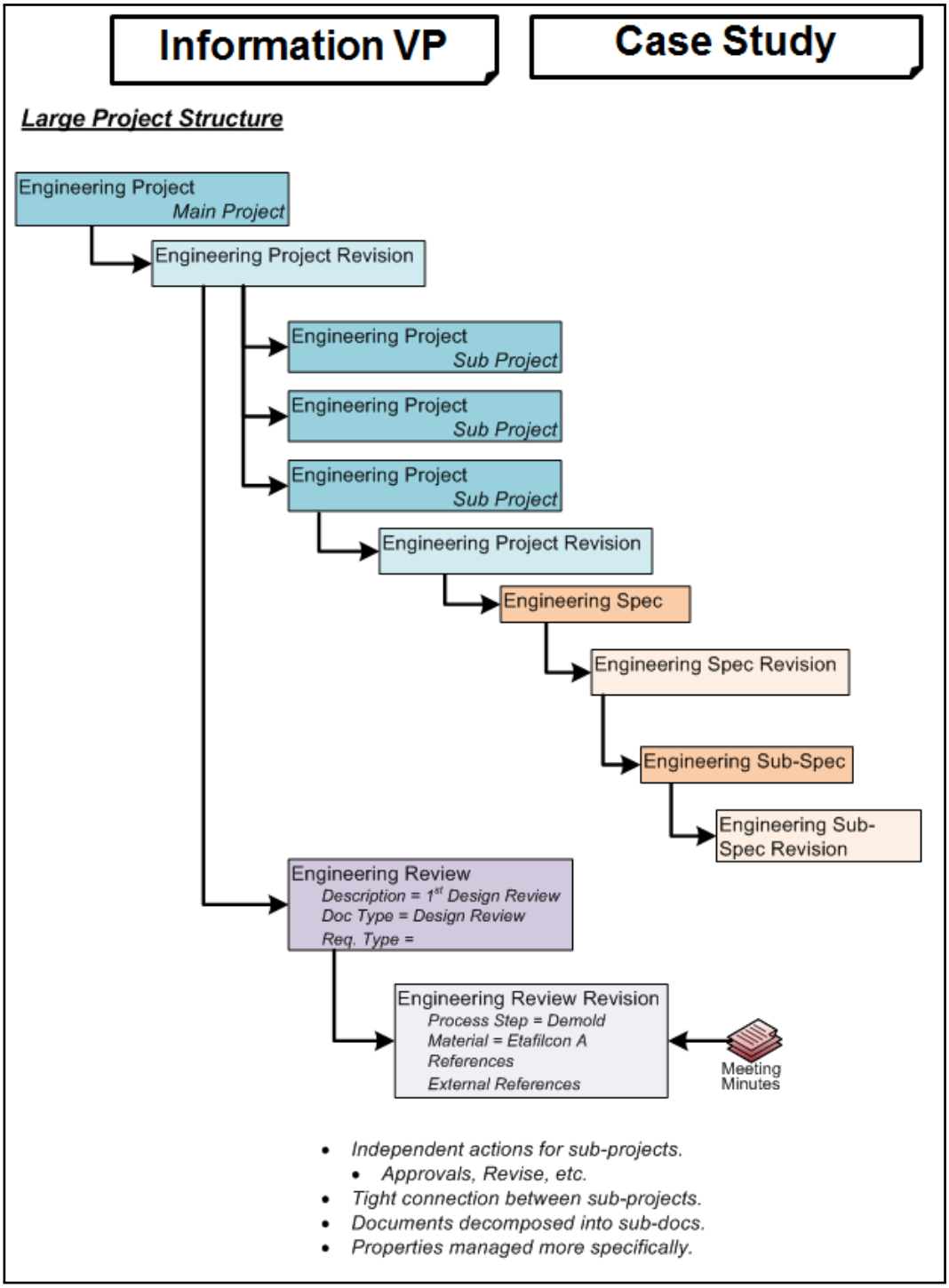


Figure 3.14: Information Model Example of a Complex Structure.

talking with several people to gather the data, which often resulted in transcription errors. The existing hand constructed report brought to light several data values that needed be to added to the Information Model. This is an example of an electronic solution enabling the automatic production of the report, thereby saving a manager many days of effort, improving the accuracy and timeliness of the report. The report was created by traversing the regulatory project information structures and examining attribute values. The need of producing the report had impact on the information structures to minimize the generation work.

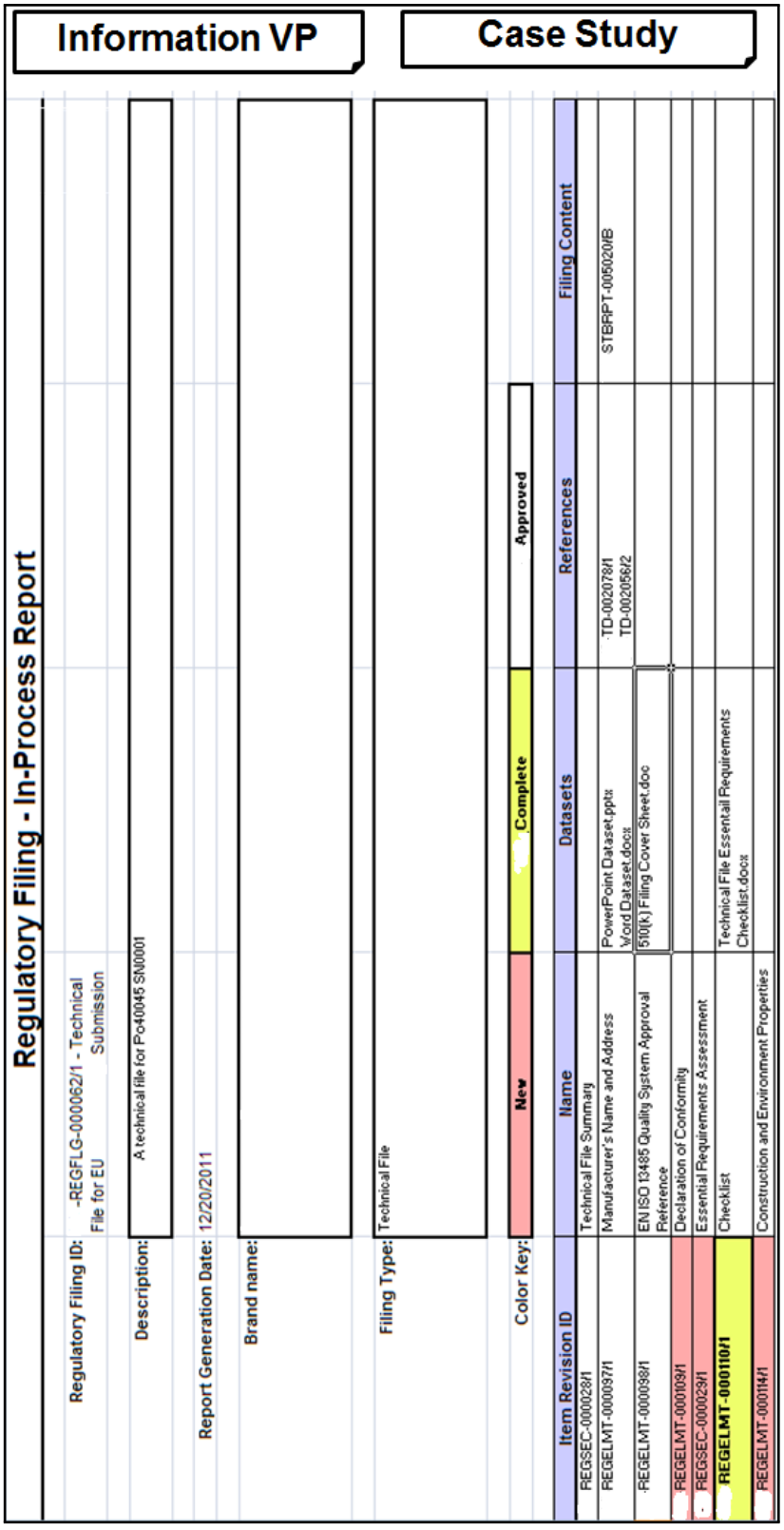


Figure 3.15: Report Example.

3.5.3 Objectives of the Computational (Behavior) Viewpoint

The official definition of the Computational Viewpoint of RM-ODP [ISO-10746] is stated as follows:

> "A computational specification defines the functional decomposition of an ODP system into objects which interact at interfaces. In the computational viewpoint, applications, and ODP functions consist of configurations of interacting computational objects."

However, for the purposes of the SAMEM and the elicitation of requirements, a modified definition is more useful. The new definition is focused on the *business behavior* needed in the solution; therefore, the change in viewpoint title within the SAMEM is to Computational (Behavior) Viewpoint. The decomposition into objects, which interact at interfaces, is left to the design work done in the Engineering Viewpoint or Technology Viewpoint. The rationale for this shift is to keep engineering and technology decisions from creeping into the requirement specifications, thereby limiting solution alternatives. For example, the computational objects can be different depending on the application of code-level design patterns such as *Factory* or *Observer*. These pattern choices are design decisions, not requirements.

There are two primary approaches to defining the business behavior. The first is to specify the business process used to create an important part of the information model. In the SAMEM, the business process is called the *Business Flow*. The *Business Flow* consists of the major business steps or tasks used to create a significant part of the information model, starting with nothing, then proceeding to the completed business information. Each *Business Flow* is specified using business vocabulary, which could be a combination of the standard business domain vocabulary and company-specific terms. The tasks are specified without any assumption as to how they will be accomplished. Perhaps some tasks will be supported by the computer-based solution under development, some by existing solutions, and some will remain manual. The *Business Flow* can be effectively expressed using UML Activity Models or BPMN.

The second approach is to apply the Create-Retrieve-Update-Delete (CRUD) pattern on the information model units used within the *Business Flow*. The CRUD pattern acts as a completeness check on the tasks in the *Business Flow*. Each CRUD action should appear in a *Business Flow* associated with an information model unit. It is possible to have more than one Business Flow associated with an information model if that is how a company views its business processes. In the medical device examples used here, information is never deleted for regulatory auditing reasons, but rather marked as *obsolete*.

The second use of UML Activity Models in the Computational (Behavior) Viewpoint is the *Approval Flow* which changes the status or state of the information unit or set of units. The *Approval Flow* is represented by a task in the *Business Flow* and its details are a level more specific. The important behaviors that are specified in an *Approval Flow* are the following: who is responsible, the order of review, and approval tasks.

The responsibilities are classified as an expertise *Group*, or an authority *Role*, or a combination of both. For example, a manager (responsibility role) in the analytical chemistry department (expertise group) is needed to approve. A complementary effect of the specifying of the *Approval Flow* requirement model is the creation of an Organization Model.

The solution Organization Model is considered to be a component of the Computational (Behavior) Viewpoint. A portion of the Organization Model is discovered and developed through the definition of the *Approval Flows*. Another portion is developed through security analysis of the *Business Flow* tasks. At each task, the answers to the following questions will generate contents for the Organization Model:

- Who does this task?
- Who should be prevented from doing this task?
- What access rights do they need to accomplish the task?
- Which actions of the CRUD pattern are involved?
- Who needs notification that the task was accomplished?
- Are there privacy concerns that limit the notifications?

The questions help to discover the requirements concerning the security aspects of access and authorization on the information and the actions on the information. In Figure 3.18, Figure 3.20, Figure 3.21, and Figure 3.22 examples of the organization requirements of access and authorization can be found. The details are collected into the full Organization Model.

The next part of the behavior requirements specification work is to examine the tasks in the Business Flow and determine which ones will be done in the solution. This planning work is accomplished near the end of the behavior modeling and is a transition activity into the design work of the Engineering Viewpoint. The common reasons that some tasks will not be supported by the new solution are that an existing tool is in place or it is a manual task that should stay manual. Once the tasks are identified, a *Use Case* is defined at the business behavior level.

The *Use Case* is the final component of the Computational (Behavior) Viewpoint collection of models. It is not the UML Use Case, rather it is a list of steps and conditions in business terms of what work takes place in the *Business Flow* task. To specify the *Use Case* requirements, a common industry template is used.

In some cases *User Scenarios* are constructed to improve understanding of sections of the *Business Flow*. A *User Scenario* is a small set of *Use Cases* with additional information describing the flow from *Use Case* to *Use Case* in business terms. In some situations, a *User Scenario* is used to describe a common variant to the *Business Flow*.

There are several mechanisms for displaying the behavior requirements in the Computational Viewpoint. The rigor behind *Business Flow* and *Approval Flow* is provided by

the UML Activity Model or by BPMN. The other factor is the application of the CRUD pattern to each information unit from the Information Viewpoint.

Each behavior requirement gets a unique identifier.

3.5.3.1 Examples of the Computational (Behavior) Viewpoint Model Artifacts

An example of the Business Flow is shown in Figure 3.16 (same as Figure 3.9, but repeated for convenience). This shows the flow of business tasks in the creation, updating, and approval of a Research Note. A Research Note is used to capture early product or technology research in the R&D department. Some of the early research might make it into a new product, but sometimes the research is not useful for a product at that point in time. This is a very flexible Business Flow. The revisions of a Research Note are used to capture the multiple experiments run to investigate a proposition or idea.

Visual notation flexibility is needed for the Business Flow model. In Figure 3.16, the use of people images or icons in task positions of the flow was useful for the customer communication. The rationale was that most of the work takes place in laboratories on various instruments that are often standalone. The scientist or engineer doing the early product research observes the measurements, does calculations, and records the experiment results in a report that is attached to the Research Note revision. In other cases, the visual notation of the BPMN or the UML Activity Model supports good stakeholder communication.

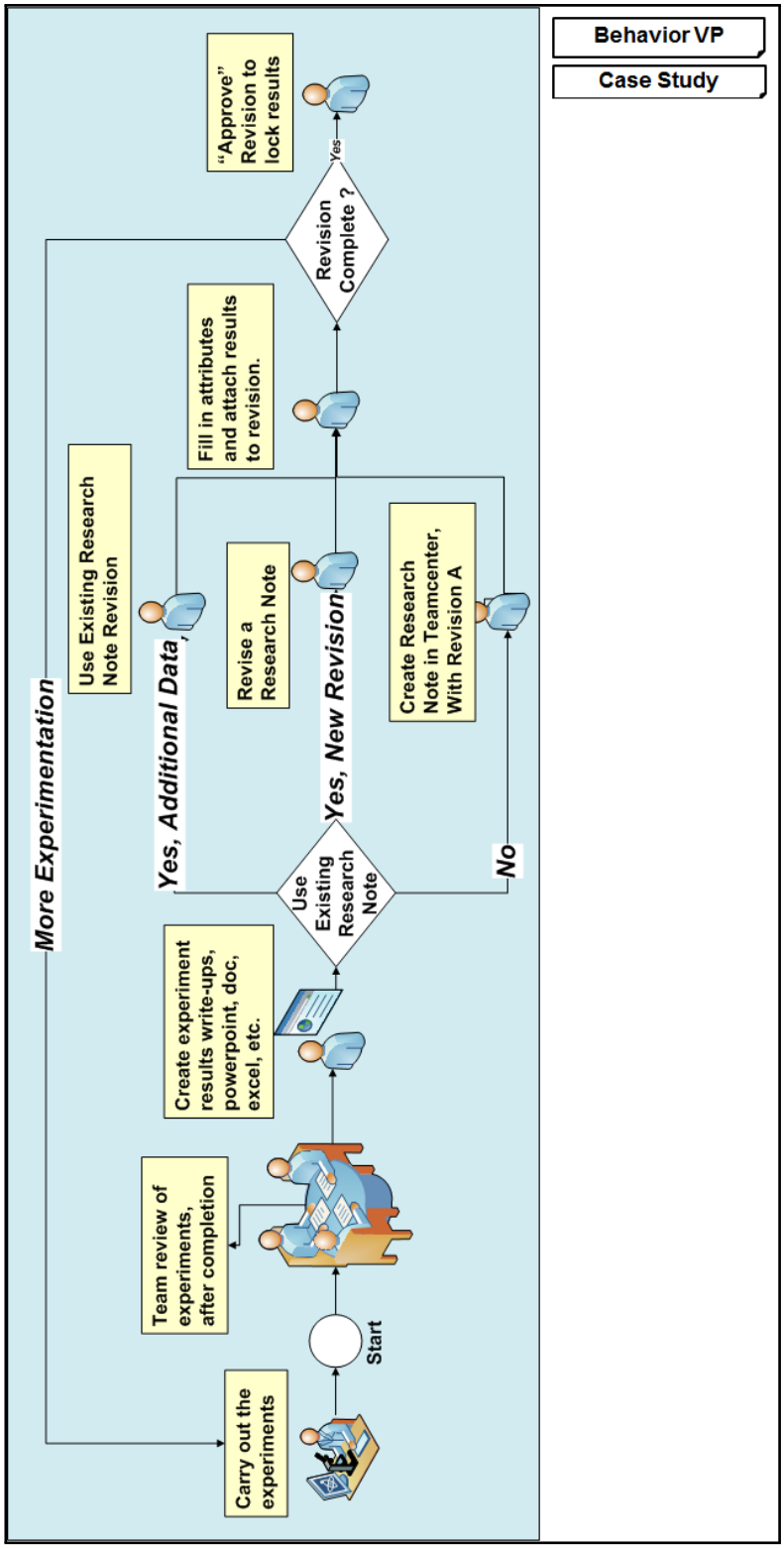


Figure 3.16: Business Flow Behavior Example.

The focus of the *Business Flow* in Figure 3.16 is on the ordering of the tasks to accomplish the business purpose. For each task that will be supported in the new solution, more requirements details are needed. For communication purposes and project iteration control, the additional task details are placed in a separate requirements artifact called a *Use Case* (see Figure 3.22 for an example). The separation of *Business Flow* and *Use Case* is an example of separating the higher level abstraction decisions of the *Business Flow* process definition from the lower level task details. With the separation, a connection must be established between the *Business Flow* task and the *Use Case*.

An example of the *Business Flow* and *Use Case* connection is shown graphically in Figure 3.17. The task of the *Business Flow* is represented by an icon of a person acting on the decision to create a new Validation project. The lower part of Figure 3.17 shows an area similar in appearance to the UML Activity Model Swim Lane and labeled *Validation Use Cases* and called the *Use Case Connection Area*. Within the *Use Case Connection Area* are rectangles with rounded corners which represent the individual *Use Case Artifacts*. Each *Use Case Artifact* graphical element has the name of the *Use Case* and in this example a brief description of the intent. The *Business Flow* task and the *Use Case* are graphically connected with a line, which in this example is a thicker red line. There can be tasks in a *Business Flow* that are not part of the automated solution and therefore will not be connected to a *Use Case*. Often the unconnected tasks are manual and there is not any business intent to automate them at the time of the project.

Figure 3.18 shows an example of Approval Flow participants from a CS-1 requirements specification. The participants are classified according to the responsibility Role they play in the Approval Flow, such as *Subject Matter Expert Reviewer* and their area of expertise as specified in the Group membership, such as *Regulatory*. During the Technology Viewpoint work people assignments to roles and groups will be made, but at the requirements level only roles and groups are needed.

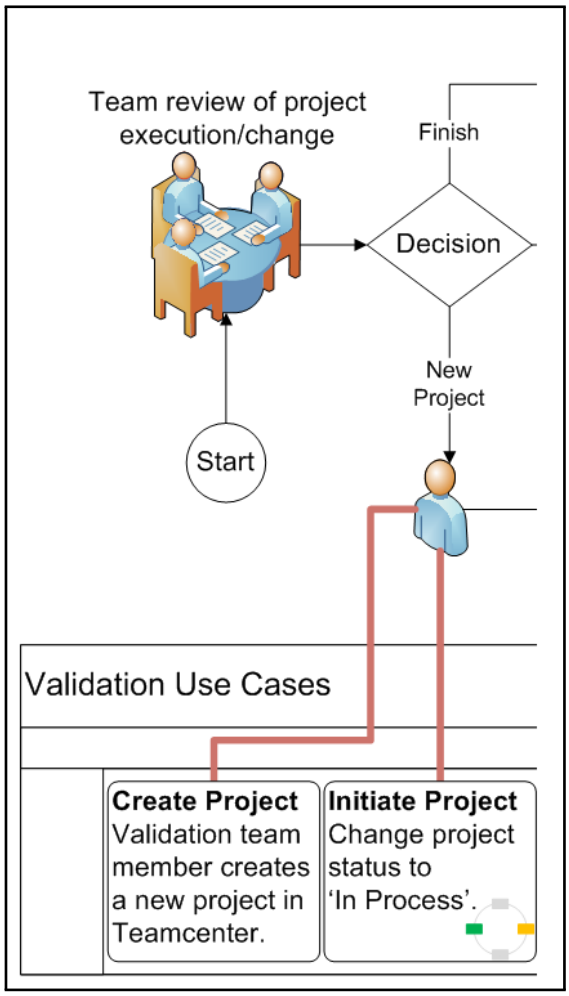


Figure 3.17: Business Flow Task Connection to Use Case Example.

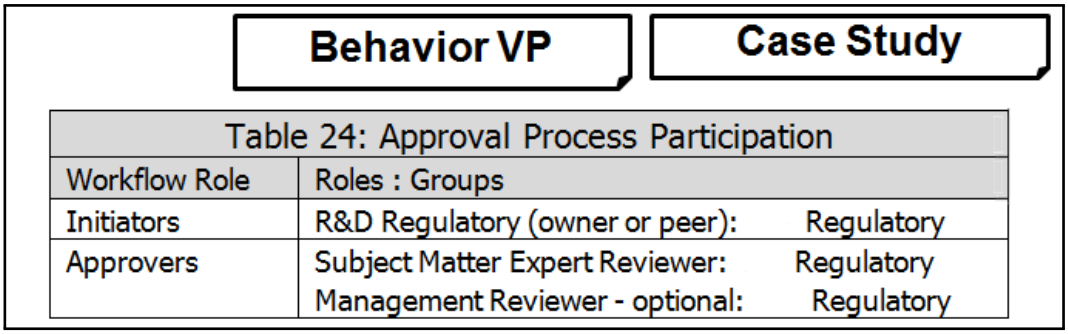


Figure 3.18: Approval Participants Table Example.

In Figure 3.19, an example of an *Approval Flow* is shown. There are four tasks in this *Approval Flow*. The first task checks that the information unit, in this case part of the regulatory filing structure, has the allowed status to start the approval. The second task is for one or more Subject Matter Experts (SMEs) to review the information unit for completeness and correctness. If the SMEs approve, then the information unit moves on, otherwise the necessary rework takes place. In the third task the management approvals happen. There are some cases when management approval is not necessary. If any manager rejects the information unit, then it cycles back for rework.

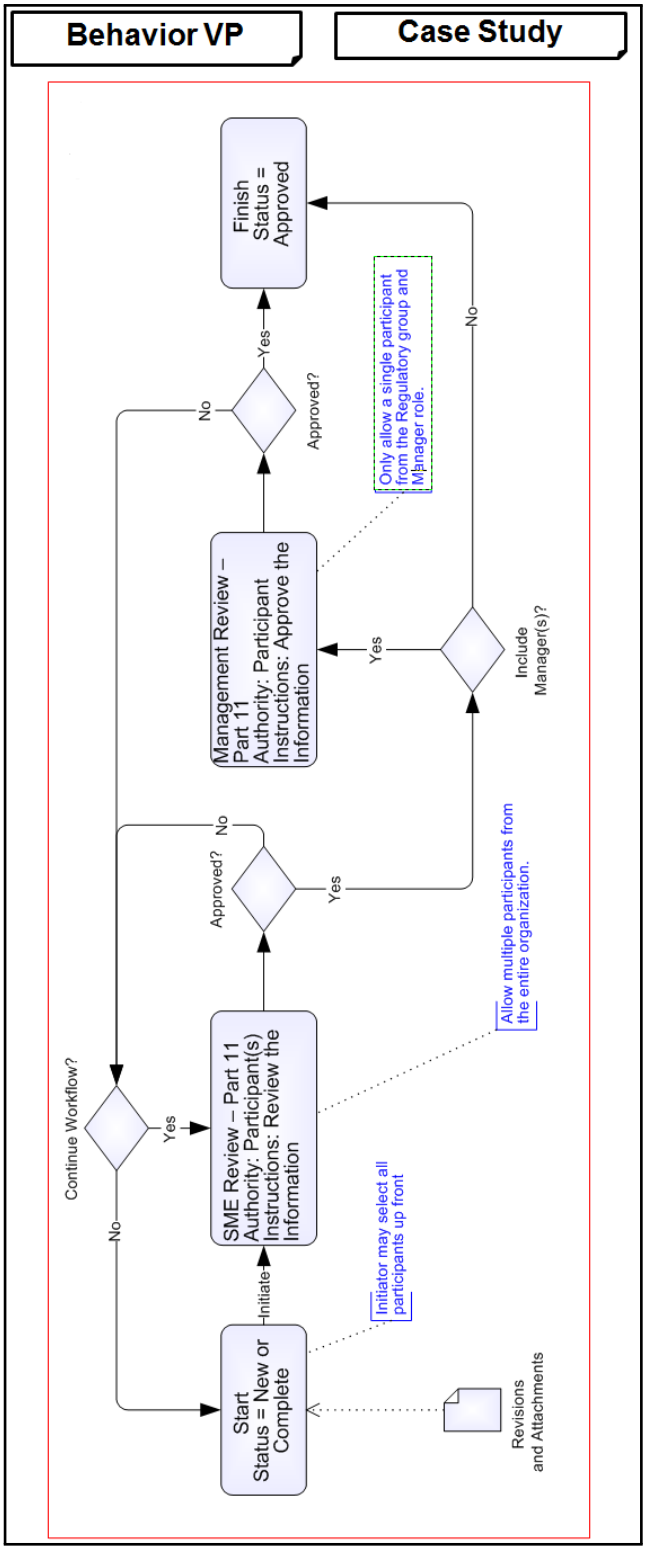


Figure 3.19: Approval Flow Example.

For each *Approval Flow* there is a list of potential SMEs by expertise and managers by authority to choose from. The list is kept in a business Standard Operating Procedure (SOP), which was in existence prior to the automated solution and is maintained externally to the solution. The SOP is used for other Approval Flows that are not automated. Without duplicating the SOP information, within the requirements specification a high-level table of participants is used, see Figure 3.18.

The part of the behavior modeling for the requirements is the Action Transformation Matrix (ATM). The ATM is the mechanism behind the transition shown by path ③ from Information Model Artifact to Behavior Model Artifact in Figure 3.3. In Figure 3.20, a full example is given from the regulatory solution. The left most column is the CRUD framework which organizes the *Extended Action* column. The Extended Action column acknowledges that there are variations to creating information and allows for the expression of the variations for stakeholder and developer communication.

The table entries are references to other places in the requirements specification where the details of the requirements and/or use cases are listed. See Figure 3.21 for an example. Inter-specification references were used extensively to enforce having the definition in only one location. The entry "N.A." means that action was considered but not applicable on this part of the Information Model. The entry "-" indicates that the cell does not apply or make sense for the combination of this part of the Information Model and the action. The OOTB entry means that the out-of-the-box (OOTB) capabilities were used and is assigned during the transition to design in the Engineering Viewpoint.

Behavior VP | Case Study

Table 20: Filing Object ATM Analysis Table

Action	Extended Action	Business Need	Who can access	When can access	Initial value	Value Range	Pre-Conditions	Action Results	Audit Action	Mgmt / Monitor Action	Display Format
Create	Item	RQ_REG_10 3.1	U_REG_10	U_REG_10	-	-	U_REG_10	U_REG_10 3.1	U_REG_10	U_REG_10	OOTB
	Item from template	N.A.	-	-	-	-	-	-	-	-	-
	Duplicate Item	RQ_REG_20	U_REG_20	U_REG_20	-	-	U_REG_20	U_REG_20	U_REG_20	U_REG_20	OOTB
	Structure	N.A.	-	-	-	-	-	-	-	-	-
	Structure from template	N.A.	-	-	-	-	-	-	-	-	-
	Duplicate Structure	RQ_REG_30	U_REG_30	U_REG_30	-	-	U_REG_30	U_REG_30	U_REG_30	U_REG_30	OOTB
	Reference (add)	RQ_REG_40	U_REG_40	U_REG_40	-	-	U_REG_40	U_REG_40	U_REG_40	U_REG_40	-
Retrieve	Values	RQ_REG_50 RQ_REG_890 RQ_REG_900	U_REG_50 U_REG_890 U_REG_900	U_REG_50 U_REG_890 U_REG_900	Property Dependent	Property Dependent	U_REG_5 U_REG_890 U_REG_9000	U_REG_50 U_REG_890 U_REG_900	U_REG_50 U_REG_890 U_REG_900	U_REG_50 U_REG_890 U_REG_900	Properties Tables
	Search / Find item	N.A.	-	-	-	-	-	-	-	-	-
	Search / Find revision	RQ_REG_60	U_REG_60	U_REG_60	Property Dependent	Property Dependent	U_REG_60	U_REG_60	U_REG_60	U_REG_60	Search Tables
	Where used in structure	RQ_REG_70	U_REG_70	U_REG_70	-	-	U_REG_70	U_REG_70	U_REG_70	U_REG_70	-
	Where references	RQ_REG_80	U_REG_80	U_REG_80	-	-	U_REG_80	U_REG_80	U_REG_80	U_REG_80	-
	Structure	N.A.	-	-	-	-	-	-	-	-	-
Update	Property Value	RQ_REG_90	U_REG_90	U_REG_90	-	-	U_REG_90	U_REG_90	U_REG_90	U_REG_90	-
	Structure - add child	RQ_REG_100	U_REG_100	U_REG_100	-	-	U_REG_100	U_REG_100	U_REG_100	U_REG_100	-
	Structure - re-org child	RQ_REG_110	U_REG_110	U_REG_110	-	-	U_REG_110	U_REG_110	U_REG_110	U_REG_110	-
	Reference	N.A.	-	-	-	-	-	-	-	-	-
	Status - Approve	RQ_REG_120	U_REG_120 Org. Model	U_REG_120	-	-	U_REG_120	U_REG_120	U_REG_120	U_REG_120	-
	Status - Obsolete	RQ_REG_130	U_REG_130	U_REG_130	-	-	U_REG_130	U_REG_130	U_REG_130	U_REG_130	-
	Status - Complete	RQ_REG_140	U_REG_140	U_REG_140	-	-	U_REG_140	U_REG_140	U_REG_140	U_REG_140	-
	Status - New	RQ_REG_145	U_REG_145	U_REG_145	-	-	U_REG_145	U_REG_145	U_REG_145	U_REG_145	-
	Revise Item (with copy)	RQ_REG_150	U_REG_150	U_REG_150	-	-	U_REG_150	U_REG_150	U_REG_150	U_REG_150	-
Delete	Item	RQ_REG_160	U_REG_160	U_REG_160	-	-	U_REG_160	U_REG_160	U_REG_160	U_REG_160	-
	Revision	RQ_REG_170	U_REG_170	U_REG_170	-	-	U_REG_170	U_REG_170	U_REG_170	U_REG_170	-
	Structure Membership	RQ_REG_180	U_REG_180	U_REG_180	-	-	U_REG_180	U_REG_180	U_REG_180	U_REG_180	-
	Whole Structure	RQ_REG_190	U_REG_190	U_REG_190	-	-	U_REG_190	U_REG_190	U_REG_190	U_REG_190	-
	Reference	RQ_REG_200	U_REG_200	U_REG_200	-	-	U_REG_200	U_REG_200	U_REG_200	U_REG_200	-

Figure 3.20: ATM for the Regulatory Solution Example.

Behavior VP | Case Study

4.3.1 Create Filing Item (RQ_REG_10)

Regulatory Filing items exist to hold all the information that pertains to a filing at the highest level. They serve two purposes.

1. Collect filing information in the properties meta-data.
2. Serve as a container for all the Section and Element Items in a submission.

The initial property values are set during the item creation process and are made required in the item configuration. Members of the R&D Regulatory Affairs group are the only system users who can create, update, or delete these items. Newly created Filing Items are automatically assigned to the KMS Regulatory Library (assuming the user's active project is Regulatory Library at the time of item creation). By default, all members of the Regulatory Affairs group, regardless of role, will be assigned as privileged members of the KMS Regulatory Library.

Figure 3.21: ATM Reference Specification Example.

Figure 3.22 shows an example of a *Use Case*. The purpose of the *Use Case* is to describe the internal business steps of a *Business Flow* task at a more specific level of business detail and to include business constraints. It can take several iteration review cycles to fill out this form. The work of filling it out is started during a *Business Flow* task review when it is decided that the new solution will support the task.

Behavior VP | Case Study

Form:	*U_REG_10*
Process Name	*Create Filing Item*
Object	*Filing Item*
Actors (Who can access)	*Any member of the R&D Regulatory Affairs group.*
Preconditions	*User must have appropriate system access.*
When Access	*Any time.*
Begins With	*A new filing needs to be initiated.*
Ends With	*A new Filing Item exists in the KMS with a unique identifying number.*
Definition/Steps	*1. User selects the OOTB function in Teamcenter for creating a new Filing Item.* *2. User assigns the new item number.* *3. User populates the rest of the required properties.* • *Name* • *Filing Type* *4. KMS creates the new item and automatically assigns it to the R&D Regulatory Library.*
Exceptions	*Exceptions due to infrastructure failure during the create process.* *User attempts to create a filing item without having the privilege to do so.*
Post Conditions	*A new Filing Item exists in the KMS and is automatically assigned to the R&D Regulatory Library.* *The new Filing Item exists in the owner's Newstuff folder or the active folder.*
Traceability/Audit	*Audit entries will be created according to the system audit policies defined in VS-0310(13).*
Constraints	*None*
Manage/Monitor	*None*
Notes	*None*

Figure 3.22: ATM Reference Specification Example.

3.5.4 Objectives of the Engineering Viewpoint

When design work is started in the RM-ODP Engineering Viewpoint, a transition, see paths ④ and ⑤in Figure 3.3 is made from the business requirements concepts, the language, and models of *What to build* and *Why to build*, into models of *How to build*. Engineering solution concepts are used in this viewpoint and should be modeled. Each requirement can often be fulfilled by multiple design alternatives. Also a design might fulfill multiple requirements.

The design activities produce the model artifacts in the Engineering Viewpoint. The models in this viewpoint are compromises of two major forces: 1) the business needs defined in the requirements and 2) the realities of the existing or proposed environment. The existing environment includes, among other things, the current infrastructure (software and hardware), project budget limits (money and time), organizational change resistance, and technical feasibility. Fulfilling some requirements might entail making changes to the existing environment, such as upgrading server machines, adding networking capability, or updating applications. The move to the Engineering Viewpoint will often expand the set of stakeholders, such as adding the Information Technology (IT) department people or infrastructure vendors.

The design work involved at this level begins by selecting a sub-set of the requirements models. The models come from a business information unit or a task in a Business Flow. The next step, an iteration, is to evaluate the engineering approaches available and build alternatives that can be objectively evaluated. The selection and evaluation work can be done in an incremental (select a requirements model, then model design alternatives) and iterative (evaluate the design alternatives against the requirements and the environmental constraints) process. A result of the select, design, and evaluate process is the natural creation of traceability from requirements to design and a record of the design decisions.

The transition to the Engineering Viewpoint consists, in general, of the following steps and produces additional information or project artifacts:

1. Select a requirements model.

2. Use existing or create new engineering design options and patterns to build alternative design models.

3. Record the design decisions used in developing each alternative design model.

4. Link the design model to the originating requirement model.

5. Evaluate how well each design model alternative fulfills the originating requirement and whether other requirements are also (partially) fulfilled.

6. Rank the design alternatives on ability to fulfill requirements and fit within constraints.

7. Through invention and innovation improve the top design alternative by using other design patterns, combining the strengths of multiple design alternatives, removing, or reducing constraints.

8. Take the final engineering design into the Technology Viewpoint for the implementation activities of technology selection, detailed technology design, test creation, and data creation.

It is not necessary to take all requirements models and move them in lockstep to Engineering Viewpoint model artifacts. The most critical models can be selected first. The selection actions are how agility is embodied in the transition to the Engineering Viewpoint. In a traditional SCRUM-managed project, the customer selects the user stories to be implemented. In the SAMEM, the customer selects or has input on which requirement models are designed and implemented first.

3.5.4.1 Engineering Viewpoint Design Work Model Artifact Examples

In the Engineering Viewpoint, the architectural design of the solution is specified. Each requirement has a corresponding design specification and each design specification has a unique identifier for traceability. The following examples relate to fulfilling FDA regulation 21 CFR §820.30, which realizes a Design Input Traceability Matrix (DITM) to map User Requirements to Design Inputs, Design Inputs to Design Outputs, and Design Outputs to Design Verifications and Validations with references to supporting evidence. Figure 3.23 is a quote (CS-1) from the overview section of the design specification with an important design decision rationale. Specifically, the design rationale relates the use of OOTB features to fulfill the requirements and the constraints impacting their use.

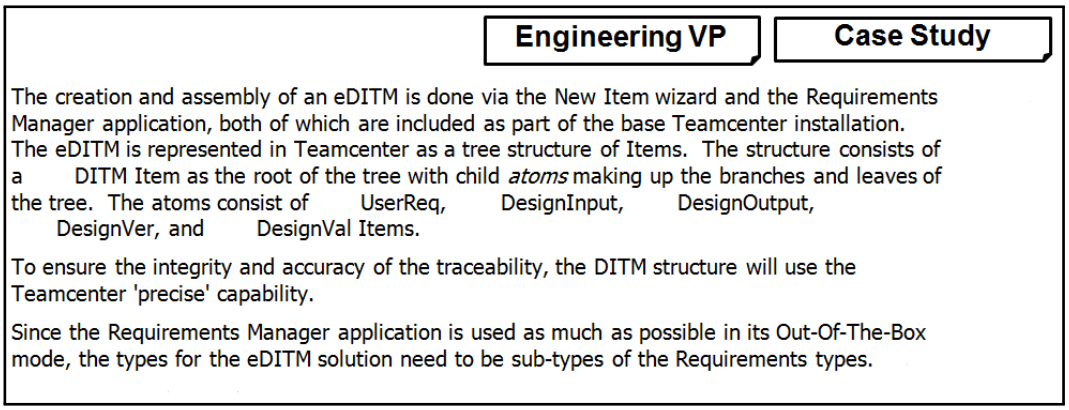


Figure 3.23: Design Rationale Example.

In Figure 3.24, the class model design in UML is shown with the generalization from the concrete solution classes to the Teamcenter OOTB starting point classes. The class model is input to the Teamcenter Domain Specific Language (TDSL) tool used to create

parts of the solution configuration. Classes with the stereotype «TCUA» are OOTB abstract classes from the TDSL tool.

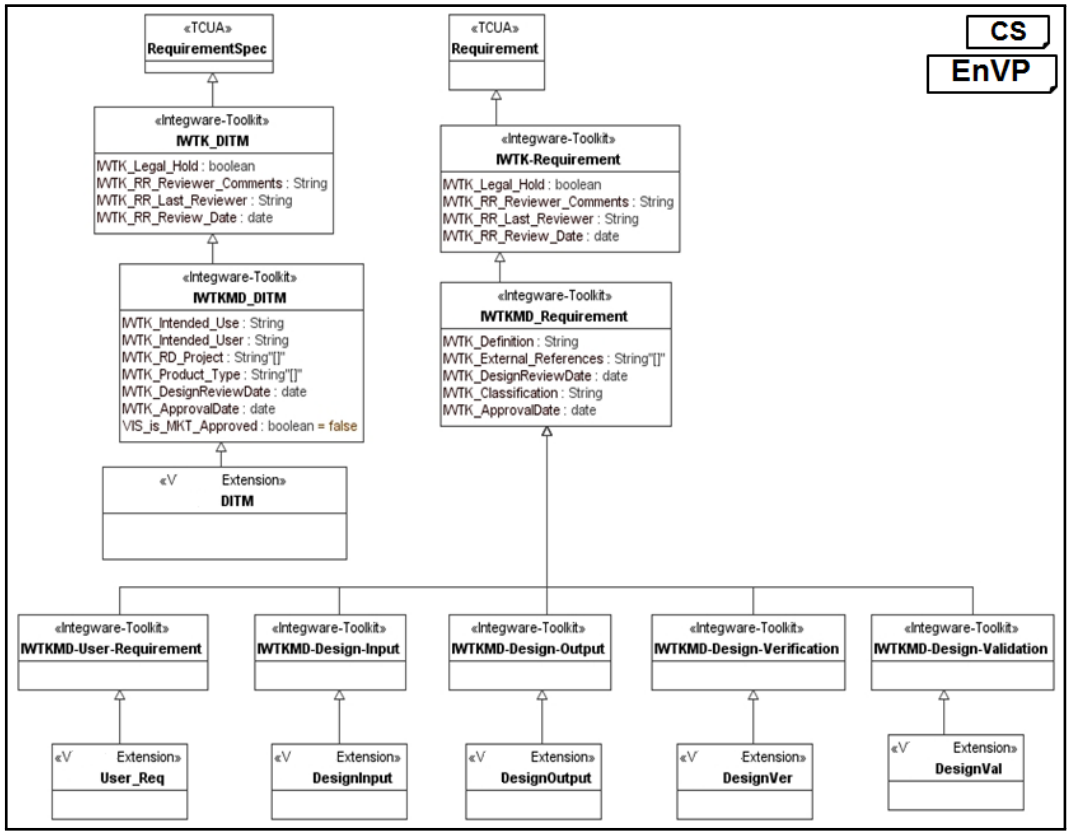


Figure 3.24: Design Class Model Example.

The stereotype «Toolkit» identifies the abstract classes that form part of the reusable toolkit definition and are artifacts from the engineering design activities. The «Toolkit» class names that begin with "TK" form the reusable base classes of the consulting toolkit. The «Toolkit» sub-classes with names beginning with "TKMD" for the base classes are for the medical device domain. It is a foreseen future expansion to have classes for the pharmaceutical domain to use a naming scheme of "TKPH". The concrete leaf classes for the customer specific solution have the «Customer-Extension» stereotype.

The following case study text snippet (CS-1), Figure 3.25, from the DITM design specification shows the design rationale for realizing the requirement *U_DITM_2*. Since an OOTB capability is the basis, some of the user interface behavior is listed. The rationale section also describes some of the preparation steps needed such as searching for a DITM to clone. This is an example of fulfilling the mantra of "File It, Find It, Reuse It AND Don't REDO It!".

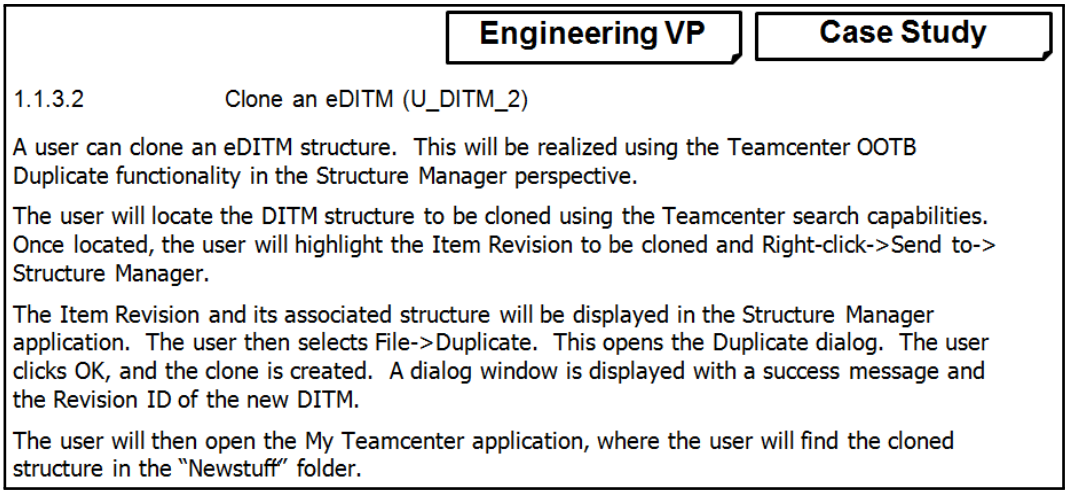


Figure 3.25: Design Rationale for Requirement Realization.

3.5.5 Objective of the Technology Viewpoint

The RM-ODP Technology Viewpoint contains the artifacts of the technology design work, the realization artifacts, such as configurations, code, and test cases. In contrast to the Engineering Viewpoint where, for example, a decision for a relational database is made to support a data persistence requirement, in the Technology Viewpoint the decision would be for a specific relational database product like Oracle™and a specific version of the product.

Using the technology decision for an Oracle™database as an example, the following design and implementation activities are done (the list is an illustrative example and not necessarily complete):

- Design the schema from the requirement information units.
- Develop the SQL statements to create the database schema.
- Develop the SQL statements to load data.
- Develop the database administrative task processes and scripts.
  - Installation.
  - Backup.
  - Standard data loading.
  - Performance benchmarks.
- Develop the database test cases.
- Develop any database behavior action, constraints, and triggers.

- Design and develop query optimizations.
- Design the database access and permissions structure.

The above list is an unordered mixture of technology design tasks and implementation tasks. These can easily be set as SCRUM user stories.

The kinds of models contained in the Technology Viewpoint expand from the requirements models and engineering design models. There are UML Class models that show the class names and attributes along with their public or private features. To continue the benefits of graphical communication, screen captures from the configuration tools are used in the technology specification. The screen captures of configuration settings are analogous to code listings.

From a specification perspective, the technology design record should enable a person versed in the technologies to recreate the realization in the case that the source was lost. The business project requirement to have a design specification of this detail is often a typical contractual obligation from the customer.

3.5.5.1 Technology Viewpoint Model Artifact Examples

The examples below are from the case studies. They show different kinds of technology implementation work. The examples come from a specific application and are configurations in the Siemens Teamcenter Business Modeler Integrated Development Environment (BMIDE). The BMIDE is the TDSL tool. Other configurations are done through tools embedded in the running Teamcenter application, for work such as User, Role, & Group definitions, workflow definitions, query (search) definitions, report definitions, and access manager definitions. The intent in these examples is not to educate or evaluate the Teamcenter application, but to provide examples of visual documentation of detailed design and implementation artifacts.

The following technology design graphical examples come from the Research Note solution, which is used to manage early product research experiment data. These images are screen shots from the BMIDE or other OOTB configuration tools.

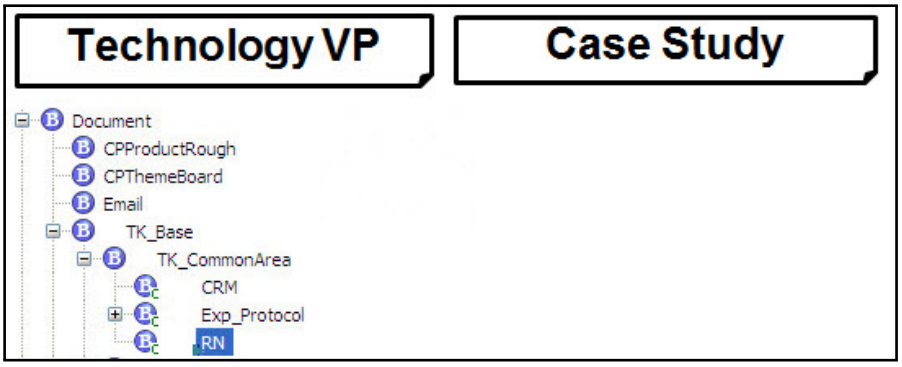


Figure 3.26: BMIDE Object Model Implementation Example.

Figure 3.26 shows the class inheritance hierarchy from the OOTB Document class, to the TK_Base class, to the TK_CommonArea class, to the solution specific classes of CRM (Chemistry Research Method), Exp_Protocol (Experiment Protocol), and RN (Research Note).

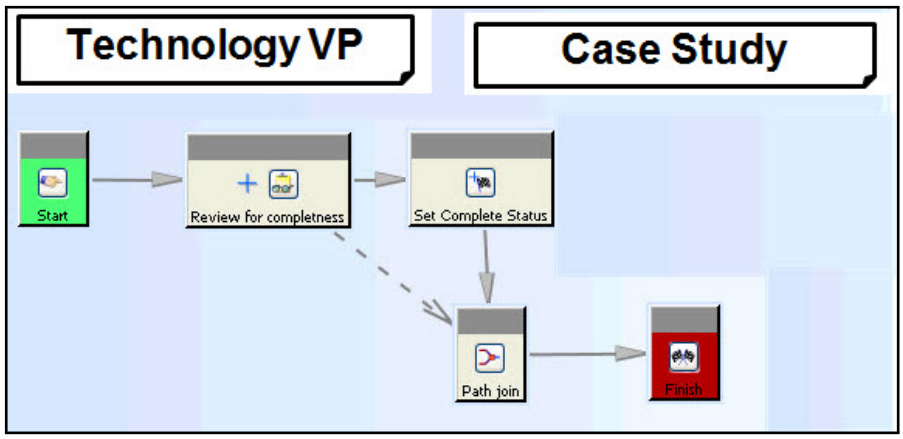


Figure 3.27: Detailed Design Approval Workflow Example.

The screenshot from the Workflow Configuration Editor in Figure 3.27 illustrates the implementation of one of the approval flows for the Research Note. The tasks "Start," "Finish," and "Path join" are administrative tasks. The approval is accomplished in task "Review for completeness" and the status attribute is updated to the "Complete" value in task "Set Complete Status."

In Figure 3.28, the configuration of a search action is shown. This screenshot comes from the OOTB tool used to define searches. The attributes of the Research Note Revision classes are shown in the order of importance as specified by the stakeholders. Screenshots from some of the configuration tools proved useful in getting partial user interface feedback before the complete prototype was available.

In Figure 3.29, part of the organization model hierarchy of expertise groups, V – RnD – Research can be seen. Listed under Research are the authority roles, Approver, Author, Leader, Manager, and Viewer.

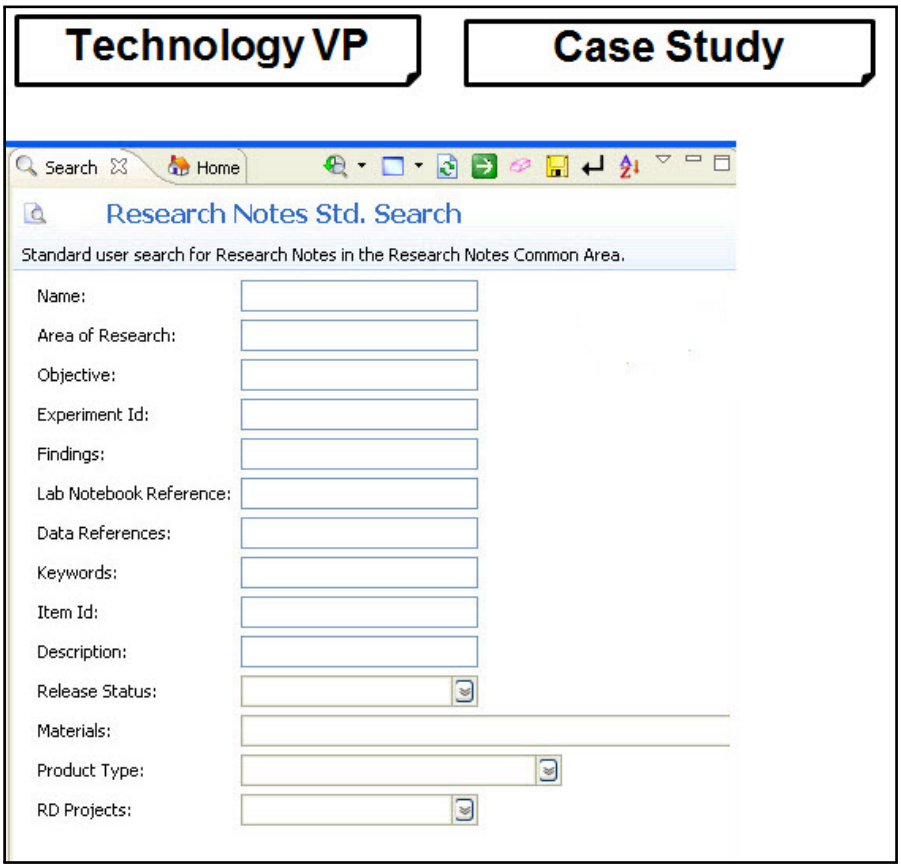


Figure 3.28: Detailed Design Standard Search Example.

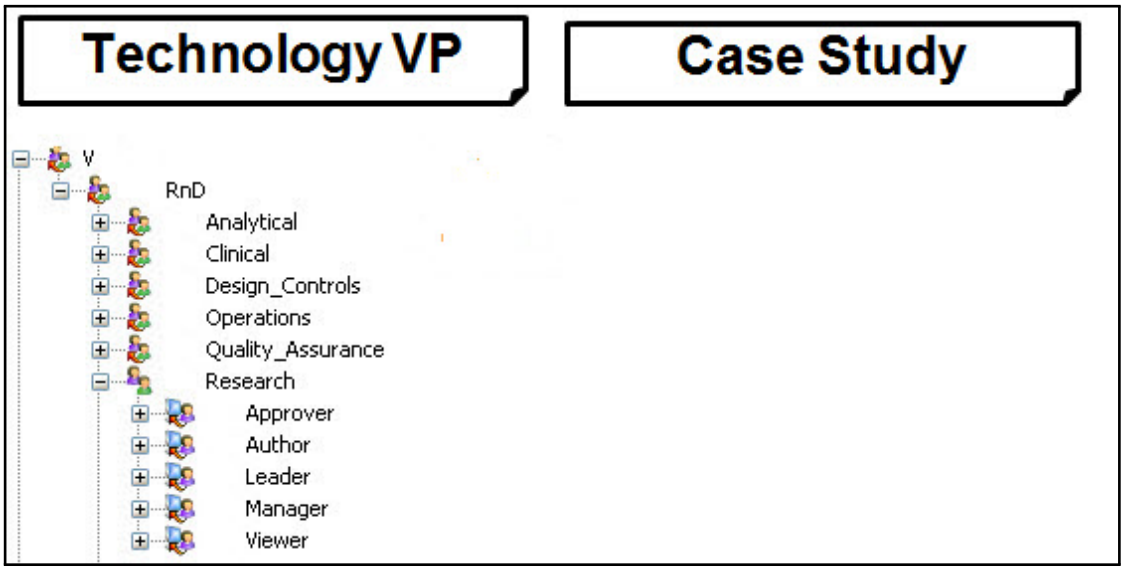


Figure 3.29: Detailed Design Organization Roles & Groups Example.

3.6 The SAMEM Model Artifacts Component

The SAMEM **Model Artifact** Component will be described in a distributed manner via examples, primarily from the case studies. The SAMEM does not prescribe any specific modeling approach, but rather emphasizes that the **Model Artifacts** should fit the domain. In the case studies, the UML was the starting source for the **Model Artifact** representations. The majority of the **Model Artifact** examples are found in Sub-chapters 3.5.1.1, 3.5.2.1, 3.5.3.1, 3.5.4.1, and 3.5.5.1. In addition modeling lessons learned are discussed in Sub-chapters 4.4.1.3, 4.4.1.5, and 4.4.1.6.

3.7 The SAMEM Tool Component

The SAMEM **Tool** Component does not prescribe any specific tool. This is deliberate. If the SAMEM emphasized any specific **Tool**, then domain flexibility innovation possibilities, and project effectiveness are compromised. For any particular project, the **Tool** choices should be determined by the domain, the **Tool** state-of-the-art, the communication needs, project **Model Artifact** transformations, and the experience of the people, developers and customers, with a specific set of **Tools**. Sub-chapter 4.4.1.7 describes the lessons learned in the area of computer tool flexibility. In addition, the communication lessons learned (Sub-chapter 4.4.1.3) have some tool choice rationale.

3.8 Non-functional Requirements Handling

The handling of the various non-functional requirements is accomplished in a non-traditional manner in the SAMEM. The following list of non-functional requirement areas is a compilation from several sources, each of which has a similar, but slightly different list [BPKR09], [FM15], [Gli10], [RR99]:

- Look & Feel
- Usability
- Maintainability & Portability
- Performance
- Reliability
- Operational
- Security
- Legal

Each of the non-functional areas listed above is addressed in detail below. In particular, how the SAMEM handles the specification through visual models and in the context of an iterative & incremental process.

3.8.1 Look & Feel

Specific *Look & Feel* requirements are not usually gathered. Rather, a goal of consistent user interface style across the solution is desired. The specific details of the *Look & Feel* style are often developed through the prototypes which are created and shown to the stakeholders. The feedback on the *Look & Feel* causes iterations on the first several prototypes. A balance between the various demands of the stakeholders is incrementally achieved.

The biggest area to balance is not between the various domain areas of the stakeholders but between novice and expert users. Expert users can deal with more data on the screen, longer lists of functions, and need little process guidance. The opposite is true for novice users. The *Look & Feel* non-functional requirement blends into the *Usability* non-functional requirement when the knowledge and experience of the user are considered.

There are two exceptions to the normal practice of developing the *Look & Feel* specifications. The first is when the product is used in a specific eco-system, such as Microsoft™, Apple™, or a product line where a user interface style guide exists. While the requirement to adhere to the style guide is a simple statement, it is possible that sketches of the user interface are appropriate visual requirement models to direct the prototype work. The second exception is in the domain of a product depending on an attractive user interface for success, such as a game or mobile application. From one standpoint, the two exceptions listed can be seen as constraints in the freedom allowed for the *Look & Feel* requirements.

3.8.2 Usability

Usability can have a broad definition with many aspects. Part of *Usability* is the navigation through the solution to accomplish the business task. How the data and information of the system is presented is also a part. The organization of the features and functions of the system both visually and logically affect *Usability*. Fitness of the features to the business tasks has a big impact on *Usability*. While there can be some psychological measures of *Usability*, it is mostly the opinions of the stakeholders that determine if the solution is reasonably usable or not.

When navigation through the user interface is critical to the success of a solution, such as a game or mobile application, visual navigation models are appropriate requirements artifacts. The navigation artifacts, such as Story Boards, Data/Information Display Mockups, or Functional Hierarchy Layout [BH98], [BC12], [IDF], can be used to plan and guide the prototype development.

The iterative & incremental process of the SAMEM allows for achieving good *Usability* through prototyping. Prototyping can take place on two major levels. At the Engineering Viewpoint level, paper-based prototypes such as story boards can be created to show the logical and functional flow through the user interface screens [BC12], [BH98]. Presentation tools such as Microsoft PowerPoint™can be used to create electronic versions of the model prototypes, which are easily shared via email or a central repository. In addition, paper sketches of the screen layouts can be done. The paper prototypes are

models that are used for the initial solution prototype development.

One or more of user interface paper prototypes can be selected for implementation prototyping in an iteration. How many paper prototypes are selected for implementation prototyping will depend on the time scope of the iteration. A key to effectively choosing a set of user interface models to prototype is to pick related models, such as following a hierarchy of actions down the chain. Note that the prototyping effort can also be limited to the visual components while the actual functionality is only stubbed out. By prototyping the visual components, hard coding simulated data and stubs for the functionality, the navigation and presentation styles can be evaluated while minimizing the effort invested. When the solution prototypes are developed, more concrete feedback on the good aspects and the weak aspects is received. The cost of the improvements can be estimated and the next iteration can include the selected improvements.

3.8.3 Maintainability & Portability

The SAMEM looks at *Maintainability & Portability* not as requirements, but as goals and constraints to be considered during the engineering design. As constraints these two aspects are viewed in the context of the known or envisioned evolution of the solution over time. Part of the context is the business evolution or growth and part is the infrastructure change. *Maintainability & Portability* are grouped together as they have very similar goals and often impose the same restrictions on design. In practice, porting the solution to a new platform is usually viewed as a type of maintenance since platform improvements are continuously occurring.

Understanding the *Maintainability & Portability* impacts is enabled through multiple design alternatives of solution architecture models of the software and hardware. The initial architecture model will be of the current infrastructure environment. Multiple future architecture models are created to evaluate different growth or cost options. The models of the future show both hardware infrastructure patterns and software architecture patterns. The differences between the initial solution architecture and a future solution architecture will indicate the components that are least likely to need adaptation, therefore must be maintained and those components most likely to need adaptation, and therefore ported. The **S2V** SEFP is applied in this situation to help define and separate the maintenance aspects from the portability aspects of the solution design.

The work of establishing the model of the hardware and software architecture components likely to remain the same and the components likely to change is work within the Engineering Viewpoint. The current architecture model and the alternative future architectures are part of the normal design alternatives that should be created.

3.8.4 Performance

The SAMEM does not have any specific models or techniques for the gathering of *Performance* requirements. There are several reasons for the apparent lack of *Performance* requirement specification.

The first reason is the application of the SEFP of *Optimum Performance* (**OP**), see Sub-chapter 3.2.8. The **OP** principle states that: *optimum performance is achieved by manipulating the fewest pieces of data the fewest times.* A corollary to the **OP** SEFP is that grouping or structuring of related data can minimize the data units. This principle is applied in all work throughout the whole project. This is shown in the following illustrative list of examples, which is not a complete list:

- Have the fewest functional requirements specified.
- Have the fewest data structures defined with the fewest attributes.
- Have the shortest, fewest tasks, business processes defined.
- Have the simplest user interface possible.
- Choose the most economical algorithms (balance between speed and space).
- Align the algorithms to minimize data conversions.
- Choose the most efficient compilers.
- Minimize the network traffic.
- Minimize the number of storage accesses and volume of data in each access.
- Use the fastest hardware.

Often when *Performance* is stated, it is interpreted as the system performance of the solution [BPKR09], [RR99]. For the SAMEM, *Performance* means the effectiveness of the user in accomplishing the business task. In a general sense, the user can be another application interacting with the new solution. Effectiveness is a combination of how fast the user can accomplish the task and the quality of that work. If the user must reenter data or redo actions, then the effectiveness suffers. Effectiveness can be enhanced by having exactly the features needed, having them arranged in a logical order for the business task, and presented in an unambiguous manner, therefore the design of the Human Computer Interface (HCI) has a big impact on effectiveness.

The grouping or structuring of the data can impact the solution performance in several ways. One impact on the effectiveness is the presentation of the data or information. Improper grouping or structuring can result in the user needing additional time to locate the data and interpret it (see Sub-chapter 7.3.1). The internal operations of the solution can be slowed by poor grouping causing such effects as multiple network transfers or extra database queries to support a user action.

The system performance supporting the user performance is ensured by good design and proper selection of the architecture components. It must also be realized that the hardware is constantly improving in performance with higher clock rates and larger capacities. *Performance* possibilities are constantly improving, within the limits of physics, without any solution design changes.

The *Performance* of cyber-physical systems is another matter. The *Performance* specification in that domain is a case of ensuring correct behavior often for safety reasons. The SAMEM has not been applied in the domain of cyber-physical systems and would need enhancement (see Sub-chapter 8.2.2 for discussion).

3.8.5 Reliability

Reliability is not viewed by the SAMEM as a requirement category. It is a business goal that could appear in the Enterprise Viewpoint, such as 24/7 availability. The next appearance of reliability is in the engineering and technology designs. The SAMEM views a goal at the Enterprise Viewpoint level as creating design constraints at the Engineering Viewpoint and Technology Viewpoint levels.

If very high reliability is a goal, then the architecture patterns and the components would be selected based on that goal. For example, if a database is chosen for persistence within a constraint of 24/7 availability, then a database technology that supports disk mirroring and automatic failover is used to achieve that high reliability goal.

3.8.6 Operational

Operational concerns of the solution are viewed by the SAMEM similar to *Reliability* concerns, which are enterprise business goals, such as 24/7 availability, and design constraints at the engineering and technology levels. The primary stakeholders concerned with *Operational* needs are the solution administrators in the Information Technology (IT) department. Some of the administrative concerns are listed below:

- Effort to install the solution for the users.
- Effort to load initial data or migrate existing data.
- Effort to install the behind the scenes supporting infrastructure, like servers.
- Effort to test the solution and any applied updates.
- Effort and frequency to update to a newer version.
- Effort to support the daily workload of users, such as answering questions, installing new users, loading data, etc.
- Time and resources needed for fault tolerance, such as backups.
- Need to learn new technologies.
- Impact on existing solutions and infrastructure.
- Training and certification efforts for new users.
- Cost for new hardware for users or infrastructure.

For some of the *Operational* concerns listed above, business goals can be set in terms of money, employee time, and number of employees needed, which translate into budgeting calculations. Depending on the engineering design alternatives expressed in the Engineering Viewpoint, cost/benefits trade-offs can be made visible to the stakeholders.

The cost of other *Operational* concerns can only be determined after engineering and technology design options are created. For example, training efforts can only be known after the user interface has been defined. However, the training effort goal might be one day for an existing domain expert employee. An example of a design constraint *Operational* concern is that the solution must work with an existing Web-based user interface in a Linux-based infrastructure.

In general, the *Operational* business goals are often best expressed in a text-based fact statement. At regular intervals during the engineering and technology design tasks, the *Operational* goals are used to evaluate the design options. In fact, the *Operational* goals are often one of the key selection criteria for choosing one design option over another.

3.8.7 Security

The content of this sub-chapter largely comes from the paper "Building Security Requirement Patterns for Increased Effectiveness Early in the Development Process" [MRRH05] and can be found there in detail.

The handling of Security concerns and requirements within the SAMEM primarily happen within the Information and Computational (Behavior) Viewpoints. Within some domains there can be business level goals for security which would appear in the Enterprise Viewpoint. The approach of the SAMEM is to use security patterns as described in earlier work [MRRH05].

Security is too general of a term to be effective in specifying solution goals and elicitation of requirements. While specific domains and government situations might have additional categories, the following list of security concerns covers most commercial software situations:

- Identification & authentication
- Authorization and access control
- Data integrity
- Confidentiality or data privacy
- Auditing
- Data authenticity
- Survivability
- Non-repudiation

The above terms provide a more precise set of words for the expression of requirements and designs. The *Security* concerns are not independent of each other. There are structural relationships, such as *depends* or *usedby*, and behavior relationships, such as *conflicts*. The relationships can be modeled and then used during requirements elicitation and design work to check completeness. The Security concern models become patterns that assist in the evolution of the requirements models to engineering models and design models.

The *Security* concerns listed above are refined and through the refinement process new concepts emerge that were hidden by abstraction. An example from [MRRH05] is presented to illustrate the emergence of concepts. In Figure 3.30, the refinement of the *Identification & Authentication* concern is shown. The Session concept emerges as a framework in which *Identification & Authentication* is used.

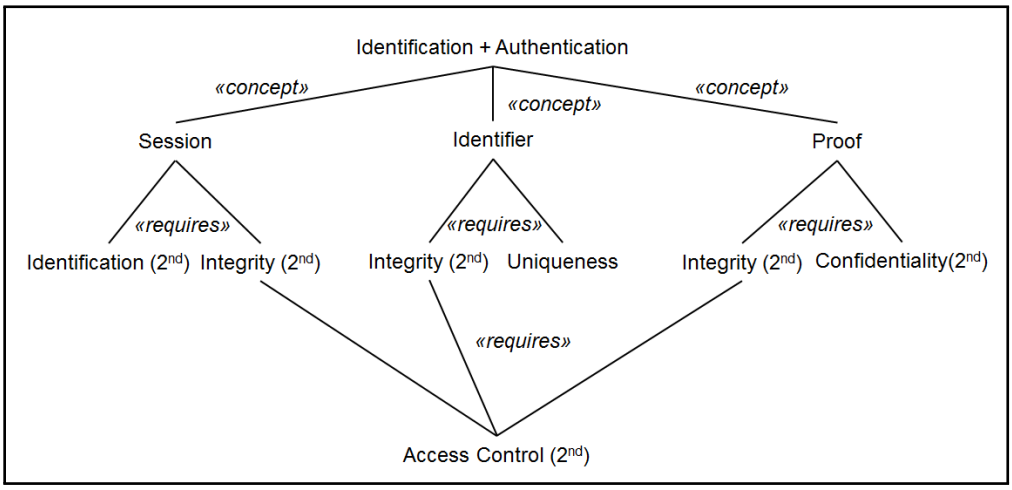


Figure 3.30: Emerging Security Concepts Example.

The *Session* concept in Figure 3.30 requires two supporting or secondary concepts of *Identification* and *Integrity*, the identification and the integrity of the session. The *Integrity* concept requires the *Access Control* concept for its completeness.

Just as there are emergent security concepts, there are emergent domain concepts that appear through refinement. The domain concepts will appear in the requirements and should be used consistently across the solution. In Figure 3.31 and Figure 3.32 two examples of emerging domain concepts are shown.

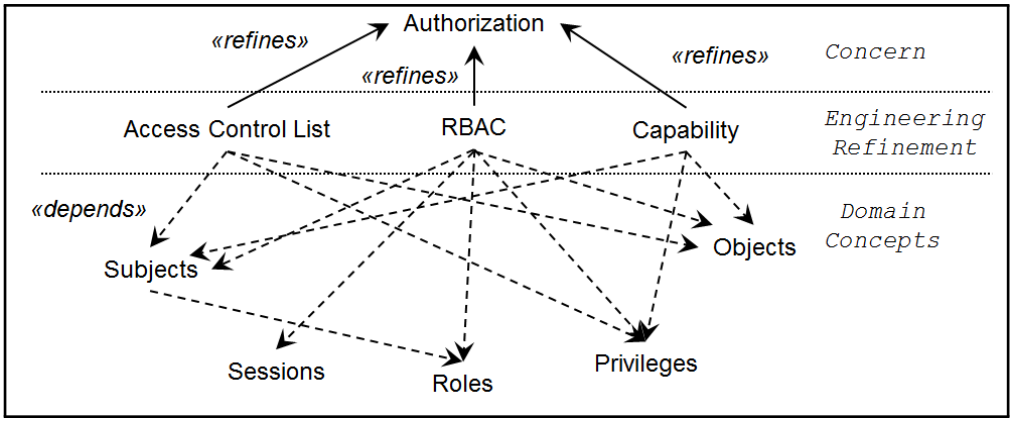


Figure 3.31: Emerging Domain Concepts from Authorization.

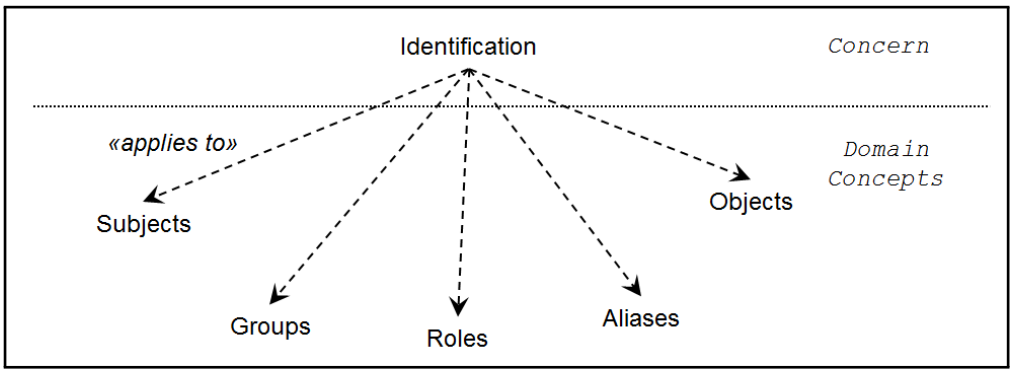


Figure 3.32: Emerging Domain Concepts from Identification.

Figure 3.31 illustrates how the *Security* concern of *Authentication* has several possible engineering refinements. The engineering refinement depends on certain domain concepts. For example, *Access Control List* depends on *Subjects*, *Privileges*, and *Objects*.

The refinements in Figure 3.32 show the «*applies to*» relationship. This is another type of refinement that produces domain concepts. The *Identification* security pattern produces five generic domain concepts. Within a specific industry or problem area, there can be additional domain concepts.

Specific instances of the domain concepts will appear in the requirement, engineering, and technology models. The security patterns are helpful as checklists to insure that important concepts for the solution are not missed.

The SAMEM handles the *Security* concerns of Authorization or Access Control, Data Integrity, and Confidentiality through the "who can access," "when can access," and "value range" columns in the ATM behavior artifact. The generic domain concept of *Object* appears in the information models of the Information Viewpoint as the information unit. Subjects, Groups, and Roles concepts are seen in the organization model.

3.8.8 Legal

The handling of *Legal* requirements is difficult because of the variety and the variations in specificity. The variety is generated by the multiple levels of governmental entities creating the laws and regulations around the world. The variety is further complicated by the revisions due to changing political priorities. The variations in specificity arise from the different domains controlled by the laws and regulations. For example, in the life sciences domain the results of experiments, used to establish the evidence that the product fulfills the claims, are expected to satisfy the regulatory standard of "Good Laboratory Practice" (GLP), however, the expectations for GLP keep improving as biological science understanding, laboratory techniques, and new equipment become the norm. The GLP "standard" evolves over time and current best practices are applied against the current submission for product approval.

The common factor throughout all the various laws and regulations is compliance. The compliance to the legal requirements must be demonstrated to the satisfaction of the regulatory bodies or legal entities enforcing the laws and regulations. Two primary mechanisms to achieve the demonstration of compliance are audit logs and design decision traceability. The SAMEM uses the Audit Action column of the ATM to record the requirements for compliance logging. The information in the audit logs will have data integrity and confidentiality requirements

Chapter 4

Applying the SAMEM

This chapter describes the experiences in applying the SAMEM with concrete project examples. As Frederick Brooks points out in his book the Design of Design [Bro10], there is a natural iteration of design modification as ideas are evaluated in real-world situations. The SAMEM is intended to be a pragmatic software engineering methodology and as such the experiences of applying it in multiple real-world projects generate useful modifications. While the definition of the SAMEM, as described in Chapter 3, is in large part driven by the goals described in Sub-chapter 2.4.1, there are definitional aspects that are the result of lessons learned in multiple industry projects. While the lessons learned come from these two specific situations, they are expressed in a more general form to be applicable to other domains. The over 35 years of industry experience of the author in multiple solution domains is the basis for the generalization.

As the old aphorism goes: "The difference between theory and practice is much greater in practice than in theory." Application of the theory of the SAMEM in real industry projects results in guidance of how to use the SAMEM and achieve success.

The projects stem from a Software Consulting Company (SCC), which specialized in developing solutions for medical device manufacturers. Two clients provide the Case Study 1 (CS-1) and Case Study 2 (CS-2) experience data. The solution domain is the management of product definition information and product design processes as required by regulatory agencies in order to achieve approval to sell the product.

Sub-chapter 4.1 contains more details about the client companies involved in the case studies. Specifically, the number of solutions, the size of the project teams, estimates of the size of the solution, and the time length of the engagement. Certain case study proprietary information is generalized because of confidentiality restrictions but remains true to the situation.

A critical project condition for adoption of the SAMEM is the education of the project people, both developers and stakeholders, in the daily workings of the SAMEM. Sub-chapter 4.2 describes the experiences in educating the team members involved in the two case studies in the SAMEM ideas and topics relevant to the business of the case study companies. The experiences in the education of the project team members highlight the requirements in transitioning from the academic definition of the SAMEM to a working understanding. The description of the application and education of the SAMEM flows from the Enterprise Viewpoint through the Technology Viewpoint.

The SAMEM operates in a real-world environment and as such it must adapt to various project conditions. Sub-chapter 4.3 discusses the application and adaption of

the SAMEM in more detail. The SAMEM is designed to work in an iterative & incremental project process environment as described in Figure 1.1 and in Sub-chapter 3.3. Although the SAMEM is intended to work in various project process environments, a project process environment designed to support iteration is preferable. The main reason for preferring an iteration approach is the real-world phenomenon of constant discovery and increasing understanding as described by Brooks [Bro10]. The project process effectively used in the case studies is described in Sub-chapter 4.3.1 and provides a basis for evolution.

Lessons learned and practical adjustments are discussed in Sub-chapter 4.4. Some of the lessons learned were mentioned in Chapter 3. These have been collected, organized, and expanded in Sub-chapter 4.4.1. The lessons learned in this research are also compared to the lessons learned from other research in Sub-chapter 4.4.2. Some of the lessons learned over the course of the case study projects have been applied to the design of the SAMEM, while others are subjects for future work. In adapting the SAMEM to other software domains, the lessons learned can provide guidance in adjusting the methodology within the constraints of the real world.

4.1 Case Study Company Descriptions

The SAMEM was used for two different medical device companies, called Case Study 1 (CS-1) and Case Study 2 (CS-2). The case study companies develop, manufacture, and sell two different sets of products. Some details of the companies and the products cannot be stated because of confidentiality agreements.

All of the medical products must receive authorization from the appropriate Ministry of Health (MOH) before the products can be sold. In the United States of America the MOH is the Food and Drug Administration (FDA). The primary regulations controlling the projects developed for CS-1 and CS-2 are contained in The Code of Federal Regulations Title 21 – Food and Drug (21 CFR).

The class of the medical device has impact on which FDA regulations are applied. FDA classifies medical devices based on the risks associated with the device. Devices are classified into one of three categories - Class I, Class II, and Class III.

Class I devices are deemed to be low risk and are therefore subject to the least regulatory controls. For example, dental floss is a Class I device.

Class II devices are higher risk devices than Class I and require greater regulatory controls to provide reasonable assurance of the device's safety and effectiveness. For example, condoms or contact lenses are classified as Class II devices.

Class III devices are generally the highest risk devices and are therefore subject to the highest level of regulatory control. Class III devices must typically be approved by the FDA before they are marketed. For example, replacement heart valves are considered Class III devices.

For the company in CS-1, the products are used on the surface of the body and are intended to correct certain physical limitations; as such they can be easily removed if there are health or comfort issues for a particular patient, Class II. The products of

company CS-2 are intended to be surgically implanted in the body to repair defective body parts, Class III.

A key impact on the application of the SAMEM for the case study companies is the need to adhere to government regulations. The specific impact is the need for the solution development methodology to generate not only the solution artifacts, but also generate artifacts about the development process details such as the decisions made, the people involved, the changes to decisions, and when the decisions were made. The combination of solution artifacts and process artifacts requires additional formality to the application of the SAMEM versus other software domains.

As described in Sub-chapter 3.5.5.1, the solutions are based on the Siemens Teamcenter Product Lifecycle Management commercial application (Teamcenter) as the Technology Viewpoint design decision. The Teamcenter product provides a strong base of functionality and data structures on which to build the above applications. A CS-1 and CS-2 business solution goal was to limit the adaptation of Teamcenter to mechanisms that are impacted to a minimum extent by product upgrades. Teamcenter has the features to configure an instance within a defined range of parameters called Configuration.

Occasionally, it was necessary to use the extension abilities of Teamcenter to create unique behavior for specific business rules. Teamcenter can be extended through Customization by linking in complied code containing the new functionality. The code extension is accomplished through the use of Teamcenter libraries in C++ that were extended, compiled, and linked via the Microsoft C++ development environment. The unique behavior was needed most often in the workflows for approval processes. It was never necessary for either the CS-1 or the CS-2 solution to extend for data modeling or user interface needs.

The Code columns of Table 4.2 and Table 4.5 hold the lines of code used in customizations while the other columns hold size estimates for the configuration actions.

4.1.1 CS-1 Company and Project Descriptions

CS-1 was the first company where the SAMEM, in an early version, was used for modeling requirements and as framework for the project plan. The CS-1 company produces a device that falls into the FDA classification of Class II (Special Controls). The scope of the full solution contract consisted of a total of 13 sub-solutions that were developed and released for production use over the course of approximately four years. The breakdown of the full solution into the 13 sub-solutions was part of joint strategy work that was done prior to and independently of the application of the SAMEM to the sub-solution development. In Table 4.1 the solutions are listed in the order in which they were developed. As each sub-solution was developed, new lessons were learned and applied to improving the SAMEM for the next sub-solution project.

Table 4.1: CS-1 Sub-Solutions.

0	Main	This sub-solution, which grew over time, gathered common or shared requirements, designs, and implementations that applied to more than one of the other sub-solutions. The other sub-solutions made references to the specifications listed here.
1	Chemical Research Methods (CRM)	The chemical research methods are developed in R&D for use in production to test for contamination and leaching effects. These are mostly for product safety and purity.
2	Toxicology	The toxicology sub-solution manages the results of several types of toxicology tests of the product. The toxicology tests are concerned with product safety.
3	Design History File (DHF)	The design history file is a collection of documentation about the design process and is mandated by FDA regulation 21CFR §820.30. This sub-solution area provides a place for electronic scans of historical paper DHF documents.
4	Clinical Studies	The clinical studies sub-solution manages the various clinical studies done to evaluate the effectiveness of the product in meeting its medical goals.
5	Sterilization	There are always improvement ideas for more cost-effective sterilization methods for the product. This sub-solution manages those trials and results.
6	Stability	The products of CS-1 are based on multiple silicone hydrogel materials which degrade over time. This sub-solution area manages the tests for material stability.
7	Validation	All product production processes must be validated according to FDA regulations. This sub-solution area manages the validation protocols and reports for inclusion in the regulatory filings.
8	Physical Materials (MTM)	This sub-solution manages the various material tests for aspects such as strength, flexibility, and tearing resistance.
9	Research Notes	The research notes sub-solution provides a place to define experiments and collect results on fundamental scientific investigations which may or may not lead to product improvements.

Order	Sub-Solution Name	Description
10	Electronic Design Input Traceability Matrix (eDITM)	The eDITM is a full computer-based support for the development of the DHF as required by FDA regulation 21 CFR §820.30.
11	Technical Documents	There are many technical reports needed for a regulatory filing that do not fall into another category. This area provides a place for both new documents and the electronic scans of historical paper documents.
12	Equipment Engineering	The equipment engineering sub-solution holds all the designs and information associated with the development of production equipment and product assembly lines.
13	Regulatory	In this sub-solution all the correspondence and the submission artifacts sent to a regulatory agency for product approval are managed.

The approximate sizes of the different sub-solutions of Table 4.1 are shown in Table 4.2. The classes are the ones defined for the solution through inheritance refinement of the OOTB classes of Teamcenter. The attribute number is the count of new attributes added for the solution in addition to the OOTB supplied attributes. The workflows are approval type workflows related to the state machine progression of the sub-solution classes. The Other column covers a variety of configuration actions needed to fulfill requirements and ensure a correctly functioning solution, such as user interface appearance, access control settings, state machine definitions, icons, localization, list-of-value definitions for attributes, and integration of Microsoft Office™. Some of the work results in Code-based customizations and Other-based configuration which produces components that are reused among the sub-solutions. All of the solution size measures require design and development effort in the Engineering and Technology Viewpoints.

The important business drivers for CS-1 were to move from a paper-based product documentation environment to a more efficient electronic-based environment. Work was being repeated at high cost because reports were lost, incomplete, or untrustworthy. For example, the cost to redo an unreliable toxicology report for a product material would be approximately $200,000 and take about six months to complete. For the CS-1 company, the added time cost to the medical device development project was the bigger issue as the repeated work was slowing product releases. Since the various domain experts were redoing work and not working on new products, the company felt that its market leadership position was being eroded. Good financial practices placed limits on the R&D budgets for personnel and equipment, so that people could not be arbitrarily thrown at the issues. The domain experts usually needed to have a Ph.D., M.D., or both degrees and many years of experience, so that means there is a small pool of people to potentially

Table 4.2: CS-1 Solution Size Measures.

Sub-Solution	**Classes**	**Attributes**	**Workflows**	**Group & Role**	**Code (lines)**	**Other**
Main	12	19	5	11	1420	70
Chemical Research Methods (CRM)	2	19	5	9	0	25
Toxicology	2	26	1	8	320	16
Design History File (DHF)	22	12	2	5	600	89
Clinical Studies	16	23	13	23	500	82
Sterilization	2	32	1	4	0	22
Stability	7	19	8	5	0	51
Validation	8	24	4	6	0	51
Physical Materials (MTM)	2	19	2	0	0	7
Research Notes	2	12	1	6	0	18
Electronic Design Input Traceability Matrix (eDITM)	30	15	4	2	1500	90
Technical Documents	2	27	5	3	300	33
Equipment Engineering	6	11	2	5	150	34
Regulatory	8	21	4	5	150	80

hire.

The work with the CS-1 stakeholders was primarily with groups in the R&D department. As can be seen from the solution areas in Table 4.1, all are of significance to product development activities. There was consistent personnel involvement on the customer side for all 13 sub-solutions in the project management/director roles, which helped keep the SAMEM and project process education activities to a minimum. However, for each sub-solution a new set of domain experts were added to the team for

the duration of that specific sub-solution development. The new domain experts were involved from the requirements elicitation work through the final sub-solution acceptance testing and needed education in the SAMEM approach. The consistent customer management personnel across all the sub-solutions aided in the education of the new sub-solution experts.

Table 4.3 lists the number of people primarily involved in the CS-1 solution over the entire project, which lasted four years (2008 – 2011). The number of subject matter experts listed is of those that were involved to a significant extent. There were more subject matter experts whose involvement was only for a quick question or two. Adding all the people up and multiplying by the total project time produces an expenditure of **268** person-years of effort, which qualifies CS-1 as a large project according to this thesis.

Table 4.3: CS-1 Participants, Roles & Numbers.

Team Group	Number of Members
Consulting Company Developers	10
Customer Management / Directors	11
Customer Subject Matter Experts	46

4.1.2 CS-2 Company and Project Descriptions

The CS-2 company produces Class III products. The project was under the management of the vice president in charge of process improvement. The three driving forces for CS-2 were the replacement of an unreliable and outdated electronic solution, the need to have more product development information in an electronic format to improve sharing, and improvements to R&D effectiveness. The existing system was unreliable in several aspects: the software version was no longer supported, it ran on hardware that was no longer produced or supported, and the software configuration did not support current business practices. Superficially there are many similarities to the solutions for CS-1 because of compliance to the same FDA regulations. The product and company process differences generated solution differences.

For the project with CS-2, the solution work was divided into areas called *workstreams*, which are similar to the CS-1 sub-solutions in intent. Each company has its own vocabulary. The workstreams were spread across a five-workstream, four-year project plan. About six months into the project, the Document Management and the Literature Management System (LMS) Integration Workstreams were combined because the Document Management workstream is the source for the contents of the LMS. Also, about the same time (six months into the project) by using the same perspective as the UML generalization relationship, the Common & Shared Configurations and Code workstream were created. Half-way through the project (year two), the workstream of Data Migration was added. The Data Migration need was identified at the start, but planning for it was delayed until the new information models were defined and some parts of the

workstreams were deployed.

The merging workstreams and discovery of new project needs deserving to be elevated to a workstream level of importance is an example of the kind of discovery or dynamic project behavior that Brooks [Bro10] talks about.

Table 4.4: CS-2 Workstreams.

Order	Workstream Name	Description
1	Engineering Change Control	This supports the mandate of FDA regulations on rigorous change control of product specifications as stated in 21 CFR §820.30(i).
2	DHF Management	Provides for the electronic management of the Design History File as mandated in 21 CFR §820.30.
3	Product Definition (eBOM)	Manages the authoritative as-designed definition of the product in the Engineering Bill of Materials (eBOM).
4	Document Management	Manages many other documents such as Standard Operating Procedures (SOP) that can be subject to compliance audits by a Ministry of Health.
	Literature Management System (LMS) Integration	The LMS system is an in-house web-based access to PDF versions of documents that are under the control of the PLM system. Some of the information is available on company external web pages for customers.
5	Data Migration	The migration of vetted data from the existing legacy solution to the new solution, primarily the Document Management and LMS systems.
6	Common & Shared Configurations and Code	Contains common or shared configurations and code across the other workstreams, where identical definitions and behavior are necessary.

The solution sizes for the different workstreams are shown in Table 4.5. The column definitions are the same as for Table 4.2. A major difference between CS-1 and CS-2 was the goal of not minimizing the upgrade maintenance of Teamcenter with CS-2. CS-2 put a higher priority on business special rules and application behavior for their specific needs, which resulted in more code development. Much of the code development was done by the subcontractors, while the main development team did the configuration.

The Data Migration work is a process of Extract, Transform, and Load (ETL). The

Table 4.5: CS-2 Workstream Size Measures.

Sub-Solution	**Classes**	**Attributes**	**Workflows**	**Group & Role**	**Code (lines)**	**Other**
Engineering Change Control	4	48	14 (8)	3	1000	23
DHF Management	4	11	4	4	1600	24
Product Definition (eBOM)	22	87	4	8	1200	8
Document Management & LMS	88	29	42	1	1200	32
Data Migration	0	0	0	uses all groups & roles	5000	0
Common & Shared	0	3	5	20	7000	37

data in the old system is extracted to a neutral format, in this case XML. The old data XML is transformed into XML that aligns with the new information or data model. The new or target data model is the combination of the definitions of all the classes and attributes from the other workstreams. The target data model will have some of the same classes and attributes, new classes and attributes, and there are deprecated classes and attributes. For the new attributes, appropriate values must be generated that are consistent with the original data. The transformed data is checked and verified for correctness, sometimes by automatic means and sometimes by manual visual inspection. At an appropriate time in the project, the verified transformed data is loaded into the new solution instance. The normal case is that the verified transformed data is loaded in small batches, which form logical information units and the loading is coordinated with other system deployment or installation activities. The coding effort for the Data Migration workstream is in the transformation tasks.

Table 4.5 lists the number of people primarily involved in the CS-2 solution over the entire project, which lasted four and a half years (2011 – 2015). The number of subject matter experts listed is of those that were involved to a significant extent. There were more subject matter experts whose involvement was only for a quick question or two, but these people are not counted as they were only needed for a few specific business

details. Adding all the people up and multiplying by the total project time produces an expenditure of **279** person-years of effort, which qualifies CS-2 as a large project according to this thesis. While the management personnel of CS-2 remained stable, there was a higher turnover of subject matter experts. The subject matter expert turnover required repeating the SAMEM education sessions.

Table 4.6: CS-2 Participants, Roles & Numbers.

Team Group	Number of Members
Consulting Company Developers	15
Subcontractor Developers (India)	10
Customer Management / Directors	12
Customer Subject Matter Experts	25

4.2 The SAMEM Introduction in the Consulting Company

The introduction of a new methodology, process, or tool faces the inertia of organization change. This is different for every organization. The development and introduction of the SAMEM was influenced by the opportunities of a new project situation, the developer's knowledge, and practical business constraints. The situation within the consulting company at the start of the project required the education of the new team members in an MDE approach, the medical device domain, agile process thinking, the Teamcenter product selected by the customer, and general software engineering principles. These real-world constraints impacted the SAMEM definition through lessons learned.

Using the SAMEM is a change, therefore the doubts and fears with new processes and approaches need to be explained through adequate rationale to ensure acceptance. In [FM15] the realities of people working in teams on engineering projects are addressed and the impacts of methodology change on the people. The same realities existed in the consulting company for the new methodology introduction. While every project situation has its unique aspects, there are common themes that occur. There are always new people being added to the team. The new people need to be educated in topics such as the domain of the project, the thought tools in operation, the process approaches used, the technology choices with rationale, and project operating procedures. Their education and acceptance of prior team decisions are critical behaviors necessary for the new people to be reliable and productive members of the team. These education topics for the CS-1 and CS-2 projects are discussed in the following sub-chapters along with impacts on the SAMEM.

The education specifics for the new team members and the customers are presented in a document format and in a slide presentation format. The two formats were kept in sync and updated with lessons learned at the end of a significant project milestone. The following list is a summary of the document format table of contents with explanations:

- Introduction – presenting the high level business rationale for the methodology.

- An Architect's Foundation – describing concepts and working ideas a solution architect needs to keep constantly in mind.
- Design First Principles – an architectural design approach.
- A short bibliography of reference books for the development team.
- PLM Definition for Solution Innovation – sets the approach for using PLM technologies in a realistic and innovative manner.
- Software Engineering First Principles – detailed list of SEFPs and expanded in this thesis.
- Action Transformation Matrix (ATM) – introduction and explanation.
- UML Use and Primer – a short introduction to the elements of the UML that will be used and how they will be used. Not a complete UML introduction.
- Methodologies and Processes – introduction and summary of RM-ODP and the Volere methodology.
- Putting It All Together – an overview of how the previous material will work within the confines of a project control process with examples from earlier projects as illustrations.

4.2.1 Medical Device Domain Education

Only one developer, the technical writer, and the author of this thesis had experience with the medical device domain at the beginning of the CS-1 project. The education of the other team members was accomplished using existing materials within the company and a list of links to web sites previously vetted. The major domain education area was in the Food and Drug Administration (FDA) regulations (21 CFR) and some of the internationally accepted regulations (ISO 13485) for medical devices. Only a base-level of knowledge, mostly terminology, was established before the project was started, so that the development team members could communicate at a basic level with the stakeholders. While the regulation terminology is common, the application within specific medical device companies varies. People picked up more customer-specific information, both product-specific vocabulary and company internal-jargon as needed. A glossary of customer-specific terms was provided from the previously executed strategy project and extended during the course of the solution project.

There were two important reasons for this education investment. First, proper terminology was needed for effective customer communication, which is a major goal (***HL-GOAL-1***) of the SAMEM. Second, the solution was subject to audit by the FDA, therefore it had to be compliant with the regulations in order to be legally used. The regulatory education had impact on the SAMEM through the creation of artifacts and process documentation required to pass an FDA audit. More specifically, the solution

development needed to have traceability from the requirements to design to implementation. A validation plan with the listed test cases and a report of the validation activities are needed by the FDA as part of the approval submission package.

The regulatory compliance requirements had impacts on the SAMEM specifics, which might not be necessary for other solution domains. For example, the SAMEM produces many different types of documentation for purposes such as stakeholder communication, distributed team communication (over time and space), and regulatory compliance audits. The SCRUM and Agile approaches advocate the minimization of documentation, because documentation is viewed as accidental complexity. The addition of the *Audit* column to the ATM is an example of an extension to the SAMEM for meeting regulation compliance.

For this thesis, the business domain education is for the medical device manufacturers. Some of the specific lessons learned can transfer to other business domains, however, independent of the specific business domain for applying the SAMEM there will always be the need for domain education.

4.2.2 UML and Modeling Education

The intent for the SAMEM is to use UML-based models within an agile framework. The author of this thesis had extensive UML experience through using UML in multiple industry projects; work on Object Management Group (OMG) standards, and teaching a graduate-level UML modeling course. A few of the people on the development team had exposure to the UML, usually through a single semester undergraduate course. Most members of the development team had no UML experience.

A one-day introduction to UML and modeling for the team was developed and it was used in the rest of the company. The UML introduction material was combined with other education material into a document format for reference and a presentation format for teaching sessions. The introduction focused on class models, activity models, and state machine models in order to support the SAMEM. The UML course focused on the definition of the modeling components as given by the UML standard [OMG15b] and included examples from the author's previous use of UML in various industry projects. The introduction was brief, quickly supplemented by actual application with the customer and guided by the author. For reference purposes, the book *UML Distilled, Third Edition* by Martin Fowler was recommended, as it is short and inexpensive. The presentation and review of the models with the customer was initially restricted to the experienced people in the development team, so that explanations of the models would be clearer and consistent.

The tool selected for internal use by the development team was MagicDraw UML™. This tool was used for the prototype designs involving the Class Model, Activity Model, and State Machine Model features. No code generation was done with this tool, as the Teamcenter product had its own set of tools for configuration and extension through code.

4.2.2.1 Process Modeling Techniques

The Business Flow plays a critical role in specifying the behavior requirements of the solution. There are several pragmatic techniques that increase the effectiveness in developing the Business Flow. There is a five-tiered hierarchy of decisions that are made in the process of specifying a Business Flow. The main idea of the five-tiered hierarchy developed from the author's experience, which consists of over 25 years of business process development in industry, covering over 100 different processes and multiple domains. The second block of experience leading to the five-tiered approach comes from the author's team interaction experiences in developing workflow standards in the Work-Flow Management Coalition (WfMC) and the Object Management Group (OMG). By generally following the hierarchy, the risk of major re-specification is minimized. The steps to define the Business Flow are:

1. Establish the process goal.
2. Define the process task graph.
3. Specify the information needed by each task.
4. List the actors needed by each task based on skills, knowledge, and data access.
5. Determine the process administration and auditing data requirements.

A *Goal* for the process must first be defined in order to verify that the development is correct. The *Goal* should be stated in domain business terms. The **CI** and **Essential** SEFPs are useful in establishing the *Goal*, which must be consistent with the solution concept.

The second tier of decisions involves defining the graph of tasks in the Business Flow. The tasks are defined with the assistance of the **Modules** SEFP. Each task should accomplish one specific action. The task names are best formulated as a verb-noun pair of words. The words should be drawn from the business domain and company idiom vocabulary. As the tasks are reviewed, the task names will often evolve to become more descriptive and meaningful for the stakeholders.

The tasks are ordered so that the purpose of the Business Flow is accomplished. The CRUD action pattern can be helpful in guiding the task ordering. For example, before a task can update an information unit, it must be created. Each Business Flow graph will have its own unique structure and appearance. To ensure clear communication, the structure of the graph should evolve naturally. In some cases the Business Flow graph will be vertically oriented and in other cases horizontally. The number of parallel tasks and the spans of the rework arcs will influence the structure, until an acceptable image is agreed upon by all stakeholders.

As the tasks are defined, an estimate of the information or data needs will start to develop. The clean specification of the information needs of each task comprises the second tier of decisions. The estimated information is reviewed in the context of the stable task definition and refined. There are two levels of Business Flow process-relevant

information to specify. The first level is the information needed to accomplish the task which consists of input information, additional processing information, and output information. The input information is the minimum set of data needed to start the task work. The output information is the data produced by the task. The additional processing information is any data that can help in accomplishing the task. Any or all of the information can come from or be written to persistent storage. For any particular task, any or all of the input, additional, or output information sets can be empty, meaning no information is needed to accomplish the task. Events or signals are considered a type of information.

The third level of information to be considered is the Business Flow administrative information. The administrative information is not necessarily used by any task but is generated by the enactment of a process instance. The administrative information is often used for auditing the Business Flow for improvements or compliance. Some examples of administrative information are task start timestamp, task end timestamp, the actual user from a set of potential users doing the task, and if the task is part of a cycle, the number of the iteration through the cycle. In the case of a review task in a cycle, a business condition could be that a new reviewer is required for each new iteration, which means that the previous reviewers are additional processing data for the task.

The fourth tier of decisions covers the specification of the users that are able to accomplish the task. The users or set of users are specified either through a skill type or a business responsible level. Specific people are never named. The skill type will align with the work to be accomplished in the task and will indicate knowledge areas or work abilities. Examples of skill type include machinist, chemist, Java programmer, or board certified cardiologist. The business responsibility level indicates the business authority needed, such as manager, director, or company officer. The specification of the users, which can be another computer application, is the result of considering the skills or responsibilities along with the information access needed to accomplish the task work. In some cases additional information access must be granted to the user to accomplish the task.

The last step in specifying the Business Flow is to list the administrative and auditing data to be collected during the enactment of the process. Compliance traceability needs will drive part of these specifications. The other major driving factor is the desire for continuous process improvement by the business. Constraints that must be considered when collecting process audit data are the privacy laws and worker council agreements in place.

4.2.2.2 Iterative & Incremental Development Process Education

There was a small amount of experience with the SCRUM and Agile development processes [Amb02] and eXtreme programming [Bec00]. Some general education took place in this area with the focus on the iterative & incremental development approach. Many of the development team members only had experience in the more traditional waterfall process method but had heard about Agile techniques. In developing the SAMEM, spe-

cific techniques and terminology from SCRUM or eXtreme programming were explicitly avoided, but some general ideas were kept in mind. The consulting company had established some development process techniques in earlier projects, but they were centered on the traditional sequential, phase-based approach. Some of the earlier techniques were adapted to work in the SAMEM, most importantly the Conference Room Pilot (CRP) idea (see Sub-chapter 4.3.4).

The "iterative & incremental" terminology is explicitly used in order to allow a natural discovery of what worked across the whole project lifecycle and to develop a non-computer vocabulary habit for customer communication. A process blank slate is desired to allow all the development team members the chance to contribute to answering the question: What are the natural incremental steps and iteration sizes for the work during requirements specification, design specification, and realization?

4.2.2.3 COTS Product Education and Constraints

While a lot of general computer and programming knowledge can be reused in a new customer engagement, there are almost always new technologies to be learned. The use of the new technologies must also be adapted to the project processes and methodologies. The impacts to the SAMEM are in the creating of Engineering and Technology Viewpoint models that communicate the design while being consistent with the target technology.

The Siemens Teamcenter Product Lifecycle Management™(Teamcenter) product was selected by both of the case study customers through an in-house process. The entire team attended official Teamcenter training. A couple of the developers and the author of this thesis went to an advanced training course on the Teamcenter Business Modeler Integrated Development Environment (BMIDE). The BMIDE can be seen as a Domain Specific Modeling Language (DSML) tool. The more in-depth knowledge was passed on to the other development team members that needed it.

The BMIDE is based on the Eclipse tool set and provides a configuration mechanism for the information model and many of the business behaviors. The configuration values from the BMIDE are stored in an XML format, then loaded into the Teamcenter repository for interpretation at run-time. There are other business behaviors, such as workflow definitions, access control, user definitions, and property values, that are configured via tools within Teamcenter during run-time and it is possible to store this information as external XML files, which can be loaded into another Teamcenter instance. The configuration set is very rich and six of the sub-solutions in CS-1 were realized without creating any traditional code.

The constraints on the manner in which the configuration mechanisms of Teamcenter work had an impact on the process and the approaches to modeling requirements and demo prototype development. Teamcenter and the BMIDE are object-oriented, therefore mapped directly onto the revised UML modeling artifacts. Without compromising the quality, the requirements can be formulated, so that the design expression in the Teamcenter configuration tools is straightforward. The requirements workflows mostly mapped directly onto the Teamcenter workflow capabilities, except when a special business rule was needed. The special business rules required code extensions that checked

both values of attributes and information structure integrity. The Teamcenter built-in object definitions have full CRUD features that map directly to the requirements. For almost all class extensions, the CRUD *Delete* operation was hidden for FDA compliance reasons. The unneeded *Delete* impacted the SAMEM by modifying the use of the CRUD pattern to a feature that changed the instance state to *Obsolete* rather than a *Delete*.

4.2.2.4 Software Principles Education

A tertiary consulting company objective was to educate and mentor the development team, so that they could make good choices in the future, for example when a developer moved into a position of greater responsibility or on to a different project. It was desired for them to be able to adapt the SAMEM when needed while staying consistent with the foundation principles. Project working principles were developed that could be applied beyond the scope of the SAMEM.

This was done through defining a set of Software Engineering First Principles (SEFP) (see Sub-chapter 3.2 for extended definition and rationale), then showing how they were used to make the decisions about modeling approaches and the new project process. The SEFPs are a project and process independent set of quality guidelines which can be applied in any software solution development situation. During the development process, the first principles were applied many times for selecting a particular task execution option. The SEFPs were applied from the definition of the solution concept through the implementation tasks. While the list of SEFPs will grow in the future with the application of the SAMEM to other domains, for the SAMEM development and the case study projects, the checklist of principles is:

- Conceptual Integrity (CI)
- Essential versus Accidental Complexity Awareness (Essential)
- Stability to Variability (S2V)
- Symmetry of Action (SoA)
- Modules as defined by Parnas (Modules)
- Coupling and Cohesion Metrics (C&C)
- Patterns (Patterns)
- Optimal Performance (OP)
- Change Language (CL)
- Ready-to-Hand (R2H)
- Form Follows Function (FFF)

To a large extent, the set of SEFPs are the result of the author's experience over 35 years of successful and unsuccessful software projects. Much of the most valuable experience comes from the interaction with other developers at the conclusion of a project, when an evaluation of the process was done. The idea of the SEFP approach arose from the author's education in Physics and the role first principles play in that domain, such as inertia and the speed of light as a limit. In the Software Engineering domain, the primary inspiring examples come from Frederick Brooks [Bro95], [Bro10], and Terry Winograd [Win96].

The education in the SEFPs consisted of defining and explaining them. During the course of the CS-1 and CS-2 projects, the author of this thesis as the SAMEM mentor guided the creation of a project habit to apply the SEFPs. The SEFP application habit became part of the standard developer review task for all activities, it was viewed as kind of regression test. At first the author of the thesis would ask which SEFPs should be considered or applied to the task at hand. The SEFP checklist, similar to a code style checklist, was applied during the SCRUM-like standup meetings and at iteration review meetings. An acceptable response is that a particular SEFP did not apply to this situation with a rationale of why not. For example, if the Pattern SEFP applies, then the follow-on questions are "Which pattern and why?". The first principles are mental tools to be used to help with the proper focus during the full solution development process. The application of these first principles will be shown in examples later.

4.2.2.5 The SAMEM Consulting Company Introduction Summary

It is difficult to introduce a new software engineering methodology without understanding the context in which it will be used. The context will impact the processes and artifacts of the methodology, as well as the approach in introducing the methodology. The SAMEM has the flexibility to adapt and it was important to involve the users of the SAMEM in some of the adaptations to ease the organizational change on the development team.

There are aspects of introducing a new methodology that took time to get correct. The initial assumptions by the author of this thesis and attempts to communicate the ideas were flawed. One mistake is that an expert in a subject area often assumes that the benefits of a technique or approach are obvious. This was the case with the benefits of modeling and the deeper knowledge and experience by the author of this thesis. On the development team, most of the people were unfamiliar with modeling and uncomfortable with using models for requirements. Their project history was of using text to express requirements. The stakeholders had many of the same issues as the development team; however, the development team had the obligation to explain the models and the value of models to the stakeholders.

Another aspect of methodology change that was difficult to overcome was in the area of project management. Most of the people on the development team and the stakeholders only had experience with a waterfall project process. For the CS-1 project, there was an assumption that the tasks in the project would be entered into Microsoft Project™(Project) and that this tool would be used to track progress. It is extra work (accidental complexity) entering each short iteration into Project and updat-

ing the task completion. The ability to represent iterations in Project is essentially nonexistent. This break with tradition was uncomfortable for the management people. The compromise reached was to assume that it would take three iterations to stabilize the requirements definition, although some of the times it took two and other times it took four or more iterations. For the CS-2 project, the SCRUM tool *Jira* (https://www.atlassian.com/software/jira) was used to manage the sprints.

Although there were obstacles to parts of the SAMEM, there was an attitude of willingness to try to improve the project approach. All of the development people had bad experiences in one project or another and recognized the issues that the SAMEM is attempting to improve. On the stakeholder side, most of the people are from the R&D organization and therefore had a background of trying new processes and a tolerance for trial and error with new technologies. The overall willingness to try was a factor in the success, along with developing new project habits.

There were fewer adaptations for the CS-2 group as a history of success from the CS-1 project existed. The success history was communicated to the CS-2 team through review of CS-1 artifacts and testimonials from some of the management people in the CS-1 company. The author of this thesis was the only person common to both companies and development teams, therefore the bulk of the education activities were done by him.

As employees had little time to learn subjects outside of direct relevance to the current project, a minimum number of references for the SAMEM were used. Many, but not all members of the development team pursued additional training in these areas on their own. The essential points from the references were summarized in a consulting company technical report and presentation written by the author of this thesis, which formed the basis for the education. That technical report was a major input to this thesis. The following references were used for specific purposes (often not the whole book or article, but a chapter or two):

- [Bro95] – Only the chapters related to conceptual integrity and essential versus accidental complexity.
- [Fow03] – The ideas of agile development and that modeling can be used.
- [Fow04] – A UML reference for self-study.
- [RM-98] – The whole specification was used for guidance and project organization.
- [Par72] – The whole article was used to move the technical members of the development team from thinking in code definitions of objects to modules.
- [TW95] – This provided the ideas of Value-Discipline Model for assistance in developing the solution concept and expressing the business value.
- [RR99] – The requirements gathering question examples and checklist were used to educate team members on how to elicit the requirements.
- [Win96] – This book was cited as containing examples of creative design work as an aspirational objective, but there were no explicit ideas used.

4.3 The Experiences in Applying the SAMEM

In developing the SAMEM, the key goals are to have small incremental steps from requirements elicitation through realization, so that it is possible to quickly verify that the project is on the correct track and show the customer steady progress towards the solution through small but accumulating deliverables. The value of small steps is a principle of SCRUM [Coh10]. The small steps are a key to minimizing risk to achieving a successful solution as evaluated by the stakeholders.

In the cases for CS-1 and CS-2, prior to starting the solution development project based on the early SAMEM ideas, a separate business strategy project was run with the customer to establish business goals, constraints, and objectives for the solution. The work that was done in the business strategy project can easily be considered as part of the Enterprise Viewpoint.

The work with a new customer begins with a business strategy assessment. The purpose of the assessment is to understand the customer's business goals, the current problems or issues, future product plans, the organization structure, the current information processing environment, and to recommend a high-level course of action to improve the customer's business. The strategy assessment is not part of this thesis but provides the initial input as a rough solution concept for the SAMEM.

The inputs to the SAMEM are the goals, business objectives, and project constraints discovered in the strategy assessment and expressed as a solution big picture. The SAMEM course of action decomposes the solution big picture into a set of sub-solutions. The sub-solutions are sized so that they can be delivered in approximately six calendar months. Each sub-solution delivers a unit of definite value to some part of the business. After the delivery of each sub-solution, the course of action is evaluated and adjustments to the order or scope are made to compensate for changing business conditions or to accommodate organizational changes. Experience has shown that a sub-solution should be realized in about a six-month timeframe in order to balance a unit with significant business value with keeping the course of action flexible enough for changing business conditions. The SAMEM must be flexible enough to operate within the business constraints of the course of action.

The domain of medical devices falls under the regulation of the Food and Drug Administration (FDA) in the U.S. Since the solutions being created managed the medical device product development, they are required to be validated to the FDA standards. This means more than just the "code" deliverable. Other required deliverables that the process needed to support include the requirements specification, the design specification, the realization components, the installation process, test cases, and traceability from requirements to design to realization to test cases. At any time, the FDA could audit these deliverables. The SAMEM needs to deliver all the pieces that could be audited.

There are three major components to applying the SAMEM. First is the use of the Open Distributed Processing – Reference Model (RM-ODP) [RM-98] to provide a mental and project framework guiding requirements elicitation and design activities. The

second major component is establishing primarily graphical requirements and design artifacts to facilitate improved communication [Mat11]. The third component is a three-phase project progression from the requirements elicitation activities to design prototype evaluation, and finishing with a nearly complete solution called the Conference Room Pilot (CRP), where the customer does a real-world evaluation. The three-phase project progression is a risk partitioning and mitigation approach.

The final step is the deployment of the sub-solution into production use. Because the deployment of the sub-solution must be validated according to FDA rules, the timing of the deployment must be coordinated with other business activities. In practice, the SAMEM process ends with the sub-solution ready to be deployed, but in the case studies the actual deployment task was timed to coordinate with other system maintenance activities for minimal business disruption.

4.3.1 The Controlling Project-level Process

The use of visual models is an effective way to partition the requirements and the design into reasonably sized increments to drive the iterations. However, management of the iterations is also required. Although the SAMEM intrinsically supports iterations, in practice the frequency and size of the iterations are under the control of a project-level management process. This sub-chapter describes the project-level process effectively used in the case studies.

In Figure 4.1, the controlling project-level management process is shown. This is an overview image used with the case study stakeholders and was supplemented by more detailed views when needed. The labeled boxes in the upper half show abstract images of the deliverables for communication and orientation purposes. The lower half has the high-level iteration cycle flows. The number of iteration cycles is not explicitly specified but depend on the convergence of the artifacts to a stable form.

The project-level process is split into three major phases of work, each of which has increasing levels of specification, verification, and confidence. The first phase of the process, titled *Paper Prototype*, corresponds to the Enterprise, Information, and Computational (Behavior) Viewpoint work of eliciting requirements. The *Demo Prototype* is the second phase which produces demonstration prototypes of some of the requirements and corresponds to the work done in the Engineering and Technology Viewpoints. The third phase is the *Conference Room Pilot* (CRP), which brings several demonstration prototypes together and is a near final acceptance test by the customer. Within each phase there would be multiple SCRUM-like iterations to deliver the relevant artifacts.

The design of the process is a compromise between the SCRUM idea of refactoring and the reality of finite project resources. Some rework takes place, as the development of the demo prototypes uncovers shortcomings in the requirements and the CRP uncovers shortcomings in the demo prototype implementations and requirements [Bro10]. However, the goal is to avoid rushing into development with major misunderstandings. To avoid having to throw away too much work, the project management process has verification tests with a sign-off by the customer that it is reasonable to proceed to the next phase. The controlling project-level management process is a constant balancing

act between the investment in time and money, the business benefits, and the risk of wasting the investment.

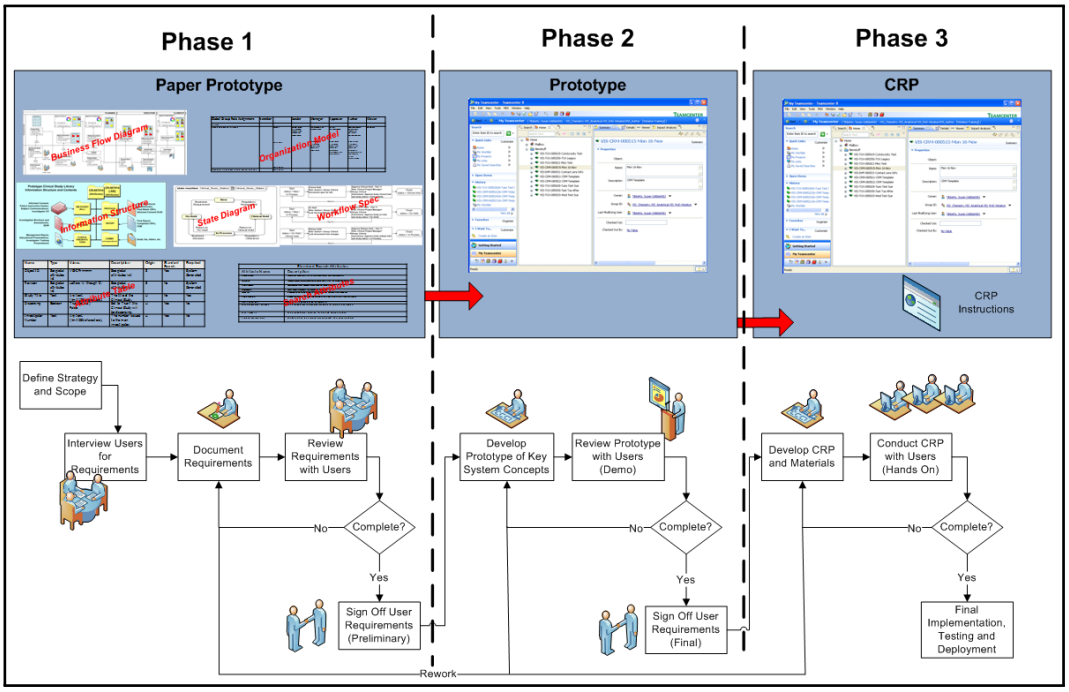


Figure 4.1: Project Management Process Phases.

4.3.2 Phase 1: Paper Prototype Phase

The first project process phase is called the Paper Prototype. It is called the Paper Prototype because the visual model artifacts were created in Microsoft Office™formats like one would write a document on paper. The use of common computer tools reduces tool learning time and provides an artifact definition mechanism that all project members, especially the stakeholders, can use which makes the creation, revising, and review of the artifact proposals as easy as possible. The Paper Prototype phase contains the main requirements elicitation work and produces the various graphical and text artifacts for the enterprise objectives models, information models, and the behavior models.

The paper prototype artifacts need to be fast to create and easy to update. Often the review takes place by projecting the Business Flow or Information Model onto a whiteboard and then the projected image is reviewed and improved. When consensus is achieved, the electronic image is updated. The artifacts are reviewed and refined until stability emerges. Some artifacts stabilize before others; however, business process re-engineering takes longer as business objectives and goals are questioned.

As parts of the requirements, mostly expressed as visual models in the paper prototype form, stabilize, those artifacts proceed to the Demo Prototype phase. It should be noted

that stability of part of the requirements was not the same as completeness of all of the requirements. The stable pieces of the requirements can be viewed as SCRUM user stories.

4.3.3 Phase 2: Demo Prototype Phase

In the Demo Prototype phase, the design and initial implementation work is done and a demonstration of the possible realization is created. The demonstration is shown to the customer and feedback on the visual organization of the user interface as well as input on missing information or incorrect interpretation of the requirements is incorporated into a new prototype increment for the next iteration.

The Demo Prototype is refined until customer acceptance. The time to create a Demo Prototype implementation in the case study examples was targeted to fall within a three-to-five-day timeframe. This is a similar timeframe for SCRUM user story implementation [Coh10]. This timeframe is consistent with implementation via configuration. In the cases where code needed to be developed for a special business rule, the timeframe often increased to several weeks. The demonstration prototype refinement time took, on average, one week. Often several demonstration prototypes were under construction or refinement in parallel. The work overlap is possible, because the time spent while the stakeholders are reviewing one Demo Prototype can be used by the developers to work on another Demo Prototype.

The management of the Demo Prototype work aligns with the traditional aspects and approach of SCRUM managed projects.

4.3.4 Phase 3: Conference Room Pilot (CRP) Phase

In the Conference Room Pilot (CRP) phase, the related demonstration prototypes are brought together. In preparation for the CRP, additional implementation details beyond the demo prototypes are completed and tested. The additional details include things like completing error handling, user message refinement, localization, icon refinement, and minor functional features such as report formatting. At the CRP point, somewhere between 85-95% of the solution is implemented, is fairly well tested, and is installed in a test environment.

The customer executes the solution for themselves by going through the Business Flow simulating real work. This hands-on experience is the last verification and is a preliminary acceptance test. After the third solution delivery cycle, the SAMEM and project management process progressed to the point where a punch list of minor technical corrections takes only a few engineering days to implement. More importantly, after the third solution delivery for CS-1, there were no major corrections due to incomplete or misunderstood requirements.

4.3.5 Using RM-ODP

The RM-ODP serves as a framework to guide the project activities. This is shared with the customer as well. The five viewpoints (Enterprise, Information, Computational (Behavior), Engineering, and Technology) are used to keep the conversations focused on the important questions and topics for the current stage in the development process. The guidance and focusing aspects provided by the RM-ODP are for the appropriate level of abstraction in the context of the project management phase. The conversation focusing aspects are part of the communication accidental complexity removal efforts. In addition, the conversation focusing aspects also support achieving due diligence in the project process.

The image in Figure 3.3 communicates the RM-ODP framework both for development team and customer education. The Enterprise, Information, and Computational Viewpoints are the critical ones for creating the requirements specification, as they use the business vocabulary. The artifacts from the Information Viewpoint are transformed into behaviors using the Action Transformation Pattern, which is explained in Sub-chapter 3.4.1.3. Architectural design activities create the design model artifacts of the Engineering Viewpoint from the requirements. Using the Architectural design artifacts as a starting point, the detailed design decisions produce the Technology Viewpoint specific artifacts for the realization. The decisions and artifacts from the Engineering and Technology Viewpoints are part of the design specification required by the FDA. In a domain without regulatory requirements such as those imposed by the FDA, the Engineering and Technology Viewpoint design artifacts can be fewer and simpler.

Figure 3.3 is useful as a different and sometimes more easily understood view of the project process, as compared to a Gantt or Pert chart. It was found that a copy of Figure 3.3 created with highlights to show the current areas of work was useful as a temporary introduction image for a meeting. For example (see Figure 4.2), a green background showed a stable or complete viewpoint, yellow background indicated a viewpoint of active work, and red background showed a viewpoint that had not been started. In some cases, a puzzle piece from Figure 3.1 or Figure 3.4 would be mapped onto Figure 3.3 to communicate what was being worked on. The mixing and matching of different visual models was dynamic and driven by communication needs with the stakeholders. Each CS-1 and CS-2 solution had a slightly different set of stakeholders and the SAMEM could adapt to the communication needs from a shared set of visual techniques.

In Figure 4.2, the puzzle pieces with the DHF label indicate that the Information Model requirements are deemed complete (green background) while the Computational (Behavior) requirements are in-progress (yellow background). The project status for the Research Notes (RN) is in-progress (yellow background) for the engineering design work. The reuse of images in combinations enhances the communication with the customer.

Figure 3.3 or Figure 4.2 helped to explain why a certain requirement appearing statement is not a requirement but a design decision or constraint which assisted in keeping the requirement specification in a *clean* state. Clean requirements are something the FDA looks for in an audit and are interpreted as refinements to *intended use*, *intended users*, and *safety*. Those three phrases are interpreted in the context of the medical

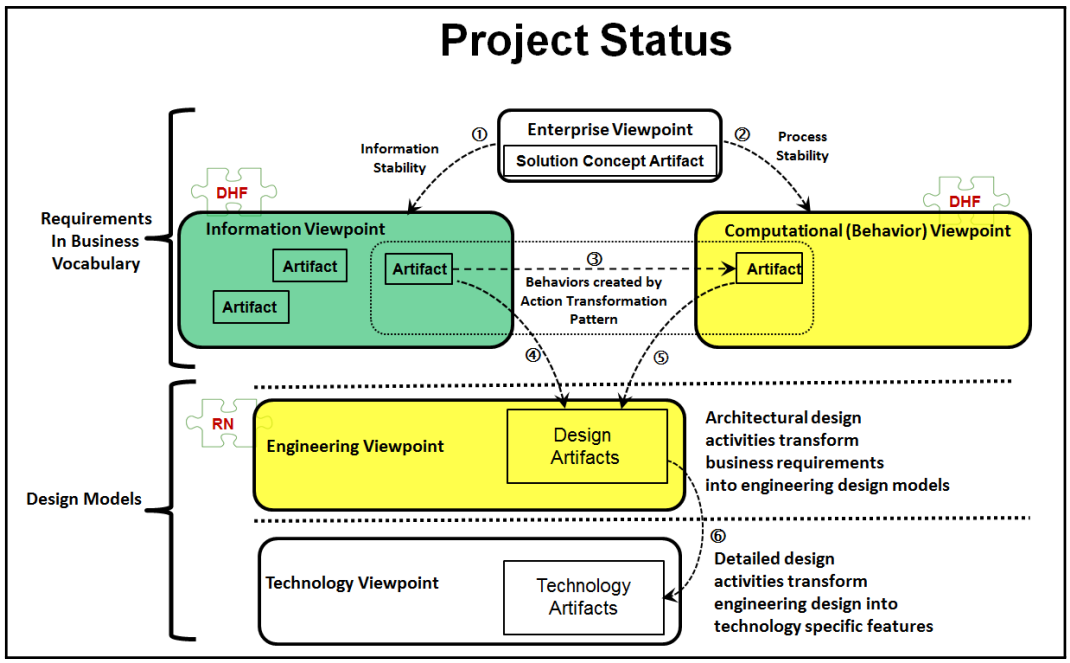


Figure 4.2: Project Status Example Mapped to RM-ODP Model.

product and current best practices. Statements in the requirements specification that reflect a certain solution technology can inhibit passing an audit, especially when the actual product did not use that technology. The working definition of *clean* varies between FDA auditors and is relative to the class of medical device.

4.3.5.1 The Enterprise Viewpoint and the Solution Concept Model

The Enterprise Viewpoint is equated with establishing the solution concept [Bro95]. There is a concept for the entire solution and an initial partitioning of the entire solution into a set of sub-solutions. The objectives of creating a set of sub-solutions are to have manageable units for development and to deliver some business value as soon as possible. A concept is created for each sub-solution, which is called a sub-concept. The sub-concept is the foundation of the Enterprise Viewpoint for the sub-solution. For each sub-solution, the sub-concept becomes the starting point for the elicitation of requirements and the refinement into design alternatives.

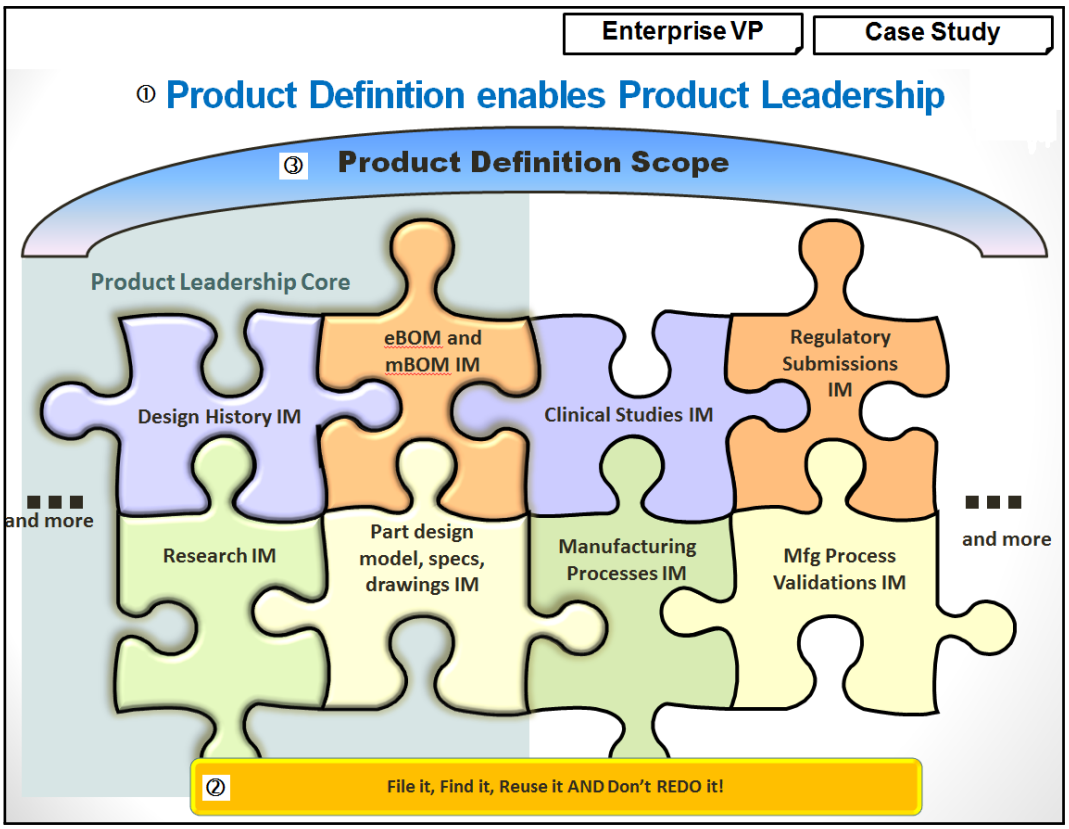


Figure 4.3: Enterprise Viewpoint Solution Concept Model.

Figure 4.3 (same as Figure 3.1 but repeated for convenience) shows an example (from CS-2) of the entire solution concept model (most abstract) for the Enterprise Viewpoint. This image is divided into three components:

1. The business goal is at the top as a title, ①. In this example the business goal is to have Product Leadership, i.e. build better products than anyone else. The company (CS-2) believes the key to achieving Product Leadership is through control of the product definition information and processes.

2. The project mantra is in the box at the bottom, ②. The project mantra is a statement of the business benefit of achieving the business goal. By controlling the product definition, new products will be created as effectively as possible (waste is reduced).

3. The graphical image of the full business solution concept with the initial sub-solution partitioning is expressed as a graphical image in the middle, ③. The image defines the parts of the business that are considered to be in scope for the project. Business areas not in the picture are either out of scope or are too minor to be included at this level of abstraction.

The major goals of establishing the solution concept model as a single graphical image are to have a high level summary that all stakeholders can agree to, have a memorable image for effective communication of the solution essence, and provide a starting point for graphical refinement in an iterative & incremental process.

Both the business goal, *Product Definition enables Product Leadership* and the mantra, *File it, Find it, Reuse it AND Don't REDO it!*, are used constantly through the process as a touchstone to ensure that the conceptual integrity is maintained. As requirements are gathered in the form of information model artifacts or solution behaviors, they are constantly questioned relative to the business goal and the mantra.

- Is the information artifact an essential part of the product definition or not?
- Does the behavior directly support the mantra or not?

The solution model is developed from the business priorities listed in the strategy, which is a separate issue from this thesis. In the case of both CS-1 and CS-2, the primary business goal is to manage the product definition and the supporting data required by the FDA. The product definition can be seen as a large information model. Refinement activities break the product definition down into a set of smaller Information Models (IM) such as Design History, Research, etc. The ones shown in Figure 4.4 (same as Figure 3.4 but repeated here for convenience) are the initial smaller IMs considered for the three-year time scope for the CS-2 project. This single image is easily remembered by the customer and at a high level of abstraction is a summary that can bring all project people into agreement or at an early stage expose disagreement.

The visual model in Figure 4.3 enables a graphical starting point for refinement while establishing context. One of the goals for the SAMEM is to show refinement through sets of more detailed graphical models, for example Figure 4.3 to Figure 4.4. The chain of refinement images provides an audit trail of design decisions. In practice, a circle is drawn around a puzzle piece, an IM in Figure 4.3, and discussions of the benefit of developing that particular IM as opposed to some other piece are held. When looking at one of the IMs, a similar but more detailed model, a sub-IM is created, Figure 4.4, which starts the next set of model refinement actions. At this point, a high level business problem scope architecture is produced but still at the enterprise level in the RM-ODP process.

The factor that is often the most important in deciding the realization order of the IMs or sub-IMs is which group within the organization is most receptive to an improvement. However, at times the customer management will target an area they think needs help even if the group is not receptive to an improvement. In a few cases, a certain IM or a sub-IM refinement produced a foundational component that should be realized sooner for the supporting benefits to other IMs.

It can take several meetings with the stakeholders to develop a clear and compact solution concept. In the case of the example in Figure 4.3, it guided the project direction of CS-2 for over three years and five sub-solution deployments. The use of graphical expressions of the project definition allows for effective navigation through the fog that

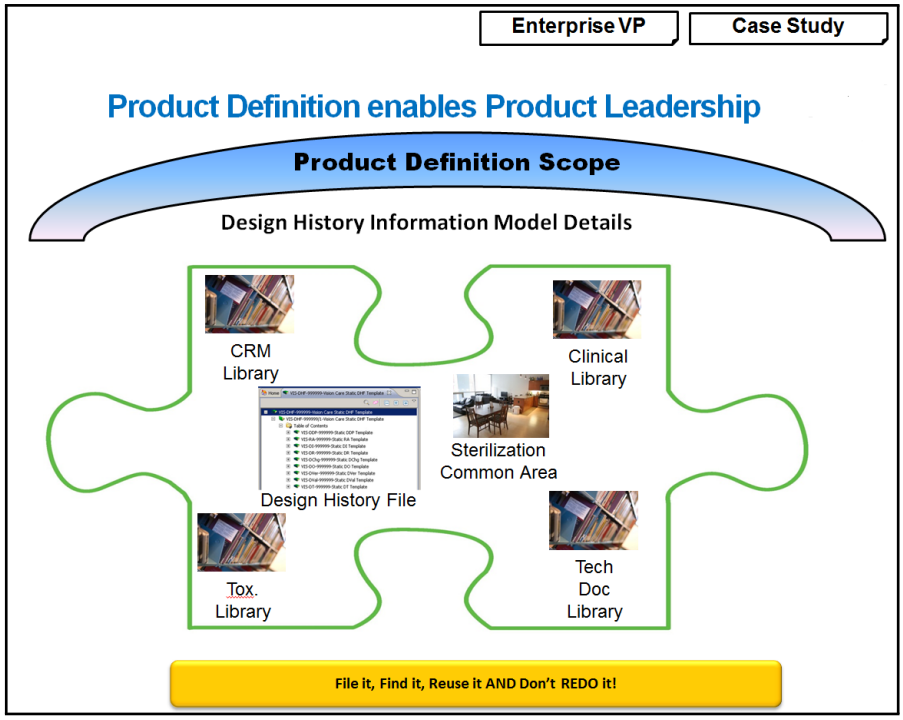


Figure 4.4: Sub-solution Concept Refinement Model.

exists early in the project. The fog is the consequence of people having a rough qualitative idea of an improvement possibility, but they are still in the early stages of learning to express the idea. Brooks [Bro10] discusses this situation at length.

The image of the puzzle is a representation of the modularization ideas from David Parnas, "...module is considered to be a responsibility assignment" [Par72]. One of the ways the image in Figure 4.3 is used is to focus a meeting by drawing a circle around a jigsaw puzzle piece. The puzzle image was repeated to specify more detailed requirements (see Figure 4.4). By repeating the puzzle piece image and adding details for the information components of the Design History Information Model, a communication bridge is created. The repetition of image styles is a great help to communication as it sets a pattern and allows context to flow down from the highest level of abstraction. This is a repeated application of the module with information hiding idea as described by Parnas [Par72]. The stakeholders verify the correctness of highest level of abstraction in Figure 4.3 and then the next level of modules in Figure 4.4 is verified. The set of these images define and maintain the solution concept. They are some of the Model Artifacts as shown in Figure 3.2 and become part of the requirements specification.

Finding the most effective models for the Enterprise Viewpoint is an action of innovation and experimentation. There is not a single answer. However, a process of trying new communication forms on a select few of the stakeholders can be done. A smaller audience of open and respected stakeholders will often be easier to try new model forms

with.

4.3.5.2 The Information Viewpoint Development

One of the first modeling lessons learned is that the standard UML model diagrams should not be shown to the stakeholders. The feedback came from the reaction, mostly emotional, by the stakeholders to the UML notations. The standard UML diagrams are too harsh, have simple lines, no color, and contained unneeded information. The case study experiences discovered a need for "softer" visual display for the same reasons that other research has discovered [MvH08]. In addition, the softer visual notations need to be developed in conjunction with the stakeholders. Sketching model ideas on a whiteboard is a good way to introduce new notation, even if the notation ideas are based on some other standards such as the UML. Examples of the new visual display for the requirements can be seen in Sub-chapters 3.5.2 and 3.5.3.

It is also necessary to reduce the content of the UML diagrams and UML elements to the essential data needed for pure information modeling. Applying the RM-ODP framework to the problem helped to discover the essence of new models. The Information Viewpoint models are based on the UML Class Model ideas but only use some of the modeling features.

In the case studies, the sub-IM level started with identifying the major business information units and structures of information. At the Information Viewpoint level, only business terms are used. The software terms of *object*, *class*, or *tree* are completely avoided; instead the terms of *information unit* or *unit of information* or *structure of information* are used. The term *document* is avoided. The rationale for not using *document* was discovered in the strategy project, because they (CS-1) had created many documents with diverse and unrelated information, which prevented reuse. In order to achieve information reuse, the CS-1 project emphasized homogenous and generic units of information. For example, a Chemical Research Method (CRM) definition should mention the material type but not any product name or development project name.

For both of the case studies, the starting point is with the information units because that was the most stable thing in the business environment. The stability was established by many years of product development and the FDA regulations. Identifying the product definition information as the most stable component was the application of the SEFP **Stability to Variability**, see Sub-chapter 3.2.3.

The behind the scenes thinking of the development team based the idea of the information units and the information structures on the UML Class Model capabilities. The structures of information are called *Bill of Information* (BOI) due to the customer's understanding of their product Bill of Materials (BOM), which is an example of the **Change Language** SEFP to improve communication. The terminology was an intentional choice to draw an analogy between the idea of standardized reusable parts in their product and standardized reusable units of information in the product definition. This is the embodiment of the mantra: create a unit of information, *File it* in the repository, *Find it* in the repository, determine if it can be *Reused* in a new product. The repository for CS-1 became known as the Knowledge Management System (KMS).

At times it is awkward, but developers need to speak two different languages. There is the business language that is comfortable for the customers and a more precise language for further development purposes. The more precise language is shielded from the customer.

4.3.5.3 Computational (Behavior) Viewpoint Development

As the information units are identified, the question immediately asked is, "What is the business process for management of this unit?" The answer to that question starts the elicitation of the behavior requirements, which belong to the Computational (Behavior) Viewpoint. For this use of the RM-ODP framework, the SAMEM redefines the Computational Viewpoint as the Business Behavior Viewpoint.

The business process of the information unit management is expressed in terms of the capabilities of the UML Activity Model. For CS-1 and CS-2, the customer seldom had a clear process definition, so that the requirements elicitation work is often defining and clearly documenting the process for the first time. This process is called the *Business Flow*. In developing the Business Flow, additional information units are often discovered, existing units modified, and on occasion some are removed. The removal was occasioned by the realization by the customer that the information is no longer applicable to the business.

In Figure 3.3, an arrow points from a unit of information in the Information Viewpoint to a unit of behavior in the Computational Viewpoint. Below the arrow appears the text "Created by the Action Transformation Pattern." This demonstrates the application of the Create, Retrieve, Update, Delete (CRUD) pattern to each information unit. This pattern is used as a checklist to ensure that the information unit management process has all the lifecycle actions. The CRUD pattern is extended in two ways. First, the major actions are specialized, for example Create has sub actions of Create Item, Create Revision, Create Structure, etc. Second, each action has characteristics suggested by the Volere Method [RR99], which results in a matrix, the Action Transformation Matrix (ATM). This matrix acts as a two-dimensional checklist to help make sure all the behavior requirements are addressed. The actual entries in the ATM are references to other sections of the requirements specification, so that it acts as a 2-dimensional index. See Figure 3.20 for a full example and Figure 4.5 for a partial example. The rows of actions and the columns of characteristics are extended as needed for the domain and customer.

Main Action	Extended Action	Who can access	When can access	Initial value
Create	Item			
	Item from template			
	Duplicate Item			
	Structure			
	Reference (add)			
Retrieve	Item Values			
	Where Referenced			
	Structure			
Update	Value			
	Structure Add			
	Structure re-org			
	Status/State			
Delete	Item			
	Structure Membership			
	Whole Structure			
	Reference			

Figure 4.5: Partial Example of an ATM.

4.3.5.4 The Engineering and Technology Viewpoints

Although the scope of this thesis focuses on requirements which correspond to the Enterprise, Information, and Computational (Behavior) Viewpoints rather than the Engineering and Technology Viewpoints, a brief description of the relationships will be given to round out the understanding.

The high-level design actions and artifacts are considered the Engineering Viewpoint. These are documented for the case study companies per FDA regulations. One major design choice that is made in the Engineering Viewpoint is the "build or buy" decision. For the case study companies, the decision was to buy Teamcenter, because its capabilities matched many of the requirement needs. Each requirement is examined in the light of the engineering architecture possibilities and mapped onto the proper set of features. The requirements for information units were mapped onto the Teamcenter built-in types and attributes. The various requirements behaviors (CRUD) were mapped onto the built-in actions. The mapping actions are the design work and choices made during the Engineering Viewpoint. The design choices and accompanying rationale are the contents of the design specification. Examples of design artifacts and rationale from the case studies are shown in Sub-chapter 3.5.4.1.

The low-level design artifacts are seen as contents of the Technology Viewpoint. The artifacts here consisted of specific configuration settings for features such as the data

types with attribute properties, access control, specification of the organization (users, roles, and groups), user interface layouts, and approval workflows. There are on occasion behaviors or business rules that cannot be configured but need a C++ code extension for realization. Examples of the Teamcenter implementation for the case studies are shown in Sub-chapter 3.5.5.1.

In creating the realization through the Teamcenter tools, sometimes constraints or limitations would arise that forced the updating of a requirement. For example, the realization of a requirement might only be possible by extending the Teamcenter product through custom code. However, an extension through custom code conflicts with maintaining an easy update path to a new version of Teamcenter, since the extension would need updating. These conflicts and compromises are discussed with the customer and an agreement is reached.

4.3.5.4.1 Engineering Viewpoint Impacts on the SAMEM and Project Process

The design work done at the Engineering Viewpoint level has two major impacts on the project process flow. The normal project process flow is from high levels of abstraction in the Enterprise Viewpoint to low levels of abstraction or concreteness in the Technology Viewpoint; however, discoveries during Engineering Viewpoint work can cause re-specification of requirements. The re-specification arises through invention, uncovering of unexpected business constraints, limitations in the engineering needs or technology, and discovery of requirement mistakes. The re-specification results in cycling back to the requirements work and making corrections. The re-specification can also occur when technology limits force an engineering re-design.

The major impact on the SAMEM is a reinforcement of the iterative & incremental approach, not only for initial development but also for the rework cycles. There is an additional impact on the artifacts, as the rework cycles are essentially a distribution of the work over time. The work distribution factor means that the intent of the artifacts and the reasoning behind them must be available for re-examination and improvement.

Brooks [Bro10] extensively discusses the cycling back to requirements and through multiple design alternatives. In any high quality design type of work there will be natural iterations or design cycles as understanding is gained of the consequence of a design choice and of technology limitations. However, there can also be unnatural iterations, often referred to as *analysis paralysis*. The SAMEM needs to support and track the natural iterations and highlight the unnatural iterations as quickly as possible. The SEFPs are a mechanism by which light and understanding is focused on the unnatural iterations. The SEFPs of **S2V**, **SoA**, **Patterns**, and **C&C** have been shown in practice to be most useful.

For example, a requirement might call for the long-term persistence of an information unit. The first design alternative at the Engineering Viewpoint level might be to save the information in an operating system file. Upon further reflection and taking into account an engineering pattern such as ACID (Atomicity Consistency Integrity Durability), a second design using a relational database is developed. When other requirements are

added, such as structures of information units, then a third design using a COTS PLM application (such as Teamcenter) is proposed. The designs and alternatives multiply as the engineers consider groups of requirements and best practices. Tracking is needed to keep the work organized and effective.

There are different ways the design alternatives can be managed (see patents [Mat04], [Mat05], [Mat09]). For a start, each design alternative gets a unique identifier for accurate traceability. The design alternative is linked back to the requirements it attempts to satisfy. Along with the link, an evaluation of the advantages and disadvantages with respect to the requirements is valuable for evaluation in a design review.

The advantage that the Engineering Viewpoint brings with the design alternatives is the opportunity for invention and innovation in the solution. The invention could be small as in the new combination of existing design patterns or the invention could be large as the creation of the Web through the HTTP protocol and HTML. The SAMEM allows for the possibility of invention by deliberately separating the focus on *what to build* from *how to build*. The explicit emphasis on short iterative & incremental cycles assists in the ability to develop multiple design alternatives.

The design work that happens during the Engineering Viewpoint is often referred to as high-level design. The larger the solution, the more critical it is in design work to separate the big decisions from the small decisions. The SAMEM supports this by abstraction and using RM-ODP to help organize the design decision considerations.

4.3.5.4.2 Technology Viewpoint Impacts on the SAMEM

The major impact on the SAMEM of the work that happens in the Technology Viewpoint is the discovery of incorrectness, incompleteness, and conflicts to the work done in the other viewpoints. This is a risk that cannot be completely eliminated with a forward-only process. These discoveries will generate a re-evaluation task in the project process that will cycle back to an earlier viewpoint. Often the cause is an incorrect assumption about the features or capabilities of the technology in satisfying a requirement or an engineering design. Sometimes an incompatibility between system components or component versions will arise during the implementation or testing. These incompatibilities are often not communicated in release notes or other vendor documentation.

When this happens, the traceability from low-level design to high-level design to requirements is crucial to effectively correcting the problem. If a requirement needs to be adapted, then the ability to explain why the technology will not support the realization is very important. This is viewed as a type of verification, although the cost is higher because the failure happened after more effort was spent.

Because the SAMEM is built around fast iterations and small increments, it is possible to slip a new cycle into the project plan with minimal impact, thereby mitigating the cost. However, a new iteration will add a cost on the project budget. Good project risk management planning in the beginning should allow for some of this to happen. The size of this risk is primarily dependent on the experience of the development team with the technologies and how hard they are pushing the technology.

4.4 Lessons Learned

This sub-chapter summarizes the lessons learned as described earlier and expands the list with additional learnings. The majority of lessons learned came in the first several solution projects done for CS-1, but from every project there is some improvement in the SAMEM methodology. In some cases the improvements are to the process and iteration management and in other cases in the form of the artifacts. There are lessons learned that relate to adapting the SAMEM to a domain or project personnel. The lessons learned are compared with results from other surveys of MDE practices.

4.4.1 Case Study Lessons Learned

There are several unique practices in the project case studies that contributed to the success of the case study projects. At several places earlier in the thesis, lessons learned are listed in the context of the case study work. The previously listed lessons are gathered together and organized in this chapter for summary purposes.

4.4.1.1 Iteration Lessons Learned (ILL)

ILL1 – the verification review time is scoped to about one hour of the stakeholders' time. It was difficult in the case study companies to schedule longer sessions due to the primary job responsibilities of the stakeholders, especially with the top subject matter experts or responsible executives. With advanced scheduling, often several weeks in advance, longer sessions such as half a day could be set up with multiple stakeholders. In practice, a review unit consisted of one or two pages of graphical images to check. This is simple to accomplish and the hour or often less time needed could be effortlessly worked into the stakeholders' schedule.

ILL2 – the review content volume had an influence on the verification review time through the amount and the coherence of the information, but the rule-of-thumb discovered is about five information units with associated attribute definitions (approximately 15 attributes per unit) and state machine or 15 tasks from a Business Flow. In cases where an information unit had many attributes, over 20, the number of information units to be reviewed was reduced. If the Business Flow tasks involved more complex cycles of work, in contrast to a linear progression, then the section size was reduced to cover the complex cycle portion. The experience and knowledge of the stakeholders had influence on the size of the review package. There were constant adjustments to fit to the stakeholders' abilities and schedule opportunities for review but always with the goal of one hour maximum of review time.

4.4.1.2 Project Process Lessons Learned (PPLL)

A major concern of the sponsoring stakeholders, a person responsible for the money being spent, is that reasonable progress is being made. The definition of reasonable often depends on the software project experience of the sponsoring stakeholders.

PPLL1 – the case study experiences show that the project process roadmaps help with establishing positive customer expectations. There are several visual images that can be included in the category of project process roadmap and were used in the case study solutions as appropriate to clearly communicate with the stakeholders.

The use of visual models in the SAMEM - as much as possible - helped in the project progress communication. One example is that two requirements images can be compared and the differences highlighted. One manner of highlighting used was to draw a border around a portion of an earlier version to indicate what was worked on in the iteration, then show the updated version with new specifications. Another mechanism is to show the new models created during an iteration. Since most of the models can fit on one page, they are easy for the stakeholders to acknowledge.

Figure 3.1 and its refinement such as Figure 4.4 show the solution area being worked on. For a more detailed picture of the current work within the information model, the SOD (example Figure 3.5) is used. The ability to trace from the high-level solution concept to a specific detailed set of information units to Business Flow tasks is crucial in establishing a logical and traceable path of what is being worked on.

An example of the first process image is in Figure 4.1, which shows the overall progression from requirements elicitation to deployment. This image establishes the flow of work for a solution and provides the basis for explaining the application of the agile-like iteration approach. It helps to answer the questions: "How is the project work being carried out?" and "Where in the process is a certain piece of work?"

Combining Figure 3.3 with Figure 4.1, as in Figure 4.2, explains which type of work was being done. In this case, the type of work relates to the level of abstraction. For example, is the project in the Enterprise, Information or Computational (Behavior) Viewpoint with the requirements being gathered or is a Demo Prototype being developed in the Engineering or Technology Viewpoint?

PPLL2 – the ability to work in a common set of tools that allow images to be combined is very helpful to accomplishing the communication needed for reporting project progress.

PPLL3 – stacked ATMs are effective as an interface between the Front-end and Back-end teams. A stacked ATM is an ATM with the cells filled with references to requirements, engineering designs, and technology designs. Front-end is the name applied to the work done by the primary development team and Back-end is the implementation work done by an external contracting group, often off-shore. Each cell in the Action Transformation Pattern Matrix gets a unique identifier. The interface to the Back-end team is the set of information consisting of a list of ATM cells, the associated requirement specification sections, and a set of implementation constraints.

PPLL4 – the key lesson learned is that different stakeholders respond best to different presentations. Some stakeholders are interested in which tasks have been accomplished and the next set to be accomplished. Other stakeholders need more information to feel comfortable about the progress, such as the rationale for the task ordering.

4.4.1.3 Communication Lessons Learned (CLL)

The high level goals of ***HL-GOAL-1*** and ***HL-GOAL-2*** are focused on communication. The following pragmatic lessons learned reflect the difficulties of finding a common communication approach among many people.

CLL1 – the primary lesson on communication is to be open to try different mechanisms. Sometimes hints on what will work can be found during the stakeholder introductions when they recount their hobby experiences. For example, one stakeholder is involved with photography as a hobby which indicated a visual orientation. A key to effective communication is the application of human cognitive psychology limitations.

CLL2 – the discovery and use of neutral terms, standard business terms, or domain-specific terms especially in the work of the Enterprise, Information, and Computational (Behavior) Viewpoints is critical. Neutral terms of "information unit" and "information structure" were used in the case studies, rather than computer science or programming terms like "object" and "class". During the Engineering or Technology Viewpoint discussion with the stakeholders, the terms from the requirements could be mapped onto the computer-specific design artifacts and terms. The mapping provides a vocabulary transition and traceability for the stakeholders and increases their confidence in the solution realization.

CLL3 – changing the terminology used in the team from the standard terms, like document, brought us to the Bill of Information concept. Deliberately changing terminology can break mindsets, so that innovation is stimulated through new perspectives.

CLL4 – the solution is built on the Teamcenter PLM product and it was discovered that by not using common PLM terminology or jargon, it was easier to introduce Teamcenter and new solution concepts to the business users of the system.

CLL5 – an important help is the socialization by the customer business side team members to other people in the company of the improvements coming in the new solution. Improvement process drawings were created to show the new customer where they had important parts to play in the process.

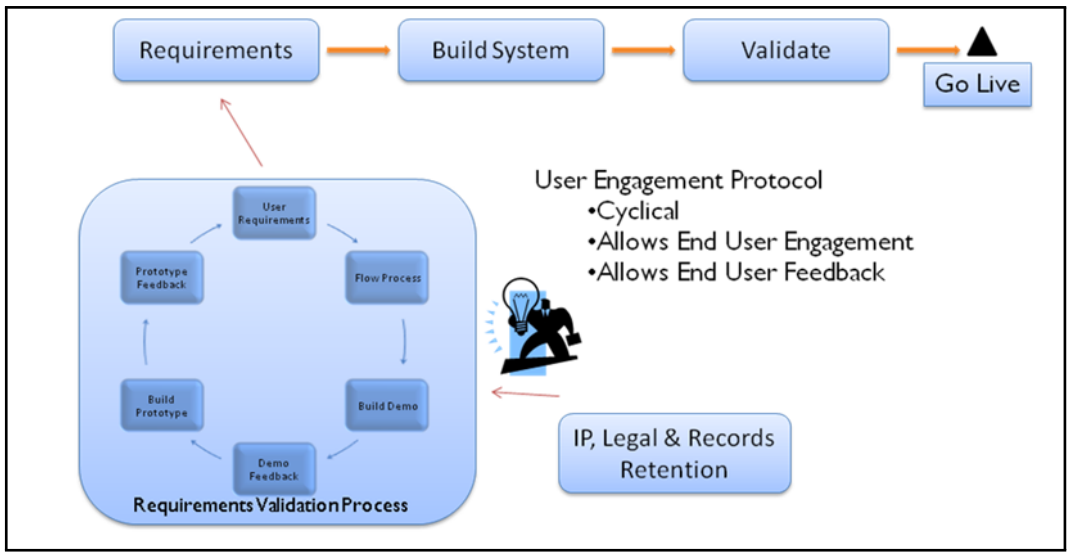


Figure 4.6: A Stakeholder-developed Image of the Project Process.

CLL6 – it often becomes very valuable to use stakeholder-developed images, not just from a political relationship standpoint, but because the image has captured a valuable insight. The value of a stakeholder being involved enough to generate such an image is very high. At times the stakeholders will develop a visual model (see Figure 4.6 for a project process example) that helps with communication within their company.

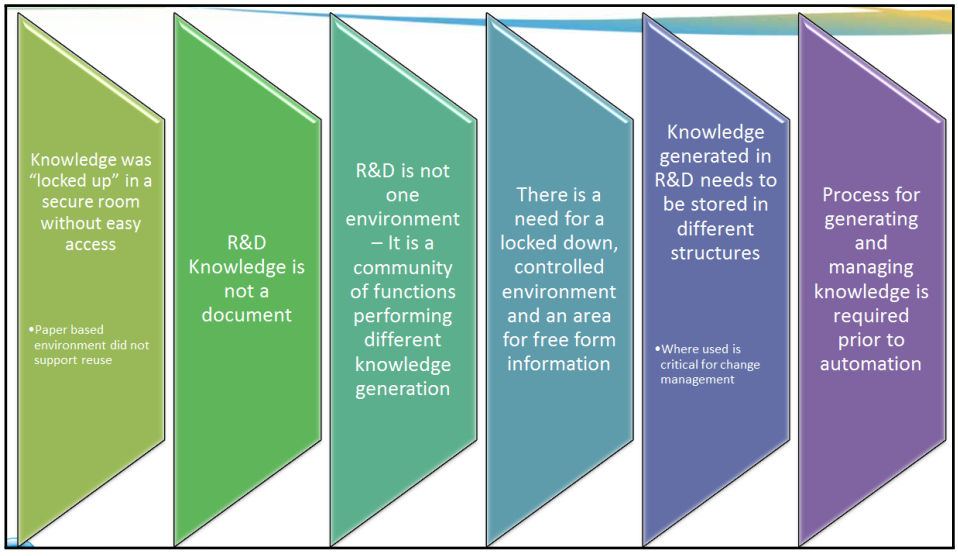


Figure 4.7: Stakeholder Image of Company R&D Environment.

The lesson is that the SAMEM artifacts are not the only source of valuable communication artifacts about the project process and the solution. The ability to integrate and

use stakeholder work for the appropriate viewpoint is critical.

Another example of a stakeholder-generated image is seen in Figure 4.7. This image was developed internally in the CS-1 company. It was used in an external conference presentation [SM11]. This image falls into the Enterprise Viewpoint as it describes the nature of research and development work in the company from an important stakeholder's perspective.

CLL7 – case study experience showed that stakeholder consensus happened quickly on the information units but took longer for the attributes.

4.4.1.4 RM-ODP Framework Lessons Learned (RMLL)

The use of RM-ODP as a framework for the coordination of modeling techniques was critical. RM-ODP helped with decisions about what concepts to model and which modeling techniques were useful. For the Enterprise Viewpoint, the initial attempts showed that the UML was inappropriate, so a different approach was developed. For the Information and Computational Viewpoints, the UML was appropriate but in a restricted and simplified graphical form. More details on UML limitations are in Sub-chapter 4.4.1.5.

RMLL1 – RM-ODP is effective in controlling the requirements gathering process by helping to identify those times when conversation drifted into implementation discussions. In the domains of the CS-1 and CS-2 companies, the RM-ODP viewpoints approach worked with Information Viewpoint first, Enterprise Viewpoint second, Computational Viewpoint third, Engineering Viewpoint fourth (start of design work), and Technology Viewpoint last. Start with what information a group is responsible for creating. With that agreement in place, the next questions of "What groups do you interact with?", "What information do they need from you?", and "What information do you supply to them?" can be addressed. Think of this as a RACI (Responsible Accountable Consulted Informed) model for information. The information model approach produced a "what" needs to be done discussion, not a "how" to do it discussion.

RMLL2 – with the agreement on the information model as a basis, the order in which the information is developed and how it evolves within the Business Flow is relatively straightforward.

For the problem domain for which PLM is the optimal engineering solution, the focus on the Information Model first will be most productive in achieving insight to the essence of the problem. It also makes reaching agreement with the customers easier. Once the Information Model is in place, the behaviors can be addressed. The PLM maturity and understanding is similar in most product producing organizations. The information model first approach was successful. It was easy to get agreement on which information belongs in the solution and the discussions were constructive as understanding was shared among the business users. In many cases, it was the first time they had such discussions. The observation that the process is secondary to the information model was confirmed. The information model is more stable than the business process, especially as most companies have continuous process improvement going on.

RMLL3 – one also needs to pay attention to the order in which information is created and linked together. The linking of information is not a violation of coupling [Mye75] if

it is a natural occurrence of building the complete information unit. The coupling and cohesion principles help to draw the boundaries in the partitioning of the implementation iterations and also help define a boundary for Front-end / Back-end partitioning of work. Front-end work is done by the main project development team, while Back-end work can be shipped off-shore or given to a contract organization.

4.4.1.5 UML Modeling Flexibility (MFLL)

While UML models are the foundation of many of the SAMEM modeling approaches, there are changes needed to improve communication. There are also modeling techniques needed that go beyond the UML.

MFLL1 – one of the important lessons learned in the case studies was having flexibility in changing the notation when stakeholder communication suffered. UML was the initial modeling choice due to its ubiquity. However, the UML visual display was changed when it interfered with communication and sub-sets of the capabilities were selected when appropriate. The starting point UML models used in the Information Viewpoint and Computational (Behavior) Viewpoint are the Class Model, State Model, and Activity Model.

UML Class Models provided the initial starting point for the information artifacts. However, one of the first modeling lessons learned was that the stakeholder communication did not work well with standard UML artifacts [MvH08], [Mat11]. The standard UML diagrams are too harsh with simple lines and geometric shapes, no color, and contain unneeded information.

MFLL2 – for non-computer experts there is a need for "softer" visual display for the same reasons that other research has discovered [MvH08].

Full UML Class Models were difficult for the stakeholders to grasp. The simplicity of the visual notation leads to confusion and the standard class appearance had many pieces that were unnecessary for requirement models. A softer UML Class Model is created through the following changes. Method definitions, which embody a *how*, do not come into existence until the design stages and are eliminated from requirements information models. Only the simple *Association* relationships are used. Peer-to-peer relationships, often a reference, and parent-child relationships, the BOI, are the categories of *Associations* discussed and documented. Bi-directional navigability is assumed for all relationships, so those UML decorations are not displayed. Cardinality is shown on relationships only when important for clear communication. An n-to-m cardinality is assumed, except of structures where a 1-to-n (parent-child) cardinality is used. Color is added to the visual notation for the class models to distinguish entities which, as previously described, were renamed to *information unit*.

MFLL3 – placing the attribute definition in a separate table is a response to a lesson learned for improved customer communication. The splitting of the artifact representations helps the communication by enhancing focus. There is an artifact that shows the information unit with the relationships between them and an artifact with the attribute details of each information unit shown separately. The two artifacts focus the conversations on either the validity of the information unit and its relationships or the

information unit details. The splitting of artifacts content makes the review content smaller for each verification step.

MFLL4 – in many cases the stability of the attribute definitions and list did not happen until the demo prototypes were reviewed and in some cases not until after the CRP.

MFLL5 – UML Activity Models were the basis of most of the behavior requirements. Due to a stakeholder suggestion, the UML Activity Model notation was switched to Business Process Modeling Notation (BPMN) after the first year of work. For requirements purposes there is no essential difference in either expressive power or visual notation. Since Microsoft Visio™has templates for BPMN, this eased some of the documentation work and the ability of stakeholders to make edits during independent review sessions. Swim lanes corresponding to user roles are not used for the main Business Flow, because it is more important to have natural flow of tasks. The use of swim lanes can often corrupt the essential business task flow by having to visually jump across the page for the next task. In addition, at the abstraction level of the Business Flow, several user roles can be involved in a particular task. During refining and verifying the Business Flow tasks, often temporary and clarifying annotations were added in Visio™to the flow models. If UML Activity Models were used more strictly, then the clarifying text could be placed in a UML Note artifact.

MFLL6 – the UML State Model is used virtually unchanged from the semantic aspects and diagram notations. Most of the stakeholders, in both CS-1 and CS-2, have either a strong engineering or scientific background and are familiar with state machines. It helped that the state machines were often a simple progression from a new information unit to an approved unit.

4.4.1.5.1 UML Notation Use Summary

In general, the UML provided a good starting point for modeling, but the essence needed to be extracted. The essence can be summarized as *data* and *behavior on data*. A factor in extracting the essence is to forget about the object orientation in UML. For UML Class Models this means seeing them as units of information, in the sense of modules as Parnas [Par72] describes and replacing the programming language methods with the behavior ideas of the CRUD pattern.

Sub-chapters 3.5.2.1, 3.5.3.1, 3.5.4.1, and 3.5.5.1 list modeling examples and the relationships to UML when present. There are parts of the UML not used or specialized for requirements modeling:

- Use Case Models – not used because the content is too shallow. A UML Use Case Model has only an Actor name and a Use Case name. The use of only two names open up many possibilities for misunderstanding. From a business value perspective, there is too much work for too little value.
- Class Models:
  - Generalization – this relationship is for design and is used at that level, not for requirements.
  - Methods – there are no methods at the requirements level other than general CRUD actions which appear as tasks in the Business Flow and as use case (example in Figure 3.17) refinement of the task.
  - Specific Associations – Qualified and Dependency. From a requirements perspective these associations only add complexity, not clarity. Neither of the concepts exists in requirements, although they can appear at the design level.
  - Interfaces – provided or required. The interface concept in UML indicates an abstract class with no implementation. The requirements information models reflect concrete business entities. In the design, abstract classes and interface appear to minimize coding effort via reuse.
  - Template Class – at the requirements level template classes for generating different language implementation are not needed, as the work is at a higher level of abstraction.
  - Association Class – in design and implementation work, association classes are used to realize a business behavior, however, they are not used for requirements as that concept is unnecessary.
- Component – organization into components is a design activity.
- Composite Structure – these structures are design artifacts.
- Communication Model – not used, because the requirements do not have the detailed objects and messages for the links. There is a certain equivalence between

the Communication Model and the Sequence Model. The developers preferred the Sequence Model.

- Package Model – translated into the Information Models (IM).
- Sequence Models – Sequence models are intended to show how a group of objects collaborate in some behavior. At the requirements level the objects with their methods do not exist, but sequence models are used in design activities.
- Deployment Model – not used, result of low-level design activity.

Although many of the UML models are not used for requirement specification purposes, several came into use during various design activities in the Engineering and Technology Viewpoints. Some Class Model capabilities, such as generalization and exact association multiplicities, are part of the BMIDE technology design and implementation work.

In practice, many visual artifacts include additional clarifying text to enhance stakeholder communication. The notes are added only on stakeholder request. This is similar to the UML Note capability, but the UML visual notation was not used.

4.4.1.6 New Modeling Notations (NMNLL)

New modeling notations were developed to address areas that currently have no graphical or modeling standards.

NMNLL1 – although not exactly a new modeling notation, a lesson learned is that color in the artifact models varies from group to group. For example, the CS-1 engineering group found color to be helpful in the communication versus the monochrome visual images in the regulatory group as shown in Figure 3.10.

NMNLL2 – the two new model paradigms are the solution concept model and the SOD. See definitions in Sub-chapters 3.2.1.1 and 3.5.1. New modeling paradigms were developed when needed. The most significant area was for the Enterprise Viewpoint.

NMNLL3 – visual models for project process were developed for improved communication. The RM-ODP framework image, shown in Figure 3.3, is one example. This figure is also supported by text definitions and objectives for each viewpoint. The other main example is the Project Process Phases Diagram, shown in Figure 4.1. The project process phases show iterations in a high project level context.

NMNLL4 – in many cases, rather than using simple text statements, tables created in Microsoft Word™were the best mechanism to organize and present the requirements, such as the attribute definitions in Figure 3.11. The tables are compact and easily understood with labels for rows and columns. Tables also provide a consistent representation with only the most important words in the cells, which emphasizes the importance of brevity for review speed.

4.4.1.7 Computer Tool Flexibility (CTLL)

Flexibility in computer-based tools was an important lesson learned.

CTLL1 – for the primary communication with the customer, tools were needed that they could use, that all team members had, and that allowed for the creation of effective graphical and text artifacts. The common set of tools is the Microsoft Office™formats. The Microsoft Office™tools enabled the emailing of the artifacts and review results back and forth when the development team was not at the stakeholder's location. The flexibility in artifact images and organization helped stimulate innovation in the solution and in the customer's business processes.

4.4.1.8 Case Study Lessons Learned Summary

Each of the preceding lessons learned has a unique identifier. They are collected and summarized in Table 4.7 below.

Table 4.7: Lessons Learned Summary.

Iteration Lessons Learned	
ILL1	the verification review time is scoped to about one hour of the stakeholders' time
ILL2	the review content volume has influence on the verification review time through the amount and the coherence of the information, but the rule-of-thumb discovered is about five information units with associated attribute definitions (approximately 15 attributes per unit) and state machine or 15 tasks from a business flow
Project Process Lessons Learned	
PPLL1	project process roadmaps help with establishing positive customer expectations
PPLL2	the ability to work in a common set of tools that allow images to be combined is very helpful to accomplishing the communication needed for reporting project progress
PPLL3	stacked ATMs are effective as an interface between the Front-end and Back-end teams to separate work responsibilities
PPLL4	different stakeholders respond best to different project progress presentations
Communication Lessons Learned	
CLL1	be open to try different communication and modeling mechanisms
CLL2	the discovery and use of neutral terms, standard business terms, or domain specific terms especially in the work of the Enterprise, Information, and Computational (Behavior) Viewpoints is critical to good communication
CLL3	changing terminology used in the team from the standard terms can lead to insights and innovation

LL ID	Description
CLL4	initially avoid the use of technology jargon until the customers are comfortable with the concepts
CLL5	socialization by the customer business side team members to others in the customer company of the improvements coming in the new solution
CLL6	use stakeholder-developed images, not just from a political relationship standpoint, but because the image captures a valuable insight
CLL7	the communication of requirements artifacts can affect how quickly the artifacts stabilize
RM-ODP Framework Lessons Learned	
RMLL1	control the requirements gathering process by identifying those times when conversation drifted from the issue at hand
RMLL2	with the agreement on the information model as a basis, the order in which the information is developed, and how it evolves within the Business Flow is relatively straightforward
RMLL3	need to pay attention to the order in which information is created and linked together
UML Modeling Flexibility Lessons Learned	
MFLL1	adapt the notation visual or text formats to maximize communication of the tasks at hand
MFLL2	be prepared to alter standard notation to improve communication
MFLL3	use multiple related artifacts for communication and work task focus
MFLL4	realize that the artifacts might take several weeks and iterations to fully stabilize as the prototypes and CRP will reveal new understandings
MFLL5	be willing to switch to equivalent modeling notations for stakeholder relationship maintenance
MFLL6	use existing modeling notations when they work for the team
New Modeling Notations Lessons Learned	
NMNLL1	color can help to organize and differentiate artifacts in visual models
NMNLL2	invent new modeling paradigms when beneficial
NMNLL3	visual models for project process status
NMNLL4	use tables to organize data rather than simple statements
Computer Tool Flexibility Lessons Learned	
CTLL1	artifact creation tools are needed that all team members have and can use

4.4.2 Comparison to Other Lessons Learned Research

Other research lists factors that contribute to the success of initiating a MDE practice and factors that contribute to the failure. The development of both the SAMEM and the CS-1 case study began before this other research was done. However, research cited in [HWRK11], [HRW11], and [WHR14] provides a good set of factors by which to evaluate

the SAMEM as it was used in practice. The experiences of the case studies reported here do align with the other research.

4.4.2.1 Success Factors for MDE

The lessons learned and success factors to a large extent parallel the assessment work reported in [HWRK11], [HRW11], and [WHR14]. Their work reported on several success factors and several failure factors in multiple MDE efforts.

1. Choose a narrow domain.
   - CS-1 and CS-2 companies produce medical devices. While there is great variety in the devices, they all must comply with the same set of regulations.
2. Use DSLs, where appropriate.
   - There was no DSL for the requirements specification, but the freedom of using Microsoft Office™tools for the artifacts enabled innovation and project speed.
   - The Teamcenter product includes a DSL facility called the BMIDE.
3. Integrate MDE with other efforts.
   - The modeling effort was extended to the whole project, not just the realization or code generation part.
4. Look for quick wins.
   - The quick wins in CS-1 and CS-2 were the visual modeling of most of the requirements and producing demo prototypes within a few weeks of starting.
   - The use of the Business Flow models resulting in fundamental business process improvements for both case study companies.
5. Know your people (the psychology of MDE).
   - On the development team side, the people were picked in part because of their experience and eagerness to try something new. Some of the people on the development team were not able to fully work at the abstract thinking levels that modeling requires. The people that had trouble with modeling naturally migrated and picked work in the iteration they were more comfortable with.
6. Key business drivers were explicit.
   - For the consulting company, the key business drivers are known and are listed in Sub-chapters 1.1 and 4.2
7. Know the domain.
   - The consulting company exclusively specialized in the medical device domain for many years.
8. Don't be a software company.

- The consulting company delivered more than just code, there were business strategies and software products as well.

9. Modeling is on the critical path (removes the optional use factor and ensures modeling needed for project success).
   - Modeling was used as much as possible through the whole project process. There was a great emphasis on modeling and people were required to model as much as possible.
10. MDE works well for multi-disciplinary systems.
    - The CS-1 and CS-2 work was not multi-disciplinary, so this success factor did not apply.
11. MDE and modeling works well for long-lived products (make MDE and modeling essential to the long-term success as engineering process documentation value).
    - The CS-1 and CS-2 solutions are intended to be in place for years.
    - The regulatory compliance factor increases the value of modeling and MDE beyond the initial solution generation.
12. Modeling and MDE works well when Solution quality is an important requirement (modeling enables a deeper understanding of the solution and MDE can automate solution generation which avoids human programming errors).
    - The medical device field has high quality standards enforced by regulations and audits.
13. Management buy-in to the MDE and modeling approach.
    - The consulting company and stakeholder management support the new methodology, which at that time was called the Paper Prototype Process.
14. Pick the right problem.
    - The problem to which the new methodology was applied was a good fit, as there was a possibility to try new things with a new customer.
15. Team communication is enhanced through models of what to build.
    - Team communication was through frequent meetings and explicit focus on the artifact communication quality.
16. Target Modeling and MDE on areas where informal modeling is already happening to build upon current trends and reduce organization change opposition.
    - The consulting company was using informal modeling for strategy work and was informally using UML for design work.
17. A methodology and modeling champion is needed to provide education and encouragement when team members encounter difficulties.

- The methodology and modeling champion was the author.

Several of these success factors work together. In the case studies associated with this thesis, the success factors were seen in the following ways. The factors 5, 13, and 17 worked together through the team education activities. From a consulting business standpoint, the factors of 1, 3, 4, 6, 7, 8, 9, and 14 were fulfilled and explicitly listed as business objectives. From a more technical side, the factors 2, 11, 12, 15, and 16 were satisfied. The only success factor not applicable was 10.

The focus of the research cited in [HWRK11], [HRW11], and [WHR14] was on the use of MDE as an approach for code generation. The SAMEM extends the MDE concepts to applying modeling for other project needs, especially requirements specification. The extension of MDE to earlier phases of the project places the burden on communication for multiple purposes. Additional success factors to the ones cited above are summarized in the lessons learned Table 4.7.

4.4.2.2 Failure Factors for MDE

The following list is of important failure factors when using MDE and modeling for the first time. Only one of these factors applied to the projects of the case studies used in this thesis.

A Use MDE to cover an entire industry.

- Each case study was a single company, although in the same domain. A general domain-level solution was not attempted.

B Rely too much on commercial tools.

- This is primarily targeted towards code generation tools and a dependence that might not match the needs of the domain, so that compromises are made in favor of the tool and not the solution needs. In the case studies, the tools were generic visual tools or the Teamcenter specific ones, so that there was freedom to put the solution needs first.

C Try to generate whole systems.

- This refers to trying to use a model to generate everything. There was never an attempt to generate a whole solution in the case studies, only specific sub-solutions were partially automated.

D Don't understand how gains can be offset elsewhere.

- The project effectiveness gains from using models and MDE do not necessarily enable corners to be cut elsewhere.

E Expect a return the first time MDE is applied.

- This was expected in CS-1 projects.

F Obsess about code generation/productivity.

- In both the case studies the focus was on modeling primarily for clear communication.

G Make changes to generated code.

- Neither CS-1 nor CS-2 involved generation of code, so there was no code to attempt to change.

H Have a strong requirement on efficiency of generated code but use MDE in any case.

- The focus was on making the organization more efficient by automating manual tasks and removing work that did not add business value.

I Don't allocate sufficient resources.

- There were enough development team people for all the work without overtaxing anyone.

J Use MDE as a way to remove your existing human expertise from the development process.

- The aim of the modeling and MDE approach was on innovative solutions that brought real value to the customer, not in minimizing the people involved and their costs.

K Apply MDE at the wrong level of abstraction.

- o This failure factor refers to trying to do the wrong things in a project with MDE, but in the case studies the focus was on modeling for communication.

L Don't support developers in MDE.

- The developers were supported with training as needed.

Factor E, expect first time return on investment, was the only one that applied to CS-1. In CS-2 the experiences of CS-1 provided confidence in the SAMEM approach. Despite E being a failure factor, the design and planning of the SAMEM along with years of team member experience mitigated this failure factor. In addition, the consulting company was willing to change approaches if the SAMEM showed signs of not working.

4.5 Applying the SAMEM Summary

This chapter has covered the experiences and lessons learned from applying the SAMEM, starting with an early version, in real industry settings. The practical experience and feedback helped to refine the foundational premises of the SAMEM, but the project successes support the basic ideas. In Sub-chapter 7.1 the results of a survey of project participants substantiate the claims.

The introduction of a new approach, process, or methodology into an organization is an organization change. Whether the change is successful depends on several factors:

- The business situation or environment around the change.
- The effectiveness of educating the participants.
- The willingness of the participants to try something new.
- The recognition of a situation that can be improved.
- The clarity of the rationale for the change and the proposed changes.

The business situation that the SAMEM was applied to is the management of product development information for medical device companies (CS-1 and CS-2). One unique aspect of the medical device development environment is the regulatory requirement for traceability through the product development process. The traceability requirement impacted the SAMEM through the additional number of artifacts created and maintained. The agile program development approaches of SCRUM [Coh10] and eXtreme Programming [Bec00] eschew the development of needless documentation. Traditional agile development believes the code alone is enough, but that is not acceptable for highly regulated domains such as medical devices. The impact on the SAMEM is to make sure that adequate documentation can be produced throughout the project process.

The education specifically about the SAMEM concerned both the development teams and the customer teams. The customer teams already knew their domain, but the development team needed education about specific medical domains and the regulatory requirements. Bringing developers into a new domain will always entail some domain education. Additional developer-specific education covered the SEFPs, UML and modeling, and the tool technology for the solution development. Both the customers and developers attended education sessions on the SAMEM project process components of RM-ODP, iterative & incremental, and the three-phase project structure. The education of the developers needed to be more comprehensive, as they needed to guide the customers through the new process approaches. While this went well in general, there is some survey feedback on areas for additional improvement (Sub-chapter 7.1.4).

The willingness to try a new process within the customer organizations was mixed. The research and development (R&D) people had little problem with the new ideas, especially the graphical expression of the requirements. R&D people are more comfortable with new and innovative approaches. The quality and regulatory compliance people at the customer were less comfortable until positive results were achieved. The developers showed more willingness in large part, because volunteers were sought out. However, for the CS-1 development team, the author had the authority to force the SAMEM to be followed with the full support of the management team. For the CS-2 project, the successes from CS-1 were documented and the developers were quite willing.

The management of the consulting company was very experienced in software solution development (on average more than 20 years). There was clear recognition that a better

approach to running a development project was needed and possible. The combination of agile approaches (iterative & incremental) along with models to improve communication was supported.

The rationale for the SAMEM was developed by the author of this thesis and is presented in a document format and presentation format. There are also a set of objects for the new approach that outline the desired achievements. The objectives are repeated in the high-level goals for this thesis (Sub-chapters 1.1 and 2.4.1). The description of the SAMEM (early version) explicitly contains the rationale for the choice.

Through applying the theoretical definition of the SAMEM in multiple industry projects, additional improvements were made. The SAMEM has proven itself to be a practical improvement, which is substantiated in Chapter 7.

Chapter 5

Methodology and Process

In this chapter, an architectural definition for the process flow components for the SAMEM is presented. The definition is a set of concepts and relationships between those concepts [GPHS08], [Jur15]. The concepts represent an organized but not necessarily linear progression from the initial observations with the stakeholders to a successful solution.

The first purpose of the definition is to ensure completeness of the SAMEM variants when adaptations are done to fit the SAMEM to a specific problem-solution situation. The second purpose of the concepts definition is to provide a framework that enables the business management of a SAMEM instance. The concepts can be checked for completeness to ensure that there are no missing components and that project artifacts are structurally sound.

The SAMEM concepts definition is similar to the purpose of the REAM [BPKR09] and the OMG SysML [OMG15a] standard. It is a set of artifacts with relationships. Each artifact in a SAMEM instance must map to either a concept of the SAMEM Process Model (SAMEM-PM) or a concept in the SAMEM Information Model (SAMEM-IM), which is defined in Chapter 6. In extending the SAMEM to a new domain, the SAMEM-PM and/or SAMEM-IM are extended for the new concepts.

5.1 The SAMEM Process Model (SAMEM-PM) Concepts

The core SAMEM process concepts are embodied in the ideas of multiple *phases* that consist of multiple *iterations*. Each iteration can produce or revise one or more *Artifacts*. A specialization of a phase is a *RM-ODP Viewpoint*, which embodies a level of abstraction. The process definition concepts and their relationships are shown in Figure 5.1 in pseudo-UML Class Model style. The RUP [PK00] and the SEMAT [SEM98] also have the concepts of ordered phases with multiple iterations.

5.1.1 The Core SAMEM Process Model Concepts

The *Methodology* concept consists of an ordered set of *Phase* concepts through a *Has* relationship. The *Phase* concept represents the progression of the project process through a major category of work. A *Phase* has a business *Goal* to achieve, which is indicated by the *Goal* concept and the *Achieve* relationship. The rationale for the *Phase* concept is that the project is a business investment with expected benefits and associated

risks. At least at the end of a *Phase*, good business management will review the costs, risks, benefits, and degree of *Goal* fulfillment to determine if moving forward is a responsible business action, which is business due diligence. The business management of the project *Phases* could create additional checkpoints within a *Phase* to evaluate progress and project continuation. The additional checkpoints are *Sub-Goals*, which can be viewed as a specialization of a *Goal*.

The *Phases* are ordered by the *Follow-on* relationship, which indicates that starting on the following phase is conditional on some minimum amount of work accomplished in the preceding phase. There is no requirement that the preceding phase be totally complete before starting any work in the following phase. As an example, if the preceding phase is requirements elicitation and the following phase is engineering design, then not all requirements must be specified before some design work can start.

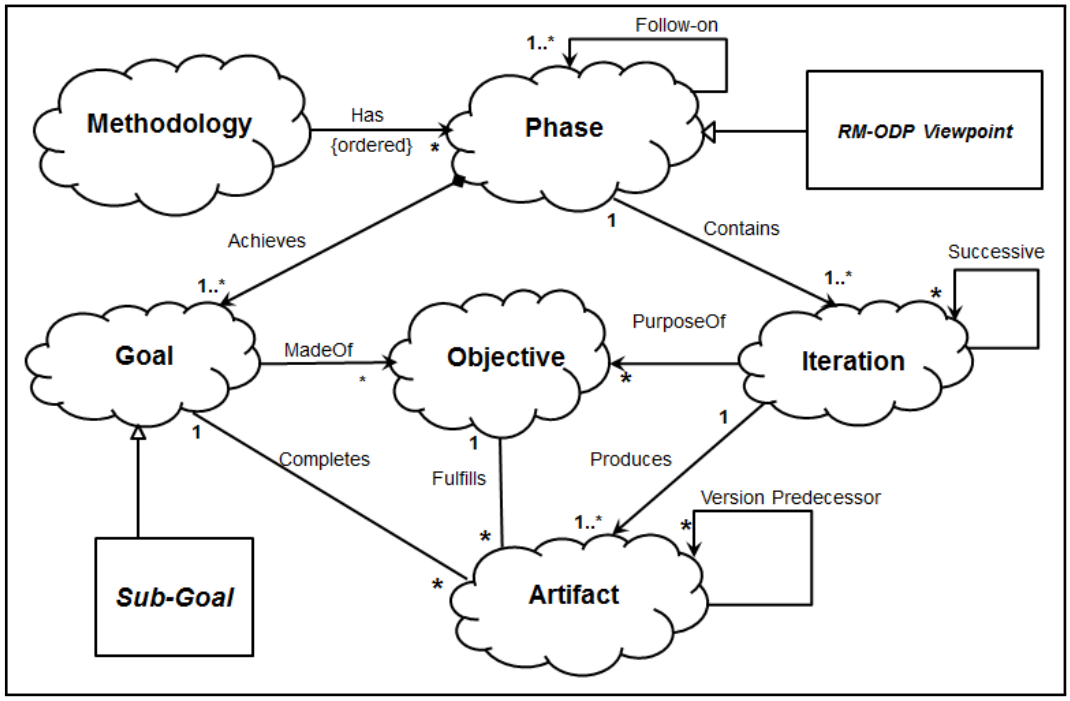


Figure 5.1: The SAMEM Process Concepts.

A specialization of the *Phase* concept is the *RM-ODP Viewpoint* concept as defined in the RM-ODP. Specifically for the SAMEM, the *Phase* concept aligns with the RM-ODP abstraction viewpoints, although other domains can have different *Phase* concept mappings. The *RM-ODP Viewpoint* concept represents an abstraction level. At the beginning of a project, the abstraction level is high, because the problem is vaguely understood and many details remain hidden or undecided [Bro10], i.e. the Enterprise Viewpoint. As the project proceeds through the *Phases*, the problem is more completely understood and design decisions are made, which moves the abstraction level to a more precise position, i.e. Informational Viewpoint (*Phase*), Engineering Viewpoint (*Phase*),

etc.

The ordering of the *Phases* from an abstract description of the problem to a more specific expression of the problem/solution provides several benefits to the SAMEM. The primary benefit from ordering the *Phases* is guidance on doing the work in the most effective manner. The ordering separates specifying the "What" from the decisions specifying the 'How." The *Goals* from the early *Phases* and the *Objectives* from the associated *Iterations* produce project management *Artifacts*. The project management *Artifact* is available to be communicated, updated, and checked-off as accomplished. Project introspection for improvement and management purposes is facilitated via the project management *Artifacts*.

A *Phase* has at least one and often several *Iterations* of work, which are connected by the *Contains* relationship. A *Successive* relationship orders the *Iterations*. The ordering of the *Iterations* allows for development traceability and process improvement through analysis of the effectiveness of the order of work accomplishment. Through understanding of the ordering of the work, process weaknesses leading to design or implementation mistakes can be identified. An *Iteration* is a bounded and well-defined unit of work that can be measured in terms of output. Each *Iteration* has one or more *Objectives*, which are defined with the relationship *PurposeOf* connecting the *Iteration* concept with the *Objective* concept. The work output of the *Iteration* is some type of project *Artifact*, such as plans for the next iteration, requirements, designs, technology realization (code), or documentation. The *Produces* relationship connects the possibly multiple *Artifacts* to the *Iteration*. Details of the *Artifact* possibilities are described in Chapter 6.

There is no specific limit on the number of *Iterations* in a *Phase*; rather, the focus is on having enough *Iteration* instances to achieve the *Phase Goal* or a *Sub-Goal*. The *Goal* of the *Phase* is partitioned into a smaller number of *Iteration Objectives*, which are more manageable and measurable. The partitioning of the *Goal* concept is defined by *MadeOf* relationships to one or more of the *Objective* concepts. The methodology *Iteration* concept is essentially the same as the sprint in the SCRUM approach, while the *Objective* is mapped to the SCRUM User Story concept.

In Figure 5.1, the *Artifact* concept represents a piece of work needed for the solution development. The *Artifact* concept in Figure 5.1 is a link to the SAMEM Information Concepts, which are shown in Figure 6.1 in Chapter 6. An *Iteration* can produce multiple *Artifacts* if needed to satisfy its *Objective* through the *Fulfills* relationship; for example, several requirements can be produced from a stakeholder interview *Iteration*. An *Artifact* is not necessarily complete and correct upon initial creation but moves toward completeness and correctness as it is reviewed in additional *Iterations*.

An *Iteration* can be devoted to *Artifact* evaluation and can produce an improved version. The improved *Artifact* versions are connected via the *Version Predecessor* relationship, so that the evolution can be understood. It is possible that upon evaluation an *Artifact* is deemed totally incorrect and that it should be removed or given a *State* value of *Obsolete*. Chapter 6 contains a more detailed definition of the methodology artifacts and the *Artifact* states. Sub-chapter 5.1.3 describes the process management

state concepts.

5.1.2 The SAMEM Process Model Concepts Mapped to Abstract Objects

The *Phase*, *Goal*, *Iteration*, *Objective*, and *Artifact* concepts from Figure 5.1 can be viewed as abstract classes and need realization in corresponding implementing classes. The implementing classes are the process management *Artifacts* as mentioned in the previous sub-chapter. An example of the realization expressed in pseudo-UML is shown in Figure 5.2. Each of the abstract concepts has a realizing *Abstract Class* with the base attributes of *Name* and *Identifier*. The *Name* attribute is for a user-convenient reference and the *Identifier* attribute is for a unique system identifier for each instance. The attributes are deliberately typed as String to allow for the most general range of values. Further domain-specific refinements are made from the abstract classes in Figure 5.2.

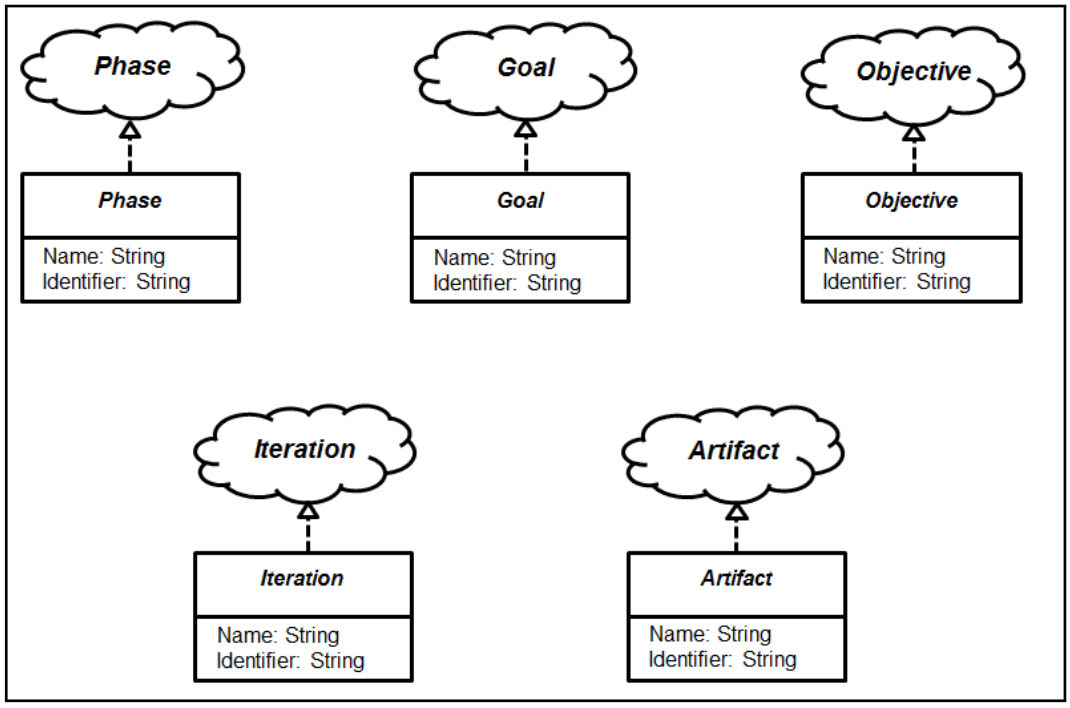


Figure 5.2: Process Concepts Realized as Abstract Objects.

5.1.3 Instance State Guidance

The project management *Artifacts* of the realizations of *Phase*, *Goal*, *Iteration*, *Objective*, and *Artifact* abstract classes from Figure 5.2 all have an associated *State*, as shown in Figure 5.3. Each of the *State* objects has a *Value* attribute which has the general type

of String to be flexible. The project management *Artifact* state concept is identical to the SEMAT state concept, as described in Sub-chapter 7.4.1.1 and shown in Figure 7.9.

The *PhaseState* value indicates the state of completion of the *Phase* relative to the related (*Achieves*) *Goal*. The *Phase* is related to its *PhaseState* via the *Completeness* composition relationship. Typical basic state values for a *PhaseState* are *Not Started*, *In Progress*, and *Complete*, although specific projects can extend the basic values. The *Compete* state applies to achieving all related *Goals*. Depending on the size of the project, the time needed to complete the work of a *Phase* can easily be weeks, if not months.

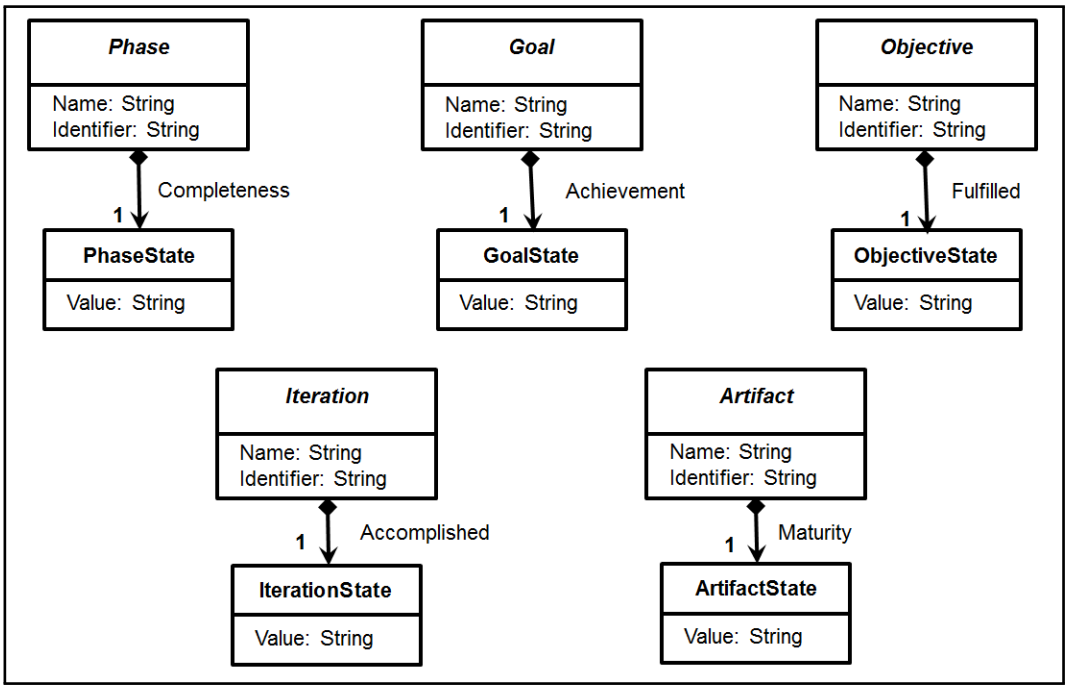


Figure 5.3: Process Instance States.

A *Goal* is connected to its *GoalState* through the *Achievement* composition association. The *GoalState* values indicate whether the *Goal* has been achieved. Typical basic state values are *Not Achieved*, *Partial Achievement*, and *Achieved*, which can be extended.

The *Iteration* has an associated *IterationState* connected by the *Accomplished* composition relationship. Typical basic values for the *IterationState* are *Not Started*, *Started*, *In Progress*, and *Finished*. For the *Iteration* to be in the *Finished* state, all of the associated *Artifact* objects would need to be *Approved*.

The *ObjectiveState* is connected via the *Fulfilled* composition relationship with the *Objective* object. Possible values for the *ObjectiveState* are *Not Started*, *Started*, *Partially Fulfilled*, and *Fulfilled*. The *Not Started* and *Started* values are useful for project management oversight and control. For example, an *Objective* that has a state value of *Partially Fulfilled* or *Fulfilled*, implies that the associated *Goal GoalState* has a value of

Partial Achievement. The *Objective* is *Fulfilled* when all of the necessary *Artifact* objects are *Done* and the *Iteration* object is *Finished*.

The *Artifact* has an *ArtifactState* which reflects its trustworthiness, completeness, and correctness at any point in the project process. Each *Artifact* is connected to a single *ArtifactState* via a *Maturity* relationship. The *ArtifactState* value indicates whether the *Artifact* can or should be used. For example, a newly created *Artifact* will have the *ArtifactState* value of *New* and after an *Iteration* with the *Objective* of review artifacts, the *ArtifactState* value changes to *Approved*. The SAMEM does not demand any specific *State* values, but rather offers the guidance that the *State* values should reflect the maturity or trustworthiness of the *Artifact* object for project use, such as *New*, *Under Review*, and *Approved*.

5.1.4 Enabling Project Management

As stated earlier, the second purpose of the concepts definition is to provide a framework that enables the business management of a SAMEM instance. Figure 5.1 shows the set of the SAMEM process concepts with the relationships between them. Each of the concepts has an instance type that corresponds to real project *Artifacts* for the *Phases*, *Goals (Sub-Goals)*, *Objectives*, and *Iterations*. In a SAMEM-guided project, there are real *Artifacts* instances of the *Phase* definition, the *Goals* for a *Phase*, the *Iterations* definitions, and the *Objectives* for an *Iteration*, which hold the project-created data and real links between the *Artifacts*.

The concept definitions allow the project management tools to check on the integrity of the SAMEM-guided project. Examples of questions that a manager could ask of the project management artifacts to ensure the project is under control and proceeding as planned are:

- Does every *Phase* have at least one *Goal*?
- Do the planned *Phases* make logical work order sense?
- Does every *Goal* have at least one measurable *Objective*?
- Does every *Phase* have at least one planned *Iteration*?
- Will the planned *Iteration* achieve the *Objective*?
- Are the *Artifacts* linked to the *Iteration* and *Objective* appropriate?

There are other project management questions that can be asked to control the day-to-day project progress:

- How many *Artifacts* have been created?
- How many *Artifacts* are “Approved?”
- Have all the *Goals* for a *Phase* been “Fulfilled?”

- How long did it take for the *Objectives* for an *Iteration* to be completed?
- How long did it take for a *Phase* to be completed?
- How long did it take for the project to be completed?

The second set of project management questions requires an extension to the *Attributes* shown in Figure 5.3. In Figure 5.4 below, a UML Class Model shows some of the attributes needed to answer the second set of questions. Depending on the level of control desired by the project manager, additional attributes can be created to answer the management questions. The attribute meanings and their behavior are the same for all of the *Instance* classes, so that an abstract class *AbstractInstance* can be created through the generalization relationship. For management needs, the attributes of *Content*, *Date_Created*, and *Person_Responsible* are added to the *AbstractInstance* class. Likewise, the *AbstractState* class provides a generalization of the common attributes of the *State* classes. The *Change_Date* attribute is added to *AbstractState* for management purposes.

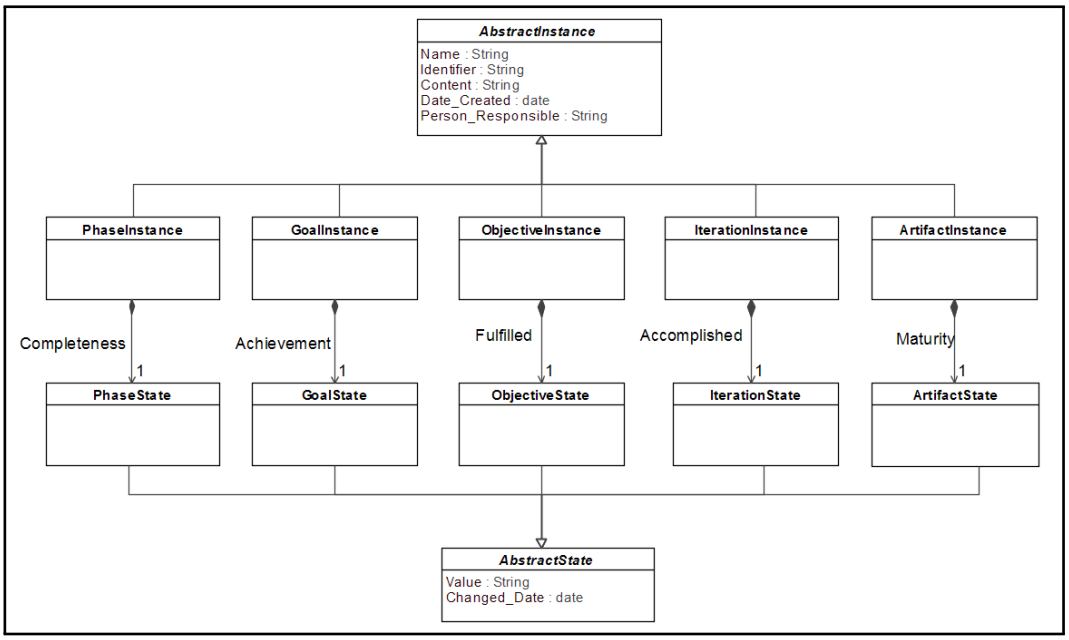


Figure 5.4: The SAMEM Process Model of Concepts in UML.

5.2 Applying the SAMEM-PM: Instance Examples

The SAMEM-PM provides a framework for the process instances that appear within the SAMEM definition. This sub-chapter describes some of the possible examples.

5.2.1 Methodology – Phase – Viewpoint Concept Examples

In Figure 5.5, an example of instances of the concepts (see Figure 5.2) and their structural relationships are shown. The primary objectives of concept structuring are project risk mitigation and due diligence support. The SAMEM is an instance of *Methodology*. The SAMEM:Methodology instance has a structure of five *RM-ODP Viewpoint* instances labeled: Enterprise, Information, Computational (Behavior), Engineering, and Technology. The *RM-ODP Viewpoint* instances are ordered through the *Follow-on* relationship from the highest level of abstraction, which is the Enterprise Viewpoint, to the lowest level of abstraction, which is the Technology Viewpoint. The Enterprise Viewpoint can be followed by either the Information or Behavior Viewpoint. The specific Viewpoint, Information or Behavior, connected by *Follow-on* relationship from the Enterprise Viewpoint allows for domain adjustments, such as a domain where information is the primary driver for requirements or a domain where behavior, such as a business process, is the primary requirements driver. If the Information Viewpoint instance follows the Enterprise Viewpoint, becoming the Second Highest, then the Behavior Viewpoint will become the Third Highest and will follow the Information Viewpoint and vice versa. The Engineering Viewpoint follows the work done in the Information and Behavior Viewpoints, which is followed by the Technology Viewpoint.

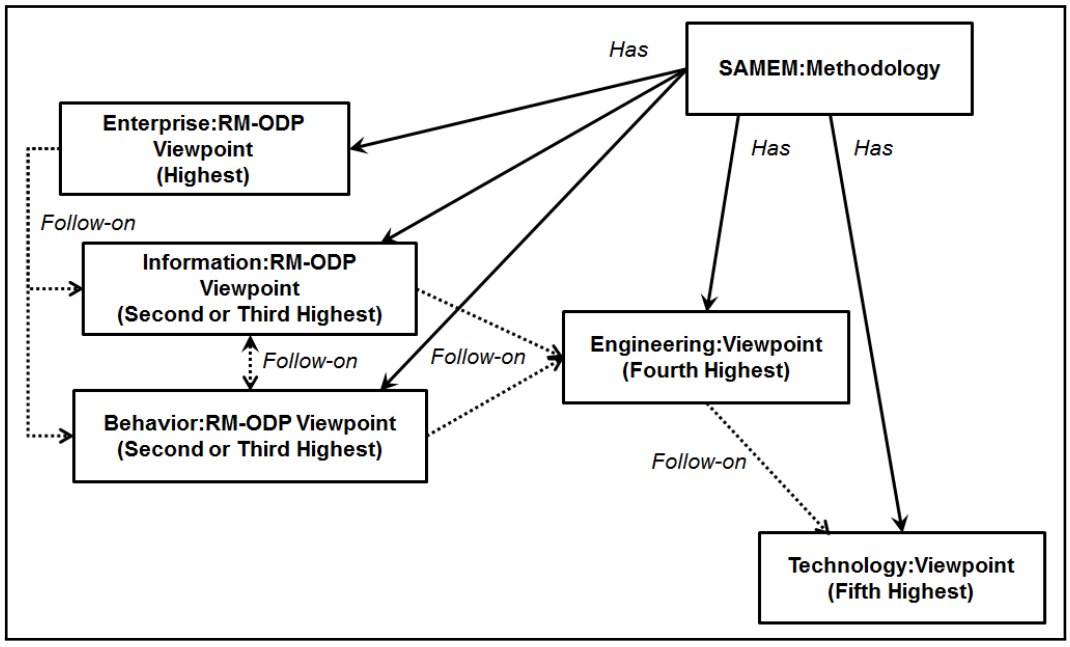


Figure 5.5: Methodology - Viewpoint Instance Relationship Example.

The rationale for the structure created by the *Follow-on* relationships in Figure 5.5 is explained in Figure 5.6. The process metamodel shown in Figure 5.2 indicates that each *Phase* has an attribute of a *Goal*. The *Phase Goals* for the viewpoints are shown in the ovals within Figure 5.6 rather than as a related class. The alternate graphical

form of Figure 5.6 is an application of the *Change Language SEFP* as a break from the UML Class Model. Figure 5.6 shows the flow down the hierarchy structure (through the *Follow-on* relationships) from the most abstract level of Business Goals to the most specific level of Technology Design & Implementation Goals.

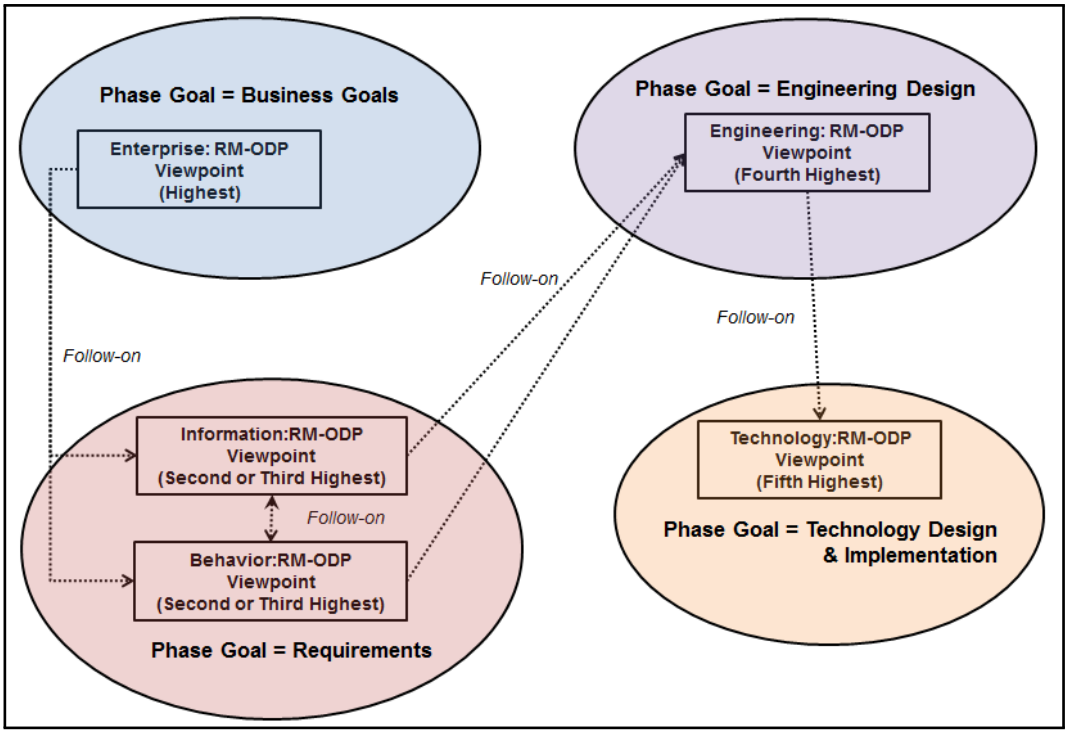


Figure 5.6: Phase Goal Explanations Example.

The Enterprise Viewpoint instance has the Phase Goal of eliciting and verifying the Business Goals of the project. Eliciting and verifying the Requirements is the Phase Goal for both the Information and Behavior Viewpoints. Requirements are defined in the context of the Business Goals and an important aspect of verifying the Requirements is to ensure that they are consistent with the Business Goals. The Business Goal discovery does not need to be complete before Requirements are generated and verified. The Phase Goal of Engineering Design is the production of solution designs at the engineering level, which depends on having some Requirements. The Engineering Designs are verified as to how well the Requirement is fulfilled by the design. The Phase Goal of Technology Design and Implementation follows from the Engineering Design work, since the choice of specific technologies depends on Engineering Design decisions.

5.2.2 Viewpoint – Iteration Concept Examples

Figure 5.2 shows that each *Phase* contains multiple *Iterations*. The *Iterations* have finer grained *Objectives* that combine to fulfill the *Phase Goal*. There is some flexibility in how

the *Iterations* work to accommodate adaption to multiple domains. Figure 5.7 shows an example of a common pattern of *Instances* from the SAMEM-PM metamodel using the *Phase* specialization of Enterprise Viewpoint as the start.

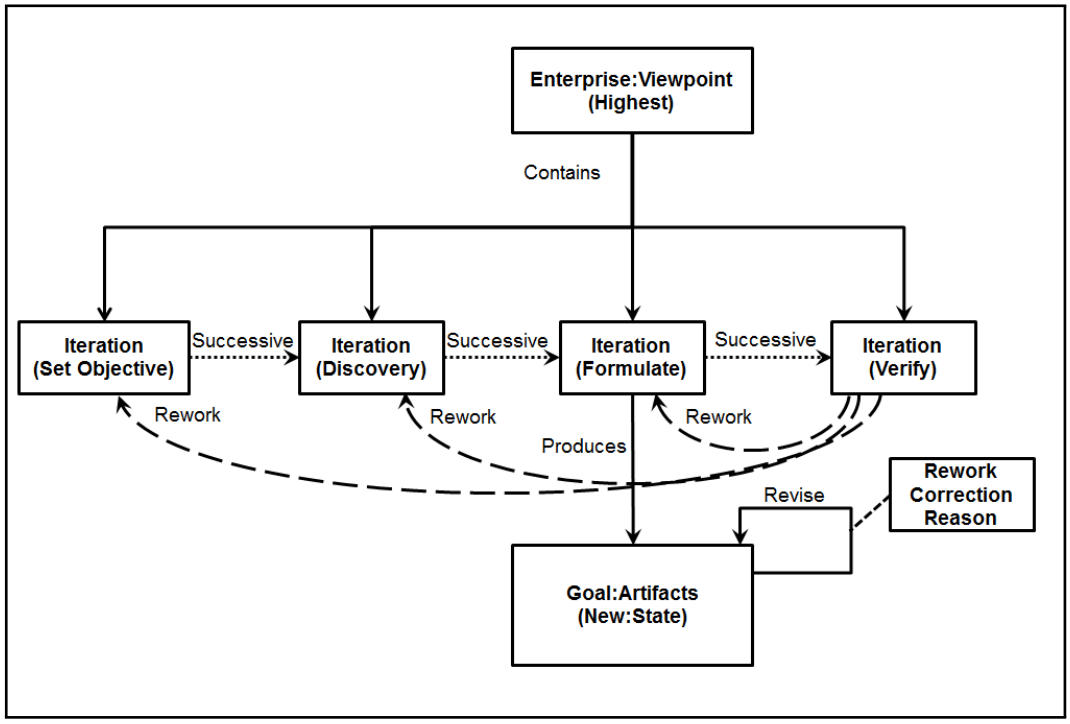


Figure 5.7: Viewpoint, Iteration Instance Example.

The Enterprise Viewpoint has *Contains* relations to four *Iteration* instances with the *Objectives* of Set Objective, Discovery, Formulate, and Verify. The Set Objective *Iteration* is a breakdown into smaller units of work of the *Phase Goal*, which in this example is to establish the solution Business Goals. The succeeding *Iteration*, Discovery, covers the work of discovering what the objective is about, which is often a kind of elicitation activity. Following the Discovery *Iteration* work, the knowledge gained is *Formulated*, in this case as a Business Goal, and is documented in an *Artifact*. The *Artifact*, and there can be multiple, is the primary input to the Verify *Iteration*, where the *Artifact* is checked against SEFPs and with the appropriate stakeholders.

5.2.3 Iteration – Artifact Concept Examples

There are several new relationships and items shown in Figure 5.7. The major new relationship is *Rework*, which is shown as a large dashed line. This relationship acknowledges the possibility that the *Artifact Verification Iteration* fails one or more of the tests. Depending on the failure, a new *Iteration* for the same purpose is done at some point earlier in the *Successive* chain. For example, if the *Artifact* is poorly formulated, which would be seen through a communication difficulty, then a new version is created with an

improved formulation. If the *Discovery* work were poorly done, then the *Rework* relationship would go back to that type of work. The *Artifact* versions are related through the *Revise* relationship, which contains the rationale, *Rework Correction Reason*, for the new version. The *Rework Correction Reason* is optional; however, there are domains, such as Lifesciences, where the regulations require traceability through design improvement decisions. The pattern of *Iterations* connected by *Successive* relationships can be applied to the other *Viewpoint* instances.

5.3 An Activity Model Example for the SAMEM-PM Flow

The illustrations in Figure 5.2, Figure 5.5, Figure 5.6, and Figure 5.7 describe the components or concept artifacts of the SAMEM-PM. The other perspective is the behavior flow of a project process that results in the creation of the SAMEM-PM artifacts. In Figure 5.8 ,an example of project process modeled with the UML Activity Model capabilities is shown. This example process is one possible approach which can be used as a starting point for further refinement and extension.

Figure 5.8 is the project process that describes the basic cycles of working with the SAMEM Process Concepts at an abstract level. In Sub-chapter 3.5.3, the definition and purpose of the *Business Flow* are described. The project process defined in Figure 5.8 is the *Business Flow* for using the SAMEM and its process concepts.

The project process starts with an initial task of *Choose Viewpoint or Phase*. The phases or viewpoints can be adjusted to the project or domain needs, but for this example the use of RM-ODP viewpoints are assumed, which means that the initial viewpoint is the RM-ODP Enterprise Viewpoint. With the viewpoint set, the next task is *Goal Elicitation*. The output of *Goal Elicitation* is a number of Business Goals, *Phase Goals* for the project, which are gathered from the various stakeholders. The *Phase Goals* are incrementally discovered and collected as the work loops through the process, since it is virtually impossible to know all the Goals of a large project beforehand [Bro10].

As soon as there are a sufficient number of *Goals*, the *Iteration Creation* task is initiated. Note that initiation of the *Iteration Creation* task does not depend on having all *Phase Goals* defined and that a sufficient number is determined by the project managers. The *Iteration Creation* task creates an *Iteration Objective Definition* task for each *Phase Goal*, which allows a set of *Goals* to be processed in parallel. The set of *Iteration* tasks is shown through the *Iteration 1*, *Iteration 2*, and *Iteration N* symbols. The work of the *Iteration Objective Definition* task is to formulate the *Phase Goal* into smaller, more accurate, and precise representations and then store them as one or more *Object Artifacts*. Just as with the *Phase Goals*, the *Objective Artifacts* are incrementally discovered and collected during the process execution.

Upon completion of the definition, the *Objective Verification* task is started and uses the corresponding *Objective Artifact*. The purpose of the *Objective Verification* task is to use the appropriate means to check, test, or measure the *Objective Artifact* to assure that it is accurate and correct with respect to the current level of abstraction. There are two main results of the verification activity, at least as far as the detail of this example

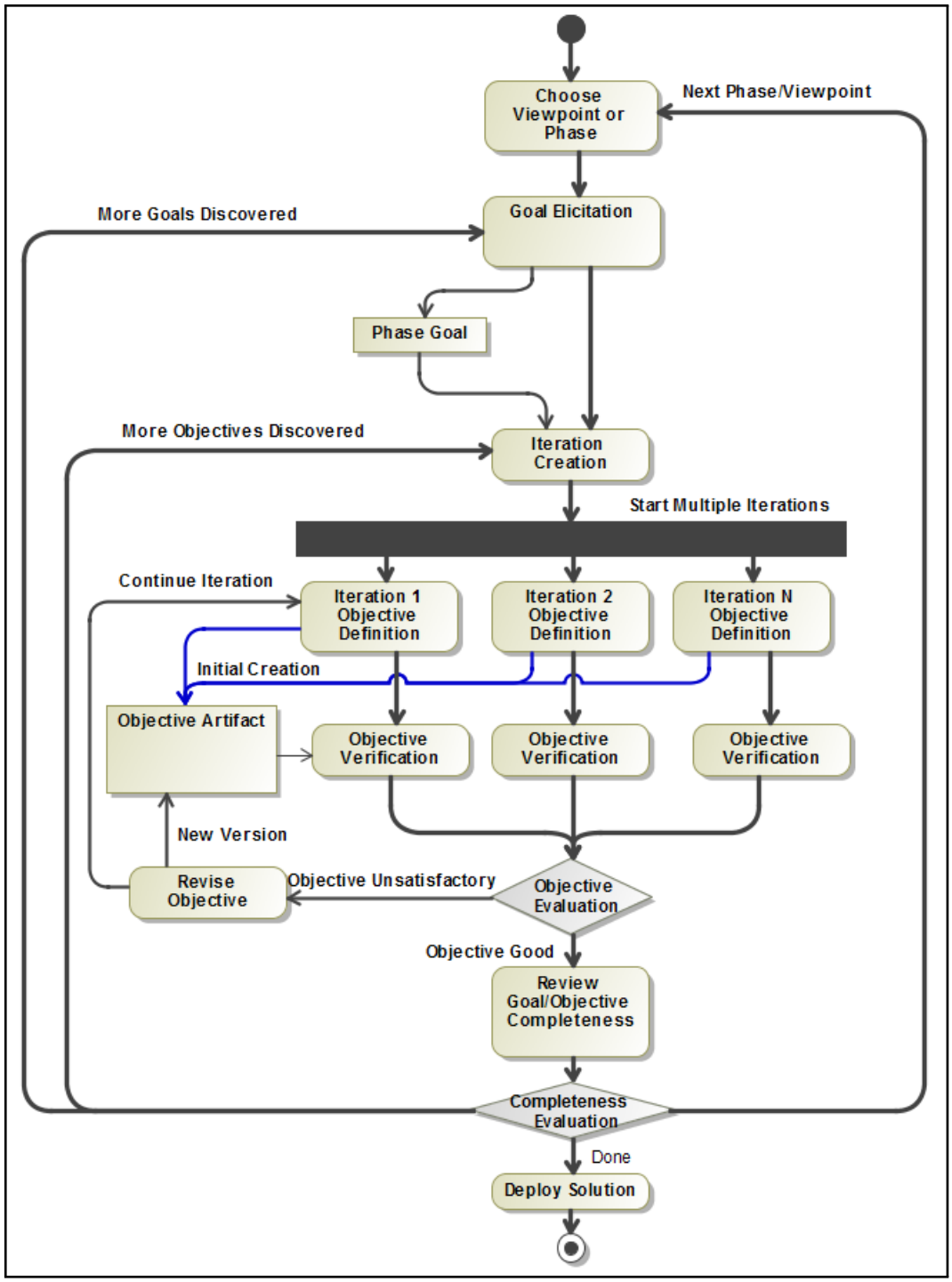


Figure 5.8: The SAMEM Project Process Example.

is concerned, the *Objective Artifact* is good or it is deficient and requires improvement resulting in a new version. If a new version is needed, the work of the iteration is continued. In practice, there are more options, such as discovery that the *Objective* or *Goal* is a mistake and should be completely discarded; however, this level of detail has been left out for readability.

As *Phase Goals* and *Objective Artifacts* are completed, the *Review Goal/Objective Completeness* task is initiated. Through three rework arcs, a return to earlier tasks in the process can occur. The review might come to the conclusion that some new *Objectives* have been discovered, resulting in activating a new instance of the *Iteration Creation* task via the *More Objectives Discovered* arc. A second review result is that a new *Phase Goal* has been uncovered which causes a new instance of the *Goal Elicitation* task via the *More Goals Discovered* arc. A third review conclusion is that all or enough of the work of the *Phase* or *Viewpoint* is done, so that the next *Phase* can be started with a new instance of the *Choose Viewpoint or Phase* task via the *Next Phase/Viewpoint* arc. The final result of the *Review Goal/Objective Completeness* task is that all work of the final *Phase* is complete and the solution is ready for deployment.

It is clear that each task in Figure 5.8 can be partitioned into finer grained tasks for improved work distribution or parallelization. In practice, there is more additional work in the management of the two simple artifact objects, *Phase Goal* and *Object Artifact*, than is shown in this simple example process. To keep the example process of Figure 5.8 as readable as possible, swimlanes are also left off, as at this level of abstraction the task flow is more important than who does the work.

5.4 The SAMEM-PM Concepts Summary

This chapter sets forth the concepts that are the foundation of the SAMEM Process Model. The base concepts of *Phase*, *Goal*, *Iteration*, *Objective*, and *Artifact* along with the relationships between them enable the verification of the project process. Missing instances of the base concepts indicate that a breakdown in the *Methodology* (SAMEM) is occurring. The project management concerns over breakdowns are important indicators for poor workmanship or that important tasks are not completed.

The base concepts also enable domain specializations. Figure 5.2 shows a domain specialization point via the *Sub-Goal* concept. For example, the CS-1 and CS-2 medical domain case studies have the *Sub-Goal* of compliance with the FDA regulation of 21 CFR §820.30. Other *Sub-Goal* domain specializations can include user Interface standards, coding standards, or test coverage.

Independent of the domain specializations, the base SAMEM Process Model concepts provide the business foundation needed to show due diligence in planning and execution of the project process.

Chapter 6

The Language Used for Information Models in the Projects

The information model concepts for the SAMEM (SAMEM-IM) are specified in this chapter. The information concepts are the basis for the *Artifacts* collected and/or updated during a SAMEM increment. As for the SAMEM-PM described in Chapter 5, the REAM from [BPKR09] and the OMG SysML [OMG15a] provide analogous ideas for the SAMEM-IM.

The SAMEM-IM concept definitions provide a context for completeness checks on a SAMEM instance and a basis for extensions. For this purpose, the techniques of a meta-definition [GPHS08] are used. The SAMEM *Artifact* instances produced in a project process *Increment*, as described in Chapter 5, will map to a concept in the SAMEM-IM.

An important aspect of the SAMEM-IM is that the *Artifact* concept has a *State* related to it. The *State* reflects the stability or trustworthiness of the information *Artifact* on its progression from newly discovered, through reviewed, and onto approved. The state machine of the SAMEM-IM is a basic model that can be extended. The *State* value is the connection point and gating criterion between the major *Phase* or *Iteration* abstraction levels of the SAMEM-PM. To move forward to the next major investment in work, a set of information *Artifacts* should be in the approved or equivalent state.

The information model *Artifacts* are used for communication with the stakeholders, the project team, and as input into a verification tool. This requires that each information *Artifact* has one or more communication formats associated with it. Multiple formats are necessary, for example a graphical format for human communication and perhaps an XML format for input to a verification tool.

6.1 The SAMEM-IM Concept Model

Figure 6.1 shows the SAMEM-IM model concepts. The core concept consists of the *Artifact*. For traceability purposes, each *Artifact* instance can be related to an *Iteration* (see Figure 5.1 and Sub-chapter 5.2.3) which created or modified it through the *Produces* relationship. Creating an instance of the *Produces* relationship is domain dependent; for example, in the Life Sciences domain regulations require this type of traceability, while other domains do not. From a software engineering process perspective, having the *Produces* relationship can help with project process improvements. Tracing an incorrect

Artifact to the *Iteration* that produced it, then to the *Objective* it *Fulfills* can help in analyzing a project process for organization or execution weaknesses.

The *Artifact* concept is the one that ties together the SAMEM-PM concepts and the SAMEM-IM concepts. The SAMEM-PM focuses on the process of creating or updating the *Artifact* instance (see Figure 5.2, Figure 5.7, and Sub-chapter 5.2.3), while abstracting away the details. The rest of this chapter defines the *Artifact* concept in detail.

6.1.1 The SAMEM-IM Artifact Concept Definition

The *Artifact* concept has two self-referencing relationships, *Version Predecessor* and *Design Evolution*. Within the scope of a single *Iteration*, the *Version Predecessor* relationship tracks the improvement and refinement of the *Artifact* instance after an unsatisfactory *Objective Verification* task (see Figure 5.8). The purpose of the *Design Evolution* relationship is to relate the different *Artifact* instances as the idea moves through the abstraction levels [Dic05], [Col05], [RvdHMRM04]. For example, a *Design Evolution* relationship would link a requirement *Artifact* at higher abstraction level to a design alternative *Artifact* which is at a lower abstraction level.

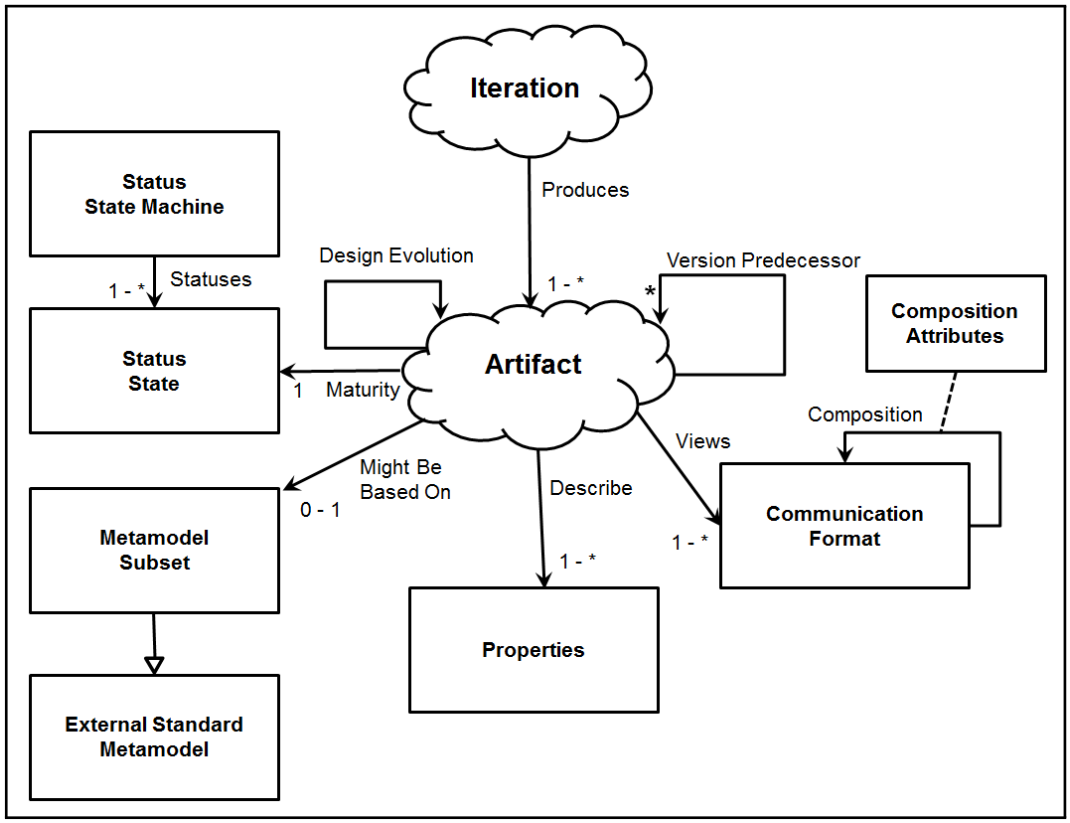


Figure 6.1: The SAMEM-IM Concepts.

6.1.1.1 Another Artifact Model

In [GHR17], a sustainable Artifact Model (AM) tailored for generator-based MDD projects is described. The *Artifacts* in the SAMEM-IM are consistent with the AM, albeit on a more general level. The AM makes restrictions consistent with managing the generated artifacts, their associated sources, and the complex relationships between the two. The SAMEM-IM *Artifact* has two relationships, *Design Evolution* and *Version Predecessor* that can be seen as specializations of the *refersTo* relationship in Figure 2.1 in [GHR17]. The *Artifact* abstract class in the AM has specializations for supporting the generation process, which are unnecessary for the SAMEM-IM.

Another difference that reflects the divergent, but not incompatible purpose is the orientation of the SAMEM-IM towards visual models for human communication in contrast to the AM emphasis on text for automated tool use. The association of a *Status State* to the SAMEM-IM *Artifact* for project process tracking is another not incompatible difference.

6.1.2 The SAMEM-IM Communication Format Concept Definition

Related to the *Artifact* concept by a *Views* relationship are one or more *Communication Format* instances. An *Artifact* needs to be expressed in at least one format in order for it to be reviewed or verified. The creation of the *Communication Format* instance happens in either the *Goal Elicitation* task or the *Iteration Objective Definition* task, as shown in Figure 5.2. An *Artifact* can have or participate in multiple *Communication Format* instances. A *Communication Format* instance can be a composition of multiple other *Communication Format* instances as represented by the *Composition* relationship. The SOD, as described in Sub-chapter 3.5.1.1, is an example of a composition *Communication Format*. The *Composition* relationship has a UML-like Association Class [OMG15b], called the *Composition Attributes*. The purpose of the *Composition Attributes* is to hold data about how the composition of the formats is structured, for example, relative locations of each format or the format of separation borders.

In Sub-chapter 3.5.2.1, examples from the Case Studies of various Information Model *Artifacts* and their *Communication Format* can be found. In Figure 3.10 through Figure 3.14 multiple *Communication Format* examples for *Artifacts* produced in the *Information Viewpoint* are shown. There are two different formats seen:

- Figure 3.5 is a SOD example of a *Communication Format* composition. The SOD consists of a text block, a state machine visual image, an information model visual image, and a Business Flow visual image composed into one format instance.

- Figure 3.10 shows a visual image of the basic information units for a FDA regulatory submission solution along with some of the primary attributes.

- Figure 3.11 is a text table that lists the attribute details for an Information Model.

- Figure 3.12 shows an example of an *Engineering Spec Item* with the most important four attributes and its first *Engineering Spec Revision* with its four most important attributes.
- Figure 3.13 is an example of a simple *Engineering Project* structure. The structure is the Information Model.
- Figure 3.14 shows where the structural information units that are used to build the parent-child relations are defined, so that they can be either a parent or a child. The example displays a *Main Project* with three *Sub-Projects* in the structure.

Figure 6.2 shows an example of how the SOD composition is realized at the SAMEM-IM concept level. There is an *Artifact* for the complete SOD and four participant *Artifacts* that compose the SOD. Each *Artifact* has a *View* associated *Communication Format*. Through *Composition* relationships, the *Communication Formats* of the participating SOD Notes, the State Machine, the Information Model, and the Business Flow are assembled into the *Communication Format* of the SOD. The *Composition Attributes* of each *Composition* relationship contain the data needed to reliably display the SOD *Communication Format* each time it is needed. A tool-independent set of display data for the *Composition Attributes* is possible as seen in the Object Management Group Diagram Definition Specification [OMG12a].

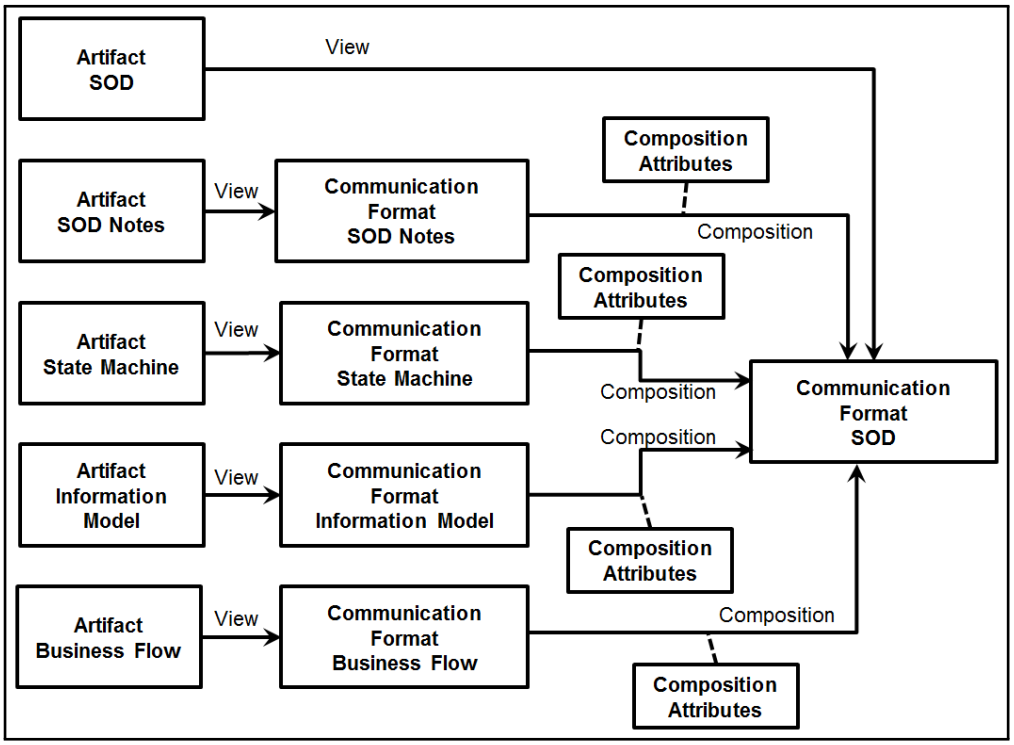


Figure 6.2: The SAMEM-IM Example, SOD Composition.

6.1.3 The SAMEM-IM Properties Concept Definition

Each *Artifact* has a set of *Properties* connected through the *Describes* relationship. The *Properties* can and will vary from domain to domain. One set or class of *Properties* that will almost always be associated is that used for *Artifact* management and control. The management or control properties will usually contain at least the following data and possibly more:

- Creation date and time – for evolution management.
- Creating user/person – for evolution management and responsibility traceability.
- Last modified date and time – for evolution management.
- Modifying user/person – for evolution management and responsibility traceability.
- Owning user/person – for possible access control.
- Unique identifier – for traceability and instance distinguishability.
- Name in a person-friendly format – for communication.
- Description – for communication.

6.1.4 The SAMEM-IM External Standard Metamodel Concept Definition

There are many modeling possibilities for *Artifacts* that exist, such as UML [OMG15b], URN [URN12], SysML [OMG15a], Domain Specific Modeling Language (DSML) (See [CFJ+17] for examples), and various programming languages, especially object-oriented ones such as Java or C++. General software modeling languages such as UML and some DSMLs operate at a higher level of abstraction than programming languages such as Java [CFJ+17], [Rum17].

When possible, it helps to examine whether an existing modeling paradigm, *External Standard Metamodel*, makes sense for a particular problem. There are several advantages for this such as existing knowledge amongst the team members, a well-defined syntax and semantics for communication clarity, and the availability of tools to support the generation of the formats. When applicable, a *Metamodel Subset* of the *External Standard Metamodel* might make sense. An example from the case studies is the use of a subset of the UML Class Model definition for the Information Model units, as shown in Figure 3.10 and described in Sub-chapter 3.5.2.1. The use of an *External Standard Metamodel* can also supply associated *Communication Format* suggestions.

6.1.5 The SAMEM-IM Status State Concept Definition

The final concept of the SAMEM-IM is the *Status State*. Each *Artifact* has a *Status State* value, connected by the *Maturity* relationship, which reflects the confidence or trust in the accuracy, correctness, completeness, and truth of the *Artifact*. The set of

Status State values are organized into a *Status State Machine*. The *Artifact* can have only one *Status State* value at any point in time.

When there is a need to change *Status State* value according to the *Status State Machine* definition, the *Maturity* relationship from the *Artifact* to the current *Status State* value is deleted and a new *Maturity* relationship to the new *Status State* value is created. The *Artifact* moves through the tasks in the SAMEM Project Process Example, as shown in Figure 5.8. In particular, the *Artifact* is created in the *Iteration Objective Definition* task, then reviewed in the *Objective Verification* task, and finally a decision is made as to done or not in the *Objective Evaluation* task. As the *Artifact* moves through the tasks, the *Status State* value is changed, through redirecting the *Maturity* relationship, to show what work has been accomplished on the *Artifact*. In the simplest form, the *Status State* Machine has the value of *Exists*. An example of a *Status State Machine* from the case studies can be seen in Figure 3.7.

6.2 Applying the SAMEM Metamodel Composition Concept to UML

One of the novel concepts in this research is the idea of the Solution Overview Drawing (SOD), see Sub-chapter 3.5.1.1, Table 7.3 in Sub-chapter 7.1.2, and for examples: Figure 3.5, Figure 3.6, Figure 3.7, Figure 3.8, and Figure 3.9. The composition of models to improve the communication of the requirements and the design can be applied to UML [OMG15b] for similar benefits.

For documentation communication benefits, the Object Management Group Diagram Definition Specification [OMG12a] can be extended to show multiple model diagrams together (see Sub-chapter 6.2.1 for an exploration of this idea). By showing multiple model diagrams on the same drawing, the interrelationships are made explicit versus requiring the engineer to mentally make the connections. When the engineer must mentally make the connections, a potential for errors arises. Some of the potential errors are forgetting a connection or relating the incorrect diagram. These errors can persist into the design and realization activities. An artifact that shows the explicit relationships between model diagrams can be reviewed, tested, and verified for correctness.

Extending the concept of the SOD, which is an assembly of related models, to the UML specification [OMG15b] has several benefits such as enabling new modeling and code generation possibilities (see Sub-chapter 6.2.2, an initial exploration of this idea). The extension idea consists of relating models to each other and to relating parts of models to each other. For example, a behavior specification such as a Sequence Model or Activity Model can be related to a method in a Class Model, thereby defining the internal logic or algorithm of the method. With the algorithm specified in a language neutral format, the models can be used to generate code in a variety of programming languages. Another example is relating a method to a state machine transition to indicate that the transition happens during the execution of the method. Such relationships can provide an additional level of verification of proper solution behavior through the models.

The two UML extensions ideas, diagram and models, are somewhat independent and

each conveys different information. Composing multiple diagrams into a SOD might only show that there is some relationship, such as they are all views of the design. The presentation is the common factor. Composing models together is a stronger relationship and one would expect the model compositions to be potentially visible in an associated diagram. Just because a feature is defined in a model does not mean that it is rendered in any diagram.

Full exploration of the UML extension ideas stimulated by the SOD requires research and analysis beyond the scope of this thesis and is a future research topic.

6.2.1 Composition Extensions for the UML Diagram Definition Specification

As described in Sub-chapters 3.5.1.1 and 6.1.2, the SOD shows the assembly of multiple visual representations of the requirements in order to improve the communication within the team. The SOD is a simple assembly of images intended to convey a larger picture and it is left to the viewer to mentally interpret them. The composition of multiple images into one is a simple but effective communication improvement. This sub-chapter explores changes to the UML Diagram Definition specification to extend the UML with this composition capability.

The Diagram Definition (DD) specification architecture, as applied to UML (Figure 6.3), has two main components, the Diagram Interchange (DI) and Diagram Graphics (DG). The DG models, figures, and shapes are the same in all tools and are not exchanged. The DI models specify that the user-defined data, such as position of nodes and line routing points, are captured for interchange between tools. There are also some common elements from a Diagram Common (DC) model. The SOD extensions will impact the DI and DG models. Figure 6.3 also shows an example of the mapping for a UML Use Case besides the DD architecture components.

The extensions needed to incorporate the SOD concepts will primarily impact the DI architecture component and to a lesser extent the DG architecture component. To see the exact impacts, more detailed DD models must be examined. A key goal of the DD specification is to reuse other specifications as much as possible rather than create new ones. This can be seen in the use of the MOF [OMG14b] specification. The SOD extensions will follow this practice. However, the language mapping specifications are designed to allow for some flexibility, such as in the specialization from the DI and the mappings to the DG.

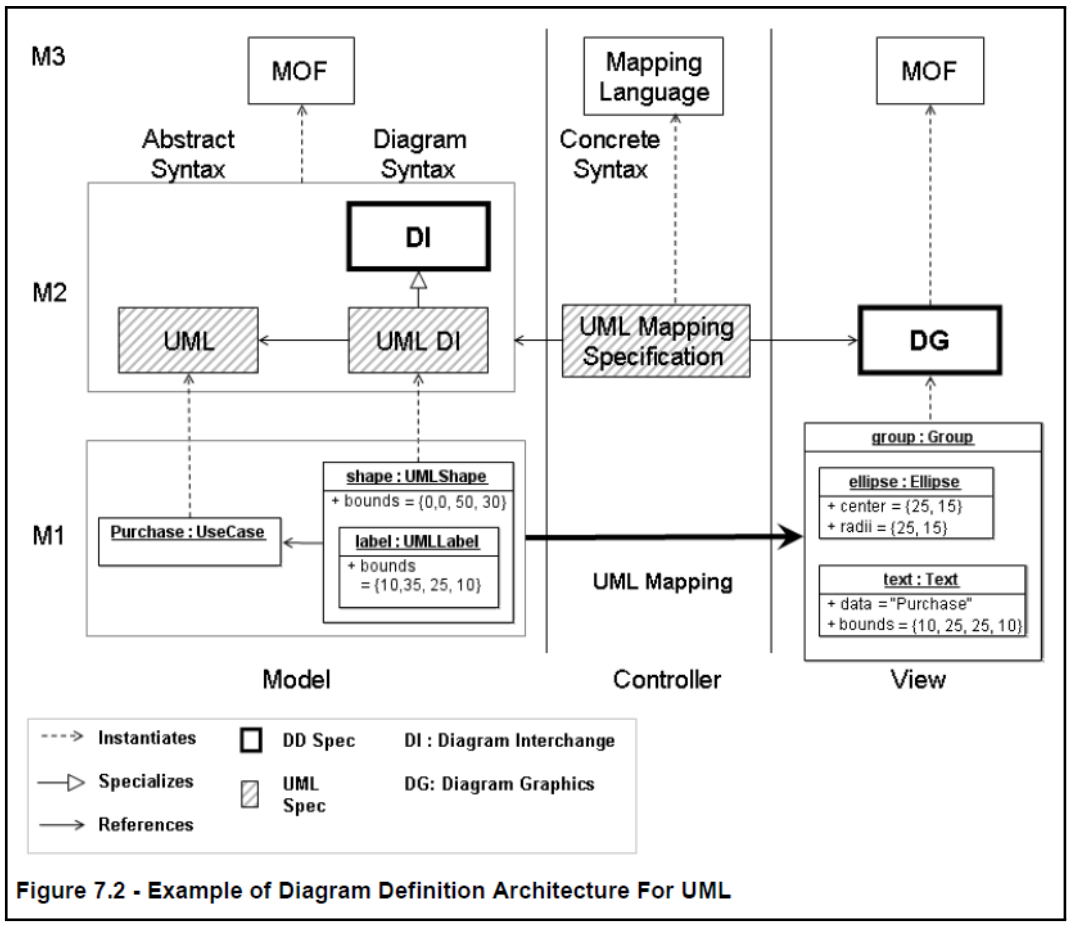


Figure 6.3: OMG Diagram Definition Architecture for UML.

The extensions for the SOD in the DI specification are based on the DI model. From the Diagram Definition specification, the DI model is shown in Figure 6.4. The *DiagramElement* abstract class has a containment relationship that allows a hierarchy of *DiagramElement* objects to be built. Through generalization relationships the capability to build a hierarchy of *Diagram* objects is realized.

In order to create a diagram that is the equivalent of the SOD, as expressed in Figure 6.1, the *Composition Attributes* relationship class needs to be added to the Diagram class of Figure 6.4. The resulting new model of the Diagram is the model in Figure 6.5, note that the focus is on the UML DD *Diagram* class as that is the only change. The UML DD *Diagram* class is extended via a generalization of the SAMEM *Drawing* class which brings the *Composition_Attributes* association class. The *Drawing* class has the minimum attribute of *Drawing_Size*, which will often be expressed as a common paper size, but that expression is a factor of the implementing tool. Each component of the drawing is another *Drawing* with the positioning attribute of *Drawing_Location* within the containing composition *Drawing*. The cardinalities on the *Composition_Attributes*

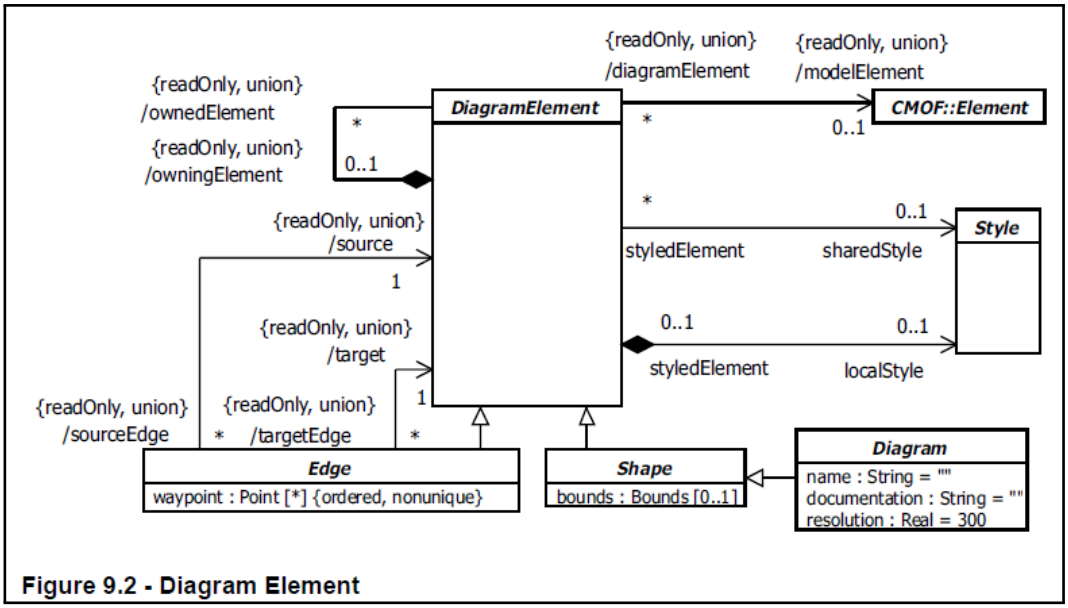


Figure 6.4: DI Model from the OMG DD Specification.

relationship ensure that the creation of a *Drawing* composition is optional.

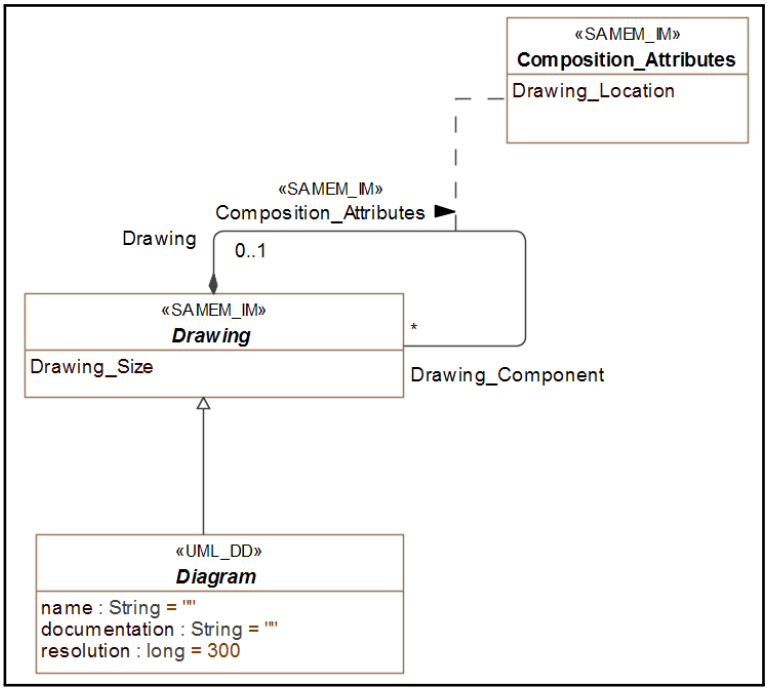


Figure 6.5: Merging of UML DD and SAMEM-IM Drawing Concepts.

The purpose of the *Drawing* object, as shown in Figure 6.1, is for communication, so that if the drawing is not needed, it does not have to be created. That purpose is transferred to Figure 6.5. The components used to create a drawing need not exclusively be UML diagrams but any visual specification that assists in communication. In practice, a specific UML tool implementation will probably add additional attributes to both the *Drawing* class and the *Composition_Attributes* class for value added extensions.

6.2.2 Composition Extensions for the UML Specification

The composition extension idea from the SAMEM, as embodied in the SOD, can be applied to the UML specification for benefits beyond the drawing for communication purposes. Applying the composition idea to the UML is different from the Diagram Definition. The composition does not result in a *Drawing* for communication purposes but rather results in relating models for extending the definition of behavior. When the composed models are rendered into diagrams and therefore drawings, as described in Sub-chapter 6.2.1, the *Diagram* or *Drawing* will have multiple models composed together.

An example will illustrate the model composition purpose and benefit. The classes in a UML Class Model have operations as part of the definition, however, algorithmic behavior of the operations are not defined in the Class Model. For many good reasons, as pointed out by Parnas [Par72], the implementation should be hidden. However, by linking a behavior specification to the operation at the right point in the design, the operation can be verified and the realization (code) can be generated. The behavior can be specified via an Activity Model, or a Sequence Model, or a combination of both.

In Figure 6.6, the high-level idea of a link between a UML behavior model and an operation in a Class Model is displayed as an illustration of the concept. The *UML Activity Model* is shown as being linked via an *Operation Behavior Definition Link* to an *Operation* in a UML Class. The *UML Activity Model* defines the behavior logic or algorithm of the operation execution. Figure 6.6 also illustrates the concept of defining the operational steps into blocks of pseudo code, which can be translated into a variety of code languages. For example, *Pseudo Code Group 1* could be an internal attribute or variable declarations and *Pseudo Code Group 2* could be initialization actions or input/read actions.

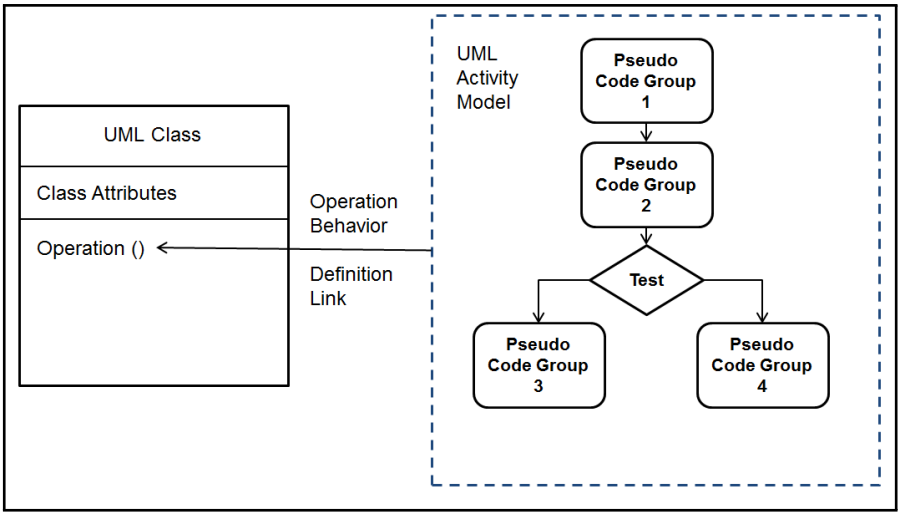


Figure 6.6: UML Class Operation to Behavior Link Concept.

There are three UML models that define behavior: Activity Models, Sequence Models, and State Machine Models. Operations in a UML Class Model that do simple calculations would probably be linked to an Activity Model for the calculation or algorithm behavior. Operations that provide execution control actions would likely be defined by a Sequence Model, which shows the order of other operation calls. When the purpose of an operation is to create or respond to an event, then a link to a State Machine Model might be appropriate.

The application of the SOD concept to UML models results in linking multiple models together. This would then require the display of the model diagrams together with the links displayed to indicate the relationships. This display requirement comes back to the extension of Diagram Definition Standard as described in Sub-chapter 6.2.1.

Specific enhancements to the UML metamodel, as described in [OMG15b] to accommodate the application of the SOD concept as shown in Figure 6.6, need to be gathered from the various chapters and diagrams in the specification. The starting point is Figure 9.1 from the UML specification which is reproduced in Figure 6.7 for reference.

The key part of the *Classifiers Abstract Syntax*, as shown in Figure 6.7, is the *Feature*. The Feature represents at a high level structural and behavioral characteristics of a *Classifier*. Refinement of the *Feature* leads to the *Operation* of a *Class*. The refinement of the *Feature* is shown in Figure 6.8 as a *BehavioralFeature*.

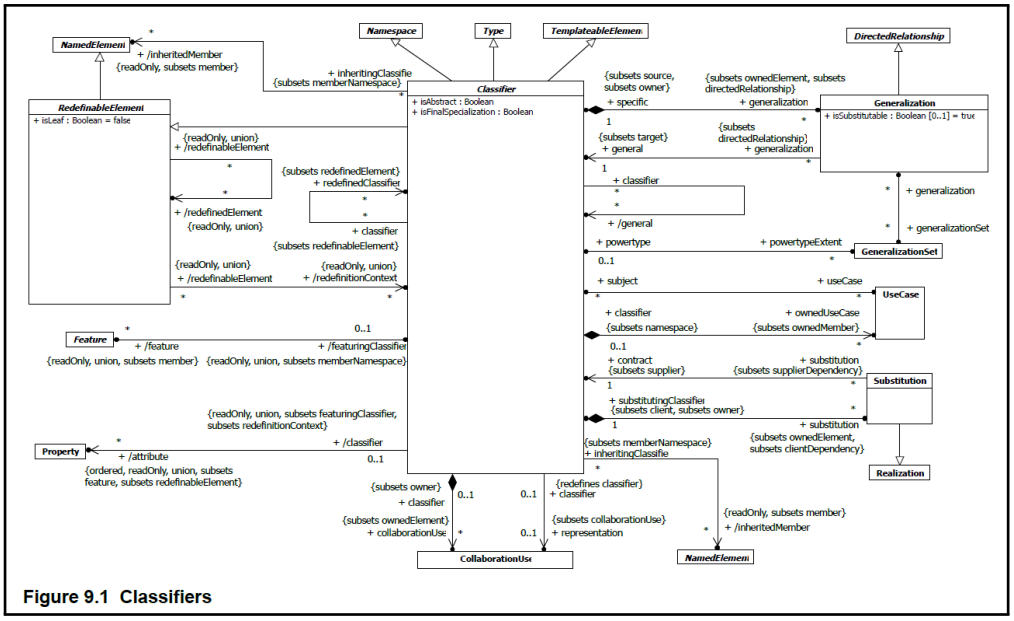


Figure 6.7: Reproduction of UML 2.5 Figure 9.1 for Reference.

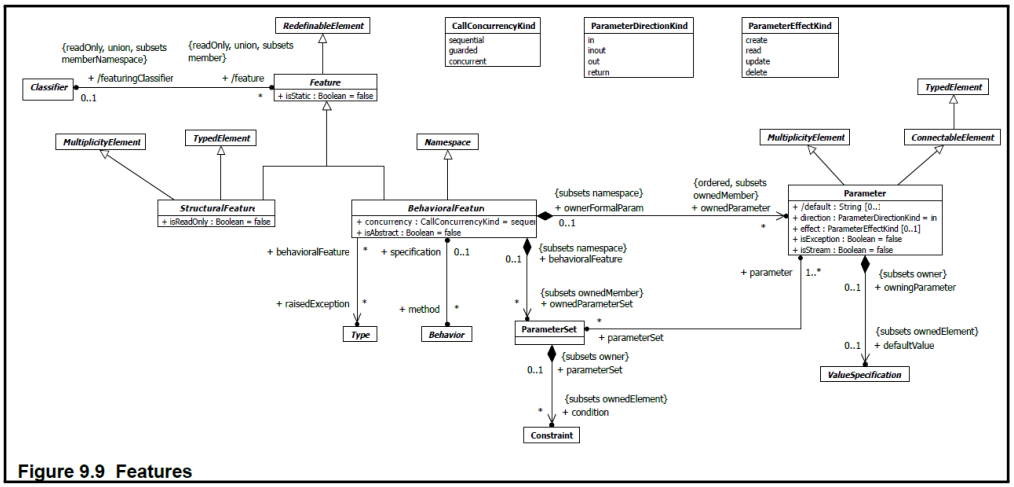


Figure 6.8: Reproduction of UML 2.5 Figure 9.9 for Reference.

An *Operation* is a *BehavioralFeature* that may be owned by an Interface, DataType, or Class. The refinement of *BehavioralFeature* to *Operation* is shown in Figure 6.9. The *Operation*, as defined in Figure 6.9, is one endpoint of the link needed to fulfill the design as expressed in Figure 6.6. The other link end is a *Behavior*.

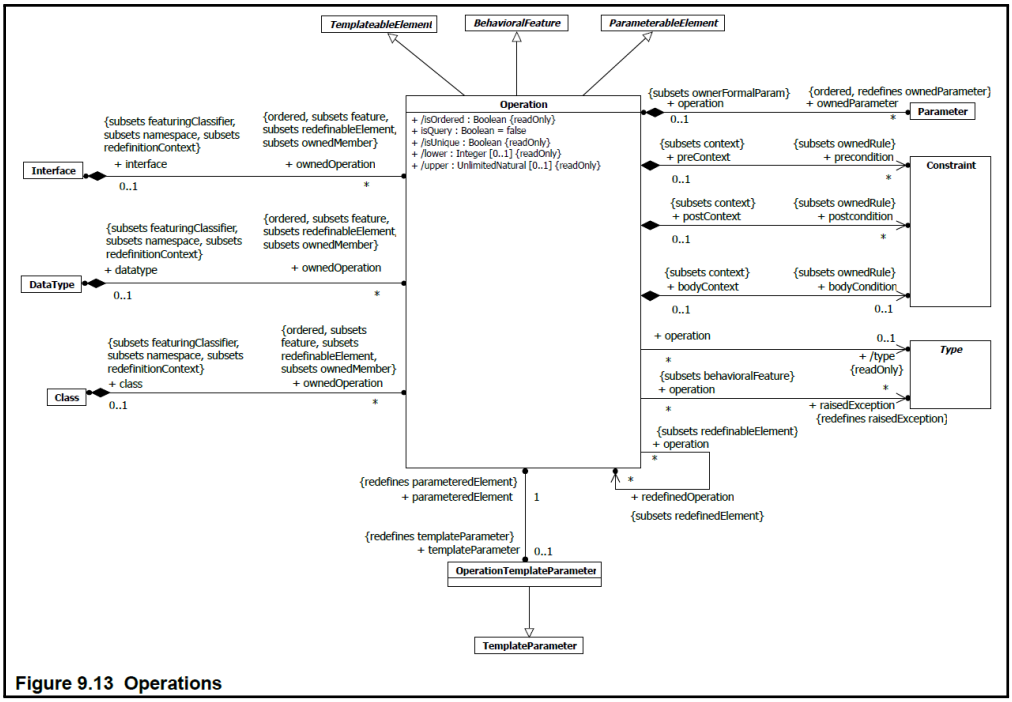


Figure 6.9: Reproduction of UML 2.5 Figure 9.13 for Reference.

There are three types of *Behavior Classifiers* defined in UML 2.5 [OMG15b]: *StateMachines* are defined in Clause 14, *Activities* are defined in Clause 15, and partially-ordered *Sequences* are defined in Clause 17. A subset of the abstract syntax definition for the Behavior is shown in Figure 6.10.

Although the generalization hierarchy has not been completely shown in Figure 6.9, the *Operation* can be traced to the *Element* concept which is allowed to have multiple *DirectedRelationship* connections. The *Behavior* element in Figure 6.6 also traces back through generalizations to the *Element* concept. This means that a new *DirectedRelationship* can be connected between an *Operation* and a *Behavior*. The UML contains a definition of a *Dependency* relationship which is a refinement of the *DirectedRelationship*. The *Dependency* relationship contains further refinements, one of which is *Realization*. The UML abstract syntax for *Dependency* is shown in Figure 6.11.

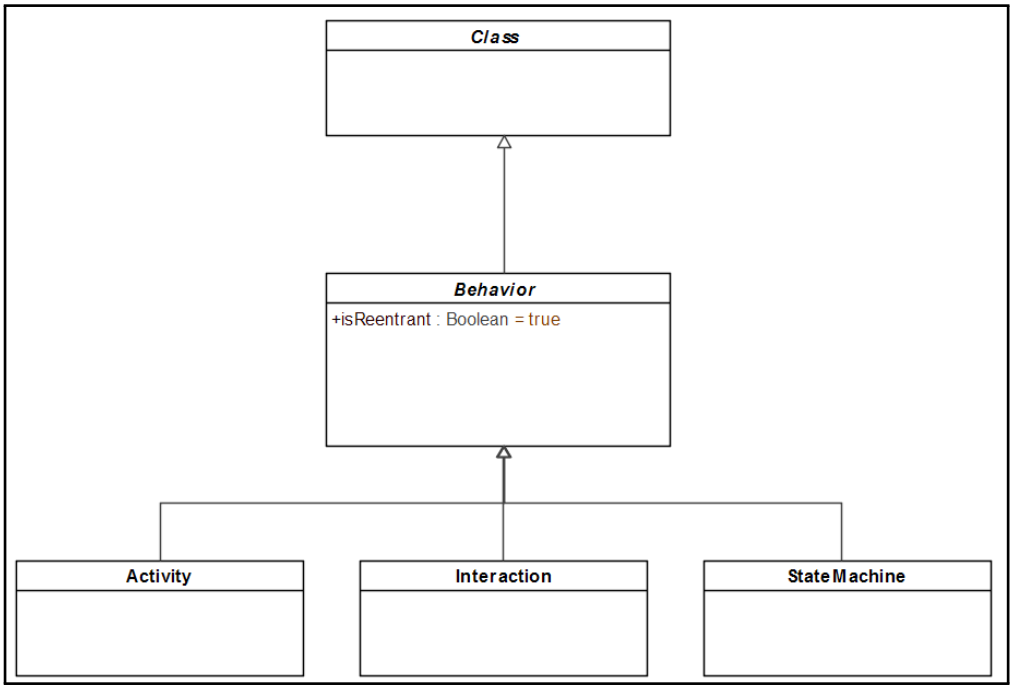


Figure 6.10: Composite of UML Behavior Subclasses.

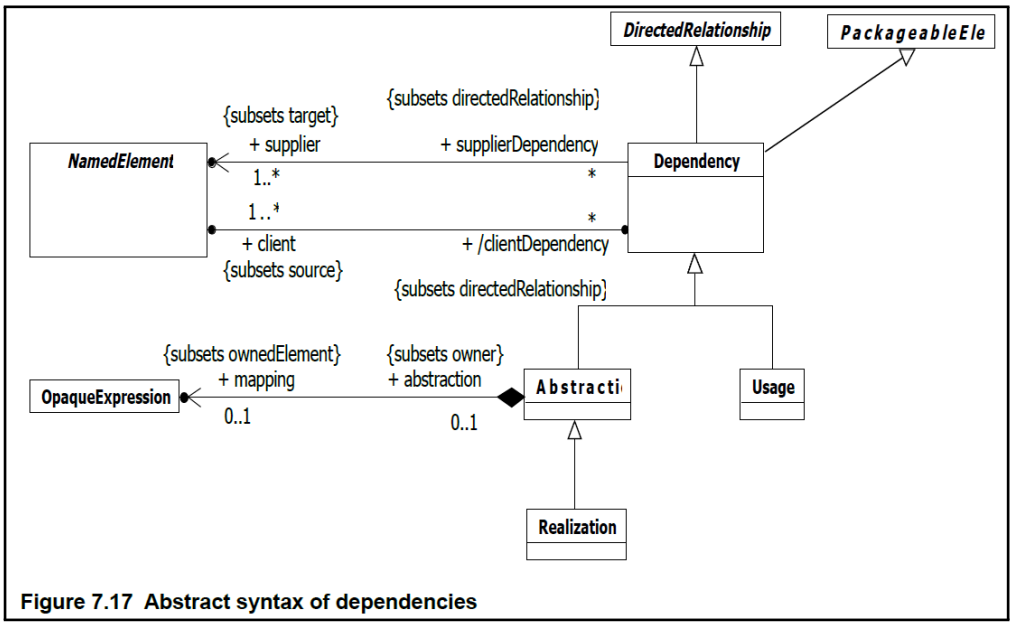


Figure 6.11: Reproduction of UML 2.5 Figure 7.17 for Reference.

The *Realization* relationship appears to fulfill the needs of the concept expressed in Figure 6.6. However, the definition of the *Realization* relationship works with sets of *NamedElements*, which is more than is needed. Rather than overloading the existing *Dependency* object, a new specialization, *OperationBehavior*, on the same level as *Dependency* is proposed and is shown in Figure 6.12. The *OperationBehavior* relationship links an *Operation* to a single *Behavior*. A *Behavior* model can provide the behavior for multiple *Operations*, for example searching behavior. Another benefit of having a specialized relationship is the ability to clearly assign appropriate code generation logic.

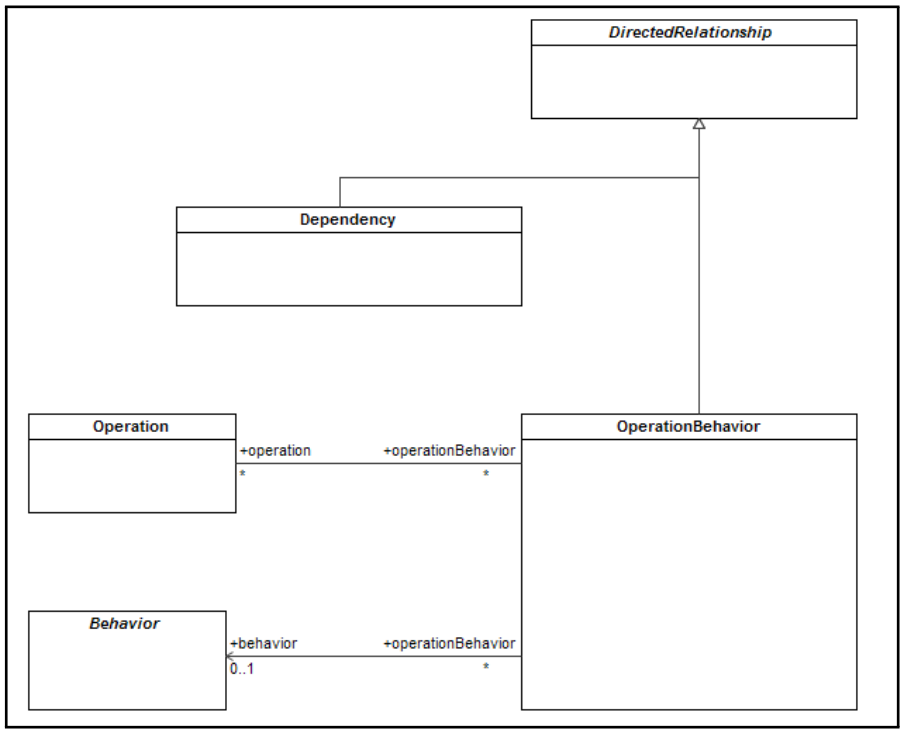


Figure 6.12: Operation Behavior Extension to UML.

There are additional improvements to the UML needed to fulfill the linking concept shown in Figure 6.6 which are extensions to the Notation clauses in the UML specification [OMG15b]. The extensions affect the specifications for *Activity*, *Interaction*, and *StateMachine*. Since *Behavior* is a subtype of *Class* and *Class* is a subtype of *Type* which is a subtype of *PackageableElement*, it is possible to define additional notations which encapsulate the behaviors and produce a Classifier that can participate in diagrams. The extensions are shown in Figure 6.13, which is a modification of Figure 6.10.

The UML specification Clause 15.2.4 [OMG15b] defines the notation for *Activities*. An explicit notation that encapsulates an *Activity* model does not exist in the current specification. The proposal is for a Class-like notation with the *ActivityModel* name that can participate in the *OperationBehavior* relationship. Associated with the *Activity Model* is a diagram, consistent with current usage, which has the *ActivityModel* detailed

definition.

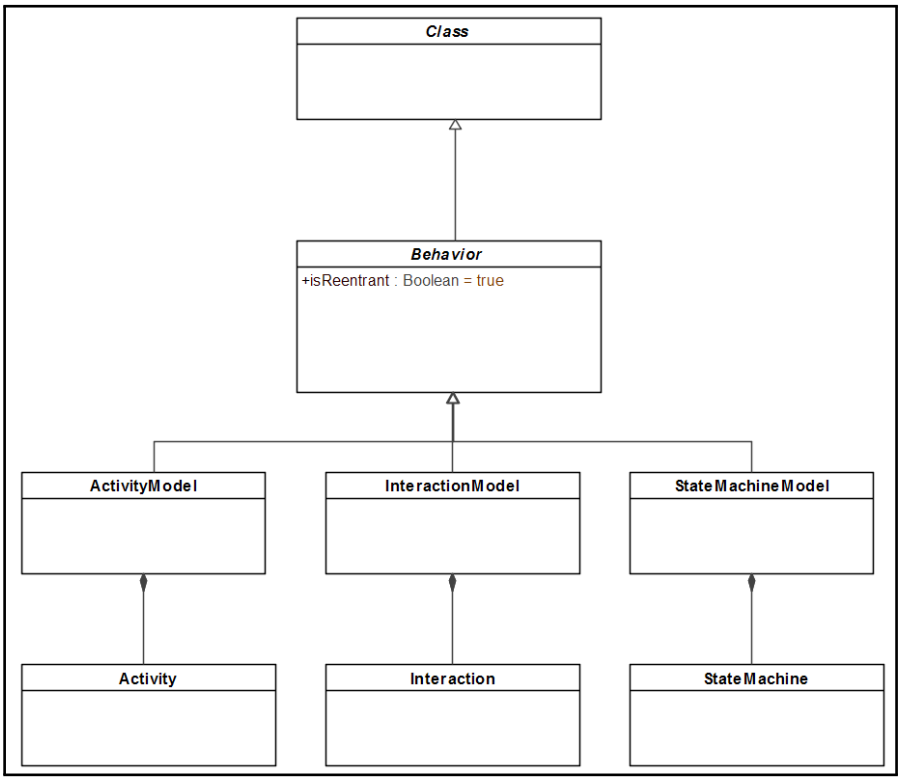


Figure 6.13: Behavior Models Extension.

The InteractionModel Classifier element in Figure 6.13 provides an explicit container for the *Sequence* diagram mode, according to the UML Clause 17.2.4, and an element with a notation that can be placed in diagrams. The notation for the *InteractionModel* is a rectangle with the name of the *Interaction*.

Clause 14.2.4 in the UML specification [OMG15b] defines the notation for *StateMachine* behavior. The notation definition is that "StateMachine diagrams specify StateMachines." The current specification does support a notation for the *StateMachine* Classifier, as shown in Figure 6.10.

An explicit notation for the *StateMachine* Classifier is proposed that is a rectangle with the *StateMachineModel* name and is a container for the *StateMachine* which holds all Regions, States, and other elements that appear in a *StateMachine* diagram.

6.3 Information Modeling Summary

A metamodel for the SAMEM Information Model concepts is proposed that allows for implementation in a variety of software tools, such as construction or analysis tools. The metamodel provides a basis to checking and extending as new concepts are needed.

The SOD concept from the SAMEM metamodel has been mapped to extensions for the UML specification. The SOD concept in its simplest form supports the creation of drawings that contain multiple viewpoints for improved communication. A more complex application of the SOD concept is to extend the UML to allow for behavior, an *Operation*, to be connected to behavior, UML Behavior Classifier, for an explicit operational behavior definition. The connection can assist in communicating design intent in a manner similar to pseudo code and in improved code generation.

Chapter 7

The SAMEM Evaluation

The SAMEM will be evaluated along 5 dimensions:

1 empirical surveys of the case study users in multiple real-world projects and their evaluations of the fulfillment of the SAMEM goals,

2 a discussion of the use of the SAMEM ideas on other projects with some initial indicators of applicability,

3 the cognitive effectiveness of the modeling ideas in communicating requirements and the unique SAMEM SOD visual artifact,

4 an analytical comparison with the Software Engineering Method and Theory (SEMAT) [SEMAT] standard as to project scope, differences, and similarities,

5 an analytical evaluation against the consulting business success measures of smaller team, faster project, and improved customer relationship.

There are multiple goals (see Sub-chapter 2.4.1) that the SAMEM is measured against for success. The primary mechanism for determining whether the SAMEM meets the goals is through a survey of people that used the SAMEM or an early version of the SAMEM in actual industry projects. The survey is an empirical evaluation primarily from the case studies described in Sub-chapter 4.1. The early version of the SAMEM was called the "Paper Prototype Process." The term SAMEM shall be used for both going forward. The Customer's Survey is targeted to the project stakeholders and their opinions of the early version of the SAMEM. The Developer's Survey is targeted towards the members of the development team and their opinions of how the SAMEM impacted the development tasks. There are common questions on both surveys with some different questions respecting the differences in the application of the SAMEM.

In addition to the two customer companies and 18 projects that are evaluated by the surveys, the SAMEM or an earlier version (Paper Prototype Process) is being applied in three other companies. At the time of this thesis, the projects are in the beginning stages and only preliminary results are available.

Since the SAMEM relies heavily on graphical or visual communication mechanisms, they will be evaluated through cognitive effectiveness measures. Cognitive effectiveness is a measurement of how easy it is to understand the graphical representations [LS87], [Moo09]. The use of techniques that help in distinguishing the symbols and conveying

their meanings will be the primary measure. It is expected that improvements will arise from the cognitive effectiveness evaluations. Cognitive effectiveness was introduced in Sub-chapter 2.3.2.

The SAMEM and the SEMAT standard will be compared. A goal of the SAMEM is the ability to adapt to new software engineering ideas as stated in ***GOAL-11***. The comparison between the SAMEM and SEMAT will give an indication of the success in meeting the adaption goal.

HL-GOAL-1, ***HL-GOAL-2***, ***HL-GOAL-4***, and ***HL-GOAL-5*** are primarily business goals of the SAMEM as opposed to technology goals. The ability of the SAMEM to be successful in a competitive business environment will be evaluated.

Weaknesses to the evaluation methods and possible threats to the validity of the conclusions will conclude this chapter. Specific threats to the construct validity, the internal validity, the external validity, and the reliability will be discussed.

7.1 Empirical Evaluation of Case Study Companies

A survey questionnaire for the Customers (CQ) (see Appendix A) and a survey questionnaire for the Developers (DQ) (see Appendix B) were created and sent to participants of the case studies and to the SEC company, see Sub-chapter 7.2.3. While there are common questions between the two surveys, there are differences based on the project tasks for the two different groups. The commonalities and differences will be discussed in detail in the following chapters. An early version of the SAMEM was called the *Paper Prototype Process* which is the term used in the questionnaires as that is the language the participants are familiar with. The mechanics of conducting the survey and gathering the results is handled through the commercial application *SurveyMonkey* (https://www.surveymonkey.com/). The survey detailed responses are listed in Appendix C through Appendix J.

The surveys were sent to thirteen people that used the early version of the SAMEM methodology (pre-SAMEM) in industrial projects that delivered real solutions. The Customer Survey was sent to a total of six people on the customer or stakeholder side of the business. The Developer Survey was sent to seven people that worked in a developer role of some sort on real-world industrial projects. Many of the developers and customers worked on the same projects. The customer response rate is 4 of 6 or 66%. The developer response rate is 4 of 7 or 57%. One Developer Survey was completed by the company described in Sub-chapter 7.2.3, which means that the response rate from the case study companies is 3 of 7 or 43%. With such small numbers, the validity of any statistical calculations is questionable, so simple counts and ratios will be used. The difficulties of performing empirical evaluations in industry projects are discussed in [WRH+00] and [SSS08]; however, despite the often lower numbers involved in the surveys or experiments, there are distinct advantages to having real-world results. The primary benefit of a real-world experiment is the confidence gained from knowing the idea works in practice, not just theory. The evaluation of the industry results must be carefully done keeping in mind possible contamination by unconsidered aspects of the environment.

7.1.1 Survey Definition Details

The survey questions are constructed to address the goals and requirements of the thesis as stated in Sub-chapter 2.4.1. The survey is designed to evaluate the use of an early version of the SAMEM during real-world use in multiple case studies [WRH+00], [SSS08], [OP97]. The viewpoint taken by the author is a combination of Critical Theory and Pragmatism (see Chapter 11 in [SSS08]). The SAMEM seeks to improve aspects of requirements work and the transition into design work for medium to large projects and as such new alternatives to the tools, processes, and artifacts are expected. The surveys are designed to evaluate the SAMEM requirements as put into practice in the case studies described in Sub-chapter 4.1. The Customer Survey has 29 questions and the Developer Survey has 31 questions.

The survey questions are organized into several sections:

- **Survey Purpose and Overview** – an introduction to the survey and its purpose.
- **Overall Paper Prototype Evaluation** – contains questions about the overall evaluation of the Paper Prototype process in relation to the progress and management of the project.
- **Communication Evaluation** – a set of questions focusing on the use of the graphical representations and models for the communication amongst the project members. A key aspect of the Paper Prototype Process is the increased use of visual or graphical representations of the requirements over the more traditional text-based mechanisms.
- **Requirements Quality** – contains questions about the hypothesis behind the Paper Prototype process of using graphical specification techniques to produce a more compact and higher quality specification.
- **Development and Testing Evaluation** (only in the Developer survey) – questions about easing the requirements gathering and specification transition to the development activities.
- **General Paper Prototype Experience** – questions about general experience, other project experiences, and role with the project.

For some questions and for each section, there is the possibility for the respondent to enter additional explanations or comments. As part of the evaluation, the comments will be used for evidence. When a comment is used to substantiate a claim, the literal text unchanged from the respondent is quoted.

There are parts of the SAMEM that affect the work of the customers and the developers equally. Some questions aim at evaluating the effectiveness of the SAMEM artifacts, processes, and tools for both groups. The questions collect qualitative data about the effectiveness (see Chapters 3 and 11 in [SSS08]). The answers reflect the personal experience of the customers and developers in using the early version of the SAMEM on real-life projects.

7.1.1.1 Common Questions of the Surveys

The questions that are the same in both surveys are listed in Table 7.1. In some cases the question number is different because the other survey has additional questions or the order of questions is changed. It is intentional that the question is formulated the same for both surveys, the possible answers are the same, and in the same order. There are 25 questions that are the same on both the surveys.

Table 7.1: Common Survey Questions.

Question Text	**Customer Question Number**	**Developer Question Number**
Overall Paper Prototype Evaluation Section		
What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.	1	1
How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?	2	2
How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?	5	3
How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?	6	4
Communication Evaluation Section		
What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?	7	5
What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?	8	6
What effect did the "1-pager" have on project iteration planning?	9	7
What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?	10	8

Question Text	**Customer Question Number**	**Developer Question Number**
Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?	11	9
Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?	12	10
Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?	13	11
Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?	15	12
Requirements Quality Section		
Do you feel that the visual models of the business information and behavior requirements affected the **speed** of the review task for correctness and completeness?	16	13
How do you feel that the use of visual models for the requirements affected the **clarity** of the requirements specification?	18	14
How do you feel that the use of visual models for the requirements affected the **size** of the requirements specification?	19	15
Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?	20	18
If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?	21	22

Question Text	Customer Question Number	Developer Question Number
Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?	22	23
What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?	23	24
General Paper Prototype Experience Section		
Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?	23	26
Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?	25	27
Experience in using the Paper Prototype process (time in months)?	26	28
Previous experience with software-based solution development projects?	27	29
What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.	28	30
What is your evaluation of the learning effort needed to participate in the Paper Prototype process?	29	31

The bulk of the questions are the same for both customers and developers. As will be seen in the analysis chapter, the differences in answers are related to the differences in tasks and perspective.

7.1.1.2 Unique Survey Questions

There are some different questions between the surveys, as the customer group and developer group have different jobs and tasks. The SAMEM must provide support for the specialized activities of each group. A dash character, "—", in Table 7.2 indicates that the question does not appear in the other survey. There are 10 unique questions between the two surveys.

Table 7.2: Unique Survey Questions.

Question Text	**Customer Question Number**	**Developer Question Number**
Overall Paper Prototype Evaluation Section		
Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?	3	—
In your experience, did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?	4	—
Communication Evaluation Section		
For a new solution, did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?	14	—
Requirements Quality Section		
Do you feel that the visual models of the business information and behavior requirements affected the **accuracy** of the review task for correctness and completeness?	17	—
Development and Testing Evaluation Section		
Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?	—	16
Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?	—	17
Do you feel that the visual components of the requirements specification had effects on developing tests?	—	19
Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?	—	20
Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?	—	21
Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?	—	25

The differences in the sections of *Overall Paper Prototype Evaluation*, *Communication Evaluation*, and *Requirements Quality* are questions that are more of a business nature than of a development nature. In the *Development and Testing Evaluation* section, the different questions are related to the transition from requirements specification understanding to design, development, and testing activities.

7.1.2 Answers to the Common Questions Merged

The 25 common questions results can be merged into a single evaluation. In the first level of results summarization, comments will be left out. As the survey results are used to demonstrate achieving the SAMEM requirements in Sub-chapter 7.1.4, both positive and negative comments will be considered for additional insight.

Most of the questions have an ordinal answer spectrum from positive to negative with a neutral middle value, for example: Yes, Better; Mostly Better, but with some limitations; Occasionally Better; Seldom Better; Not Better. The same ordinal scale was used as often as made sense. In Table 7.3, only the answers selected by any respondent are shown for that question, even though the ordinal scale is larger. An answer cell entry in the Answers column will show the answer ordinal value and a fraction of those answers over all respondents, for example an entry of 5/8 indicates that 5 respondents of the total of 8 picked that answer value. There are instances of respondents not answering a question, so the total answers can be less than the total number of respondents.

Table 7.3: Common Questions Answer Merge .

Question Text	**Cust. No.**	**Dev. No.**	**Ans 1**	**Ans 2**
Overall Paper Prototype Evaluation Section				
What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.	1	1	Very Effective 7/8	Moderately Effective 1/8
How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?	2	2	Very Positive 7/8	Moderately Positive 1/8

Question Text	**Cust. No.**	**Dev. No.**	**Ans 1**	**Ans 2**
How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?	5	3	Yes Better 7/8	Mostly Better 1/8
How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?	6	4	Very Helpful 5/8	Helpful Limitations 2/8

Question Text	Cust. No.	Dev. No.	Ans 1	Ans 2
Communication Evaluation Section				
What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?	7	5	Very Effective 8/8	Somewhat Effective
What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?	8	6	Very Effective 6/8	Somewhat Effective 2/8
What effect did the "1-pager" have on project iteration planning?	9	7	Very Effective 4/8	Somewhat Effective 4/8
What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?	10	8	Yes Better 7/8	Mostly Better 1/8
Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?	11	9	Yes Better 7/8	Mostly Better 1/8
Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?	12	10	Yes Better 5/8	Mostly Better 1/8

Question Text	**Cust. No.**	**Dev. No.**	**Ans 1**	**Ans 2**
Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?	13	11	Very Helpful 4/8	Helpful Limitations 3/8
Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?	15	12	Very Useful 5/8	Useful Limitations 3/8
Requirements Quality Section				
Do you feel that the visual models of the business information and behavior requirements affected the **speed** of the review task for correctness and completeness?	16	13	Yes Better 5/8	Mostly Better 3/8
How do you feel that the use of visual models for the requirements affected the **clarity** of the requirements specification?	18	14	Yes Better 8/8	Mostly Better
How do you feel that the use of visual models for the requirements affected the **size** of the requirements specification?	19	15	Yes Better 7/8	Mostly Better 1/8
Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?	20	18	Yes Better 6/8	Mostly Better 2/8
If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?	21	22	Yes Better 7/8	Mostly Better 1/8

Question Text	Cust. No.	Dev. No.	Ans 1	Ans 2
Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?	22	23	Yes Better 8/8	Mostly Better
What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?	23	24	Yes Better 4/8	Mostly Better 4/8
General Paper Prototype Experience Section				
Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?	24	26	Yes 3/8	No 5/8
Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?	25	27	Yes 7/8	No 1/8
Experience in using the Paper Prototype process (time in months)?	26	28	> 24 months 6/8	12 – 24 months 1/8
Previous experience with software-based solution development projects?	27	29	0 1/8	2 – 4 1/8
What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.	28	30	The respondent spectrum will be analyzed in Table 7.5	
What is your evaluation of the learning effort needed to participate in the Paper Prototype process?	29	31	Easy 4/8	Medium 4/8

In summary, the answers for the common questions were all on the positive side of the ordinal scale, although there are comments with suggestions for improvement. The comments will be addressed in Sub-chapter 7.1.4 in the context of the SAMEM goals evaluation.

7.1.3 Unique Survey Questions Response Values

In Table 7.4, a summary of the answers to the unique survey questions is presented. Only the answers selected by any respondent are shown for that question. The answer cell entry will show the answer and a fraction of those answers over all respondents, for example an entry of 3/4 indicates that 3 respondents of the total of 4 picked that question answer. There are instances of respondents not answering a question, so the total answers can be less than the total number of respondents.

Table 7.4: Unique Survey Questions Response Values.

Question Text	**Cust. No.**	**Dev. No.**	**Ans 1**	**Ans 2**
Overall Paper Prototype Evaluation Section				
Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?	3	—	Yes 4/4	No
In your experience, did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?	4	—	Yes 4/4	No
Communication Evaluation Section				
For a new solution, did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?	14	—	Very Helpful 4/4	Helpful Limitations
Requirements Quality Section				
Do you feel that the visual models of the business information and behavior requirements affected the **accuracy** of the review task for correctness and completeness?	17	—	Yes Better 4/4	Mostly Better Limitations

Question Text	Cust. No.	Dev. No.	Ans 1	Ans 2
Development and Testing Evaluation Section				
Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?	—	16	Yes Better 4/4	Mostly Better Limitations
Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?	—	17	Yes Better 3/4	Mostly Better Limitations 1/4
Do you feel that the visual components of the requirements specification had effects on developing tests?	—	19	Yes Better 4/4	Mostly Better Limitations
Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?	—	20	Yes Better 2/4	Mostly Better Limitations 2/4
Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?	—	21	Yes Better 3/4	Mostly Better Limitations 1/4
Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?	—	25	Yes Better 4/4	Mostly Better Limitations

All answers are on the positive side of the ordinal range. The questions with answers that ex-pressed some limitation will be discussed in detail in Sub-chapter 7.1.4 in the context of the SAMEM goals and requirements as outlined in Sub-chapter 2.4.1.

7.1.4 Survey Results and SAMEM Goals Evaluation

The high-level goals are restated below for convenience. Survey responses are used as evidence that the SAMEM design has fulfilled the high-level goals of the methodology. There are multiple survey questions that provide the evidence.

When a quote from a respondent comment adds clarification, improvement, or insight, it will be placed in the corresponding goal rationale. The respondent quotes are used exactly as entered in the survey response.

The following reference nomenclature is used for the related survey questions:

- To indicate the Customer Survey *CS* is used.
- To indicate the Developer Survey *DS* is used.
- The related question is a Q with a number, as in *Q1*.
- For the common questions, both the Customer Survey question reference and the Developer Survey question reference will be listed.
- Some survey questions will provide evidence for multiple SAMEM goals.
- For example, the reference *CS-Q15* indicates Customer Survey Question 15 as providing the evidence.

The following nomenclature is used for indicating quote sources:

- To indicate the Customer Response *CR* is used.
- To indicate the Developer Response *DR* is used.
- The related response as in 1. The response number reflects the chronological order of the response, i.e. *1* indicates the first response.
- For example, the reference *CR-1* indicates Customer Survey Response 1 as providing the evidence.

The last four questions of each survey are the same and ask for some respondent background information. While the surveys are anonymous, the background information can put the answers and comments into the context of the respondent's experience. In Table 7.5, the background data of each respondent is listed.

Table 7.5: Respondent Experience.

Respondent	**CS-Q26 Paper Prototype Experience**	**CS-Q27 Software Project Experience**	**CS-Q28 Primary Business Role**	**CS-Q29 Paper Prototype Learning Effort**
CR-1	> 24 months	> 4 projects	Business/ customer stakeholder	Easy: understood the concepts immediately
CR-2	> 24 months	> 4 projects	Business/ customer stakeholder	Medium: some concepts were quick to learn, while others took longer State diagrams and new acronyms and words are a bit of a learning curve but should be easy to grasp by the average corporate technology or business worker.
CR-3	> 24 months	2 —4 projects	Business/ customer stakeholder	Easy: understood the concepts immediately
CR-4	> 24 months	none	Business/ customer stakeholder	Medium: some concepts were quick to learn, while others took longer

Respondent	**CS-Q26 Paper Prototype Experience**	**CS-Q27 Software Project Experience**	**CS-Q28 Primary Business Role**	**CS-Q29 Paper Prototype Learning Effort**
DR-1	12 —24 months	> 4 projects	Technical member of solution development team, Solution technology technical expert, Other (please specify) Contributing team member in the thinking of the "team"	Easy: understood the concepts immediately Excepting the basis in RM-ODP. Still don't get that, but maybe someone will take me aside and explain.
DR-2	> 24 months	> 4 projects	Project manager	Medium: some concepts were quick to learn, while others took longer
DR-3	6 —12 months	> 4 projects	Solution technology technical expert, Project manager	Medium: some concepts were quick to learn, while others took longer Having experience with RM-ODP is essential to be effective in using the paper prototype process.

Respondent	CS-Q26 Paper Prototype Experience	CS-Q27 Software Project Experience	CS-Q28 Primary Business Role	CS-Q29 Paper Prototype Learning Effort
DR-4	12 —24 months	> 4 projects	Non-technical member of solution development team, Requirements Analyst, developed test cases, Technical Writer	Easy: understood the concepts immediately

In summary, every respondent had significant experience in using the early version of the SAMEM. All but one customer had significant experience in real-world software development projects. The learning effort was split between Easy and Medium with the comments indicating that new concepts need additional or improved explanation.

7.1.4.1 Supporting Evidence and Evaluation for High-level Goals

Each SAMEM high-level goal is evaluated against the evidence gathered through the surveys. For each high-level goal, a mechanism or set of mechanisms for achieving the goal is defined. The mechanisms are used to select relevant questions from the surveys. The evidence for each high-level goal is organized into the areas of Result, Questions, Ordinal Values Summary, Quotes, Improvements, and Related Detailed Requirements. A single question can provide evidence for multiple goals.

HL-GOAL-1: Better stakeholder communication with respect to the requirements specification and project progress.

This high-level goal is primarily focused on the customers group of the stakeholders in the solution, however, there is a developer aspect in the communication of requirements and design back to the stakeholders for confirmation. The survey questions relative to indicating that this goal is achieved are common questions in both surveys. Although the goal is stated as communication to the stakeholders, the reality is that a conversation takes place between the stakeholders and developers in achieving a common understanding on both sides.

The major mechanisms for achieving this high-level goal are the use of graphical representations of the solution goals, requirements, and designs, and the use of the RM-

ODP abstraction framework to appropriately direct the team focus during the project progression.

Result: goal achieved

Questions: CS-Q2/DS-Q2, CS-Q5/DS-Q3, CS-Q7/DS-Q5, CS-Q8/DS-Q6, CS-Q10/DS-Q8, CS-Q11/DS-Q9, CS-Q12/DS-Q10, CS-Q13/DS-Q11, CS-Q15/DS-Q12, CS-Q18/DS-Q14, CS-Q24/DS-Q26, and CS-Q25/DS-Q27.

Ordinal Values Summary: Highest: **74/96**, Second Highest: **19/96** (relative values since ranges differ, denominator is the number of questions times the respondents, or 12 x 8, sometimes a question was not answered, so the sum of the numerators will not equal the denominator).

Quotes: In addition to the ordinal values picked by the respondents, supporting comments to indicate achieving the goal are listed in Table 7.6.

Table 7.6: ***HL-GOAL-1*** Supporting Survey Quotes.

Question	Response	Comment
CS-Q13 DS-Q11	CR-2	Additional comments on business information unit modelling: Once the team resources understood the state machine model, it was very helpful in the project definition process.
	CR-3	Additional comments on business information unit modelling: State machine model was not always clear to some end users.
	DR-1	Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
	DR-3	Additional comments on business information unit modelling: We didn't successfully use the state machine models.

Question	Response	Comment
CS-Q15 DS-Q12	CR-2	Additional comments on the business flow model: Focus is on the primary flow as it should be. The 1-pager does help in consolidating solutions as well.
	CR-4	Additional comments on the business flow model: This is an effective tool to explain to upper management stake-holders.
	DR-1	Additional comments on the business flow model: If you can get the buy-in from the business to model their flow.
	DR-3	Additional comments on the business flow model: The usefulness depends a bit on who one is communicating with. There are people that just like things to be the traditional way.
CS-Q24 DS-Q26	CR-1	Gathering requirements is simply a matter of getting to the truth about what has to be delivered and in gaining consensus on the solution. Any means will do that if the participants are knowledgeable and engaged. With visual graphical approach, it makes the participants understand what is happening quickly and facilitate learning as well.
	CR-2	I have always had some tables, flows and pictures in specification so for me it fit with and extended the model I thought worked well for spec development.
	DR-2	The Paper Prototype process is very useful in nailing down many of the requirements, there is still the need for expressing some of the requirements in a traditional fashion (e.g., nonfunctional requirements).
	DR-3	Modelling takes time and addressing the necessary RM-ODP views requires experience in modelling to be successful within the time available. Therefore, it might be useful to have a paper prototype tool to support the process.

Question	Response	Comment
CS-Q25 DS-Q27	CR-2	Easier to digest, manage and handoff to other teams.
	CR-3	One major advantage was mapping processes, especially since most parts of the organization did not have those and did not have clarity on the extent of interplay and interactions with other parts of the organization. Also enabled communication to management, to ensure all parties were on the same page before committing resources.
	DR-1	A definition to defend.
	DR-2	Any tool that can help the user "see" and provide feedback on the proposed solution with a minimal time/investment is helpful. The Paper Prototype process is such a tool that can be effective with certain types of development efforts.
	DR-3	The paper prototype process forces you to think about the solution in a very concrete manner and help you visualize what you are building so that you can easier align between the team members to ensure they all agree on what it is that we are building.

Improvements or future work: The comments indicate improvements are needed in the communication of the RM-ODP standard and its application to controlling the abstraction level of project work. The role of the state machine requires better communication. A glossary explaining the acronyms and vocabulary for the start of the project would also be helpful.

Related Detailed Requirements: The related detailed goals are shown in Figure 2.8, which is in Sub-chapter 2.4.1. The survey question responses provide evidence of meeting the detailed goals. For convenience Figure 2.8 is reproduced below as Figure 7.1.

HL-GOAL-2: Better development team communication.

This high-level goal is analogous to ***HL-GOAL-1*** but with focus on communication within the development team as opposed to communication with the customer stakeholders. The developer to developer communication covers different subject areas from the developer to customer communication, such as design and technology alternatives. The relevant questions are in the Developer Survey with some common questions.

The same mechanisms used for stakeholder communication are applied to the improvement in developer team communication. The primary mechanisms are graphical models and RM-ODP for abstraction management. In addition, architecture and design patterns supplemented with design facts in text are important.

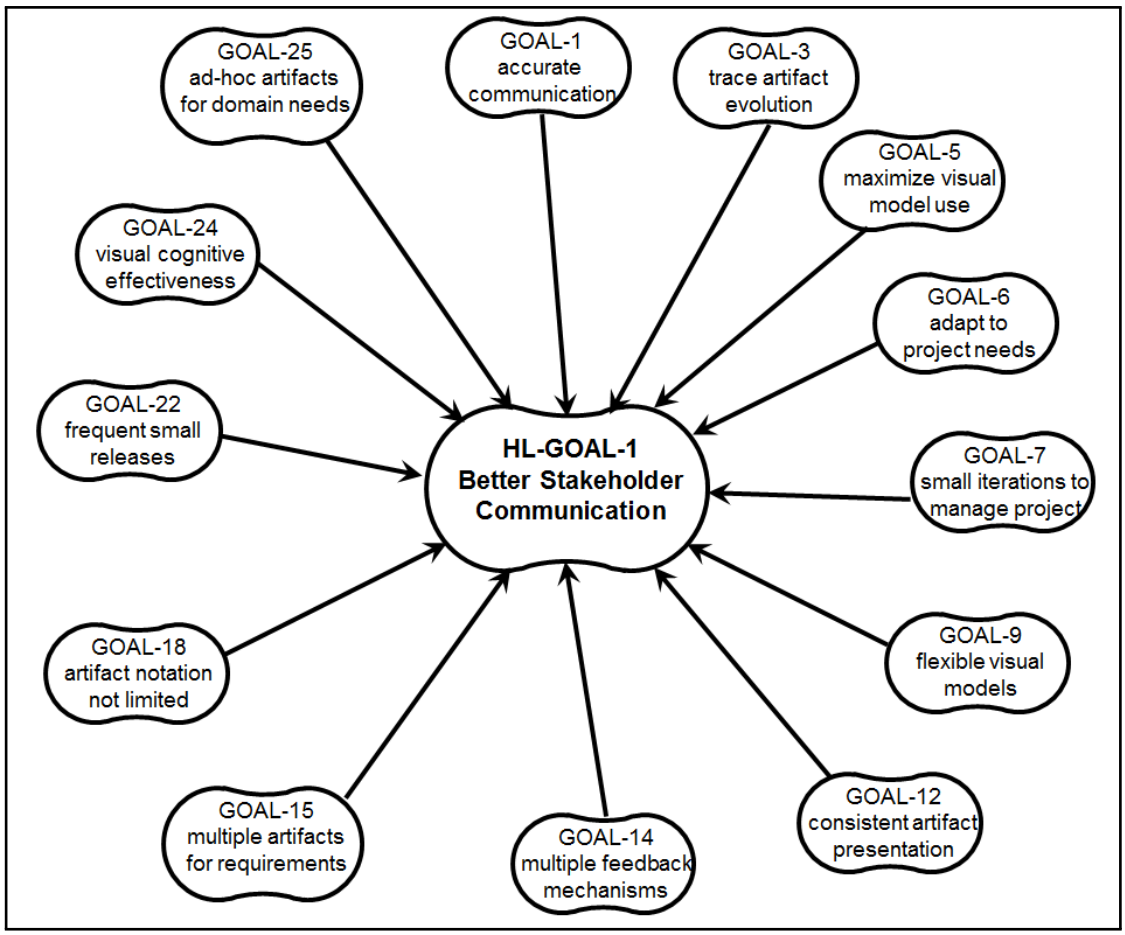


Figure 7.1: Reproduction of Figure 2.8.

Result: goal achieved

Questions: CS-Q5/DS-Q3, CS-Q6/DS-Q4, CS-Q7/DS-Q5, CS-Q8/DS-Q6, CS-Q10/DS-Q8, CS-Q11/DS-Q9, CS-Q13/DS-Q11, CS-Q15/DS-Q12, CS-Q18/DS-Q14, CS-Q20/ DS-Q18, and CS-Q23/DS-Q24.

Ordinal Values Summary: Highest: **37/44**, Second Highest: **8/44** (relative values since ranges differ, denominator is the number of questions times the respondents, or 11 x 4, sometimes a question was not answered, so the sum of the numerators will not equal the denominator).

Quotes: In addition to the ordinal values picked by the respondents, supporting comments to indicate achieving the goal, if only partially, are listed in Table 7.7.

Table 7.7: ***HL-GOAL-2*** Supporting Survey Quotes.

Question	Response	Comment
DS-Q4	DR-1	Additional comments on the project management activities: It is a model. The Paper Prototype concept would work with another model, looking abstractly. I have to admit my ignorance of some of the RM-ODP concepts, and that would make the Paper Prototype as proposed a learning curve to implement. Still, I responded that having a framework is good for the process because the process needs bones.
	DR-3	Additional comments on the project management activities: It is hard to set aside enough time to go through the whole process in practice, so we only managed to do part of the process, meaning looking at some of the viewpoints but not all.
	DR-4	Additional comments on the project management activities: I'm not familiar with the listed tools.
DS-Q11	DR-1	Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
	DR-3	Additional comments on business information unit modelling: We didn't successfully use the state machine models.
DS-Q12	DR-1	Additional comments on the business flow model: If you can get the buy-in from the business to model their flow.
	DR-3	Additional comments on the business flow model: The usefulness depends a bit on who one is communicating with. There are people that just like things to be the traditional way.
DS-Q18	DR-3	Additional comments on solution design activities: As we didn´t have time to make use of all the views, we had some but not complete effect.

Improvements or future work: Some of the comments indicate that more explanation of and rationale for the RM-ODP framework is needed.

Related Detailed Requirements: The detailed goals related to ***HL-GOAL-2*** are shown in Figure 2.9, which is in Sub-chapter 2.4.1. Meeting ***HL-GOAL-2*** also means meeting the detailed goals. For convenience Figure 2.9 is reproduced below as Figure 7.2.

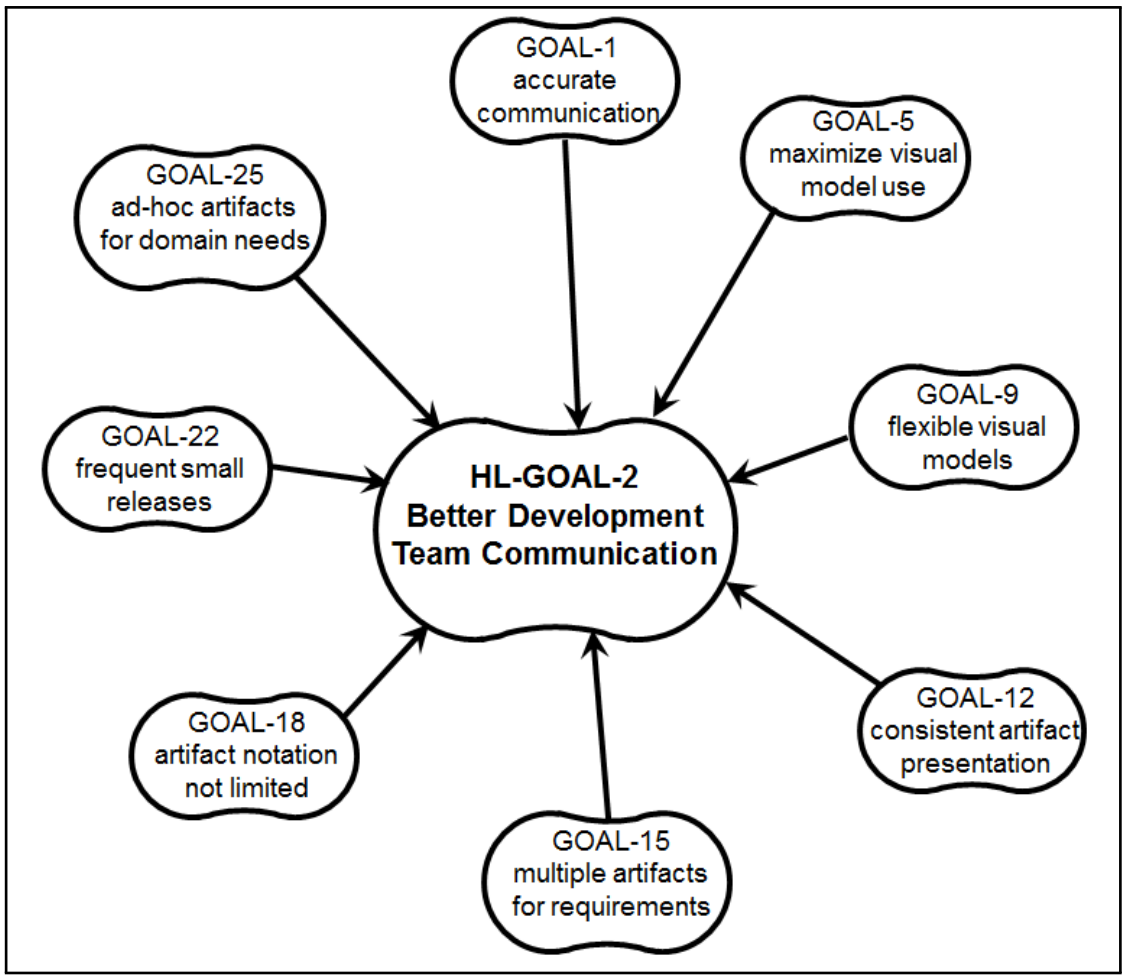


Figure 7.2: Reproduction of Figure 2.9.

HL-GOAL-3: Improved overall requirements quality in a more compact format.

Having higher quality requirements is a benefit to both the customers and developers. Quality in this case means that the requirements have clarity so they are not misunderstood, are correct with respect to the solution objectives, and contain no unneeded specifications.

The compactness of form for the requirements specification yields benefits in having less to review, less to comprehend, and less to update. A compact form also minimizes the chance for conflicts, as one does not get lost in the volume of information. There are different survey questions in both surveys, but also common questions.

The mechanisms that enable improved requirements quality and compactness are the various graphical models and visual representations, which take less space than text. The

RM-ODP viewpoints help separate true requirements from business goals, engineering design, and technology choices.

Result: goal achieved

Questions: CS-Q5/DS-Q3, CS-Q6/DS-Q4, CS-Q7/DS-Q5, CS-Q8/DS-Q6, CS-Q10/DS-Q8, CS-Q11/DS-Q9, CS-Q12/DS-Q10, CS-Q13/DS-Q11, CS-Q14, CS-Q16/DS-Q13, CS-Q17, CS-Q18/DS-Q14, CS-Q19/DS-Q15, CS-Q21/DS-Q22, and DS-16.

Ordinal Values Summary: Highest: **87/108**, Second Highest: **15/108** (relative values since ranges differ, denominator is the number of questions times the respondents, or (12 x 8 + 3 x 4), sometimes a question was not answered, so the sum of the numerators will not equal the denominator).

Quotes: In addition to the ordinal values picked by the respondents, supporting comments to indicate achieving the goal, if only partially, are listed in Table 7.8.

Table 7.8: ***HL-GOAL-3*** Supporting Survey Quotes.

Question	Response	Comment
CS-Q13 DS-Q11	CR-2	Additional comments on business information unit modelling: Once the team resources understood the state machine model, it was very helpful in the project definition process.
	CR-3	Additional comments on business information unit modelling: State machine model was not always clear to some end users.
	DR-1	Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
CS-Q19 DS-Q15	CR-2	Additional comments on achieving requirements quality, correctness and completeness: The specifications would have been much larger if a textual only approach was used.
	DR-1	Additional comments on achieving requirements quality, correctness and completeness: I must say that if images/models made it bigger that would be acceptable as well.

Improvements or future work: There were not many comments from the respondents relative to this goal and for which there is an explicit section in each survey. The

ordinal values were mostly the highest with some at the second highest score.

Related Detailed Requirements: The detailed goals supporting ***HL-GOAL-3*** are fulfilled through the fulfillment of the high-level goal, shown in Figure 2.10, which is in Sub-chapter 2.4.1. For convenience Figure 2.10 is reproduced below as Figure 7.3.

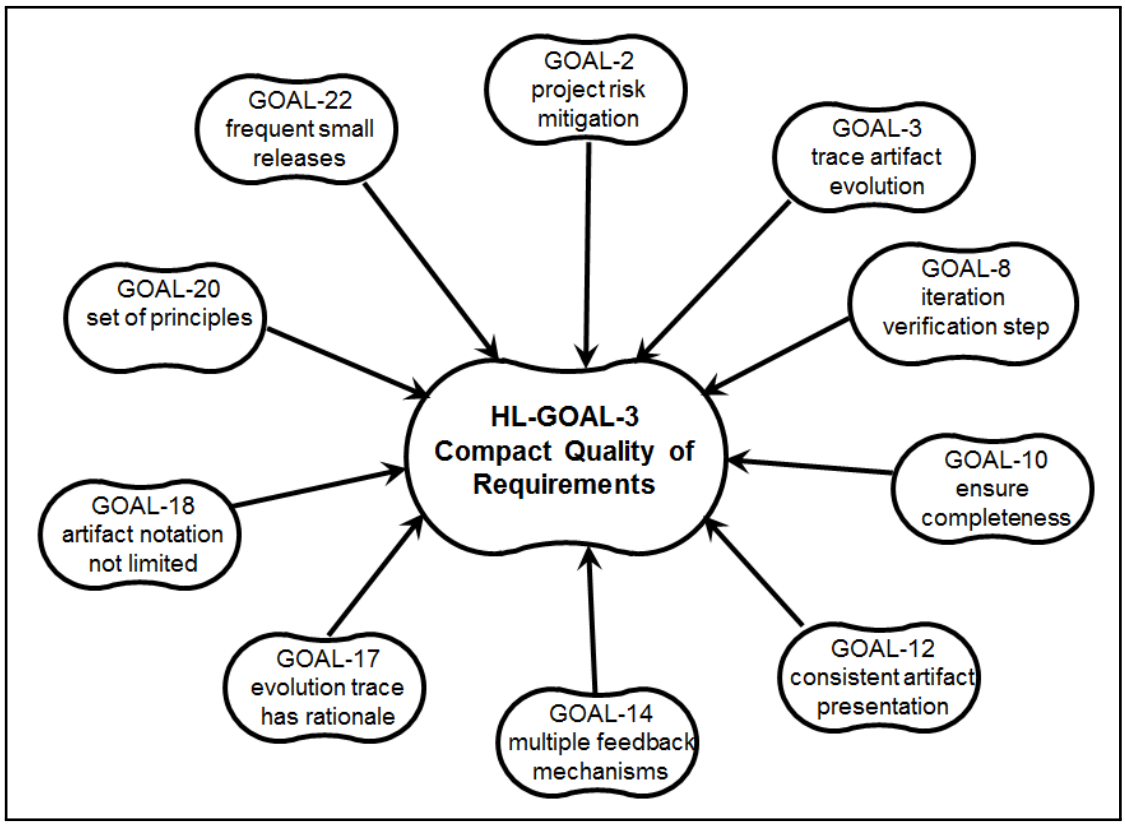


Figure 7.3: Reproduction of Figure 2.10.

HL-GOAL-4: Faster project process.

The**HL-GOAL-4** is a pragmatic project goal. The benefits of the SAMEM should reduce the work involved in producing and reviewing requirements. The improvements in requirements clarity should reduce the accidental complexity and simplify the work of moving to engineering design and technology implementations. The reduction in work is intended to speed up the project process.

A faster project process means lower cost for the customer and faster return on their investment, as the solution becomes usable sooner. The development team benefits by being able to do more projects in a given time.

The SAMEM mechanisms contributing to a fast project process are a smaller and clearer requirements specification, the RM-ODP abstraction framework to focus work on the appropriate areas, and the iterative & incremental approach to limit false work.

Result: goal achieved

Questions: CS-Q1/DS-Q1, CS-Q2/DS-Q2, CS-Q5/DS-Q3, CS-Q9/DS-Q7, CS-Q12/DS-Q10, CS-Q16/DS-Q13, CS-Q18/DS-Q14, CS-Q21/DS-Q22, CS-Q22/DS-Q23, CS-Q3, CS-Q17, DS-Q16, DS-Q17, DS-Q18, DS-Q19, DS-Q20, and DS-Q21.

Ordinal Values Summary: Highest: **85/104**, Second Highest: **17/104** (relative values since ranges differ, denominator is the number of questions times the respondents, or (9 x 8 + 8 x 4), sometimes a question was not answered, so the sum of the numerators will not equal the denominator).

Quotes: Additional comments from respondents beyond the ordinal evaluation.

Table 7.9: ***HL-GOAL-4*** Supporting Survey Quotes.

Question	Response	Comment
CS-Q1 DS-Q1	CR-1	Additional comments: I can't imagine proceeding without it.
	CR-2	Additional comments: I believe the process allows for a quick verification of the process prior to investing in technical solutions and adding complex elements.
	DR-1	Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
	DR-3	Additional comments: The effectiveness doesn't necessary have to do with the paper prototype, but rather the number of people that we had to include in the process. We are changing the way we deploy the process in the second iteration by first having a small group and then taking it to the larger group once its stable. This way it becomes more manageable. We also doing reviews with smaller groups and rather more review meetings.
	DR-4	Additional comments: Paper prototypes were an effective way to visually communicate how a solution is intended to work, is a great tool to determine if the solution meets a customer's needs, and can be used to improve or refine a solution.

Question	Response	Comment
CS-Q5 DS-Q3	CR-3	Allowed agreement that this was what was desired before extensive investment in program configuration.
CS-Q12 DS-Q10	DR-2	Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
DS-Q18	DR-3	Additional comments on solution design activities: As we didn´t have time to make use of all the views, we had some but not complete effect.
DS-Q21	DR-3	Additional comments on testing activities: As we didn´t manage to use all views, we didn't have a consistent process so we experienced various effect. Sometimes the discussion got diverted into what to capture and why, rather than using the information provided.
CS-Q9 DS-Q7	CR-1	Additional comments on overall graphical representation use: I don't want to minimize the total effort in developing a good project plan.
	CR-2	Additional comments on overall graphical representation use: Enabled a common framework for alignment of the team on over-all project direction.
	CR-4	Additional comments on overall graphical representation use: A picture that describes the process is always a good way to con-firm that all participants are understanding the transactional processes in the same way.
	DR-1	Additional comments on overall graphical representation use: It can have various effects on the planning process depending on the players and how they understand the 1-pager. It certainly im-proves planning, but can make the interactions in the meetings strained depending on how one views 1-page to equal simple.
	DR-3	Additional comments on overall graphical representation use: We found that its important that the person presenting the 1-pager have a good way of communicating it. If not, the discussion ends up focusing on what the 1-pager is and is showing rather than its content.

Improvements or future work: In general, the SAMEM helped the process move along, but in some cases the communication was less than optimal. This implies a better training of the participants in the process and artifacts about their purposes.

Related Detailed Requirements: The detailed goals supporting ***HL-GOAL-4*** are fulfilled through the fulfillment of the high-level goal, shown in Figure 2.11, which is in Sub-chapter 2.4.1. For convenience Figure 2.11 is reproduced below as Figure 7.4.

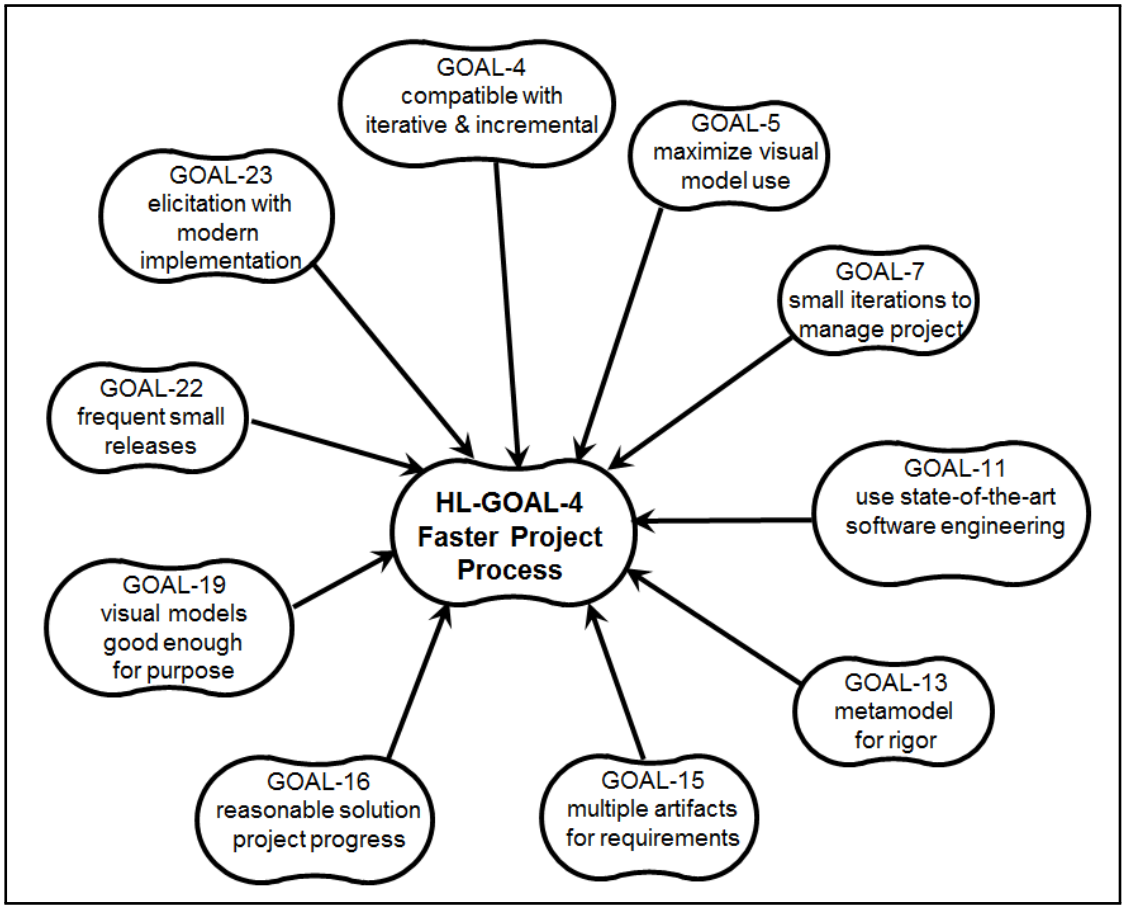


Figure 7.4: : Reproduction of Figure 2.11.

HL-GOAL-5: Smaller project team size, especially for the development team.

The rationale behind the high-level goal of a smaller project team size is based on the Mythical Man-Month observation by Brooks [Bro95]. Minimizing the size of the team, especially the development team, minimizes the design, implementation, and testing

communication overhead in the project. The goal of a small team size must be balanced with the ***HL-GOAL-4*** objective of a faster project process. The team can grow without Mythical Man-Month impacts when there are tasks that are independent and can be accomplished in parallel with little to no communication.

There are several mechanisms that can help to minimize the development team size and make the project process more effective. The clarity of the requirements is critical, especially the understanding of what not to create. Requirements clarity and completeness minimizes communication, which assists in determining possible parallel work. The RM-ODP viewpoints organize the work so that the effort is directed to the essential tasks at the optimum time. The iterative & incremental approach allows for smaller partitions of work that can enable more parallel tasks. Achieving the ***HL-GOAL-5*** is dependent on the actual planning of the project process; however, the SAMEM tries to open the possibility to reduce team size when compared to other methodologies.

Result: goal achieved

Questions: CS-Q1/DS-Q1, CS-Q2/DS-Q2, CS-Q9/DS-Q7, CS-Q16/DS-Q13, CS-Q18/DS-Q14, CS-Q20/DS-Q18, CS-Q22/DS-Q23, CS-Q23/DS-Q24, DS-Q16, and DS-Q17.

Ordinal Values Summary: Highest: **54/72**, Second Highest: **16/72** (relative values since ranges differ, denominator is the number of questions times the respondents, or (8 x 8 + 2 x 4), sometimes a question was not answered, so the sum of the numerators will not equal the denominator).

Quotes: For some questions in the surveys clarifying statements were made by the respondents.

Table 7.10: ***HL-GOAL-5*** Supporting Survey Quotes.

Question	Response	Comment
CS-Q1 DS-Q1	CR-1	Additional comments: I can't imagine proceeding without it.
	CR-2	Additional comments: I believe the process allows for a quick verification of the process prior to investing in technical solutions and adding complex elements.
	DR-3	Additional comments: The effectiveness doesn't necessary have to do with the paper prototype, but rather the number of people that we had to include in the process. We are changing the way we deploy the process in the second iteration by first having a small group and then taking it to the larger group once its stable. This way it becomes more manageable. We also doing reviews with smaller groups and rather more review meetings.
	DR-4	Additional comments: Paper prototypes were an effective way to visually communicate how a solution is intended to work, is a great tool to determine if the solution meets a customer's needs, and can be used to improve or refine a solution.

Question	Response	Comment
CS-Q20 DS-Q18	DR-3	Additional comments on solution design activities: As we didn´t have time to make use of all the views, we had some but not complete effect.
CS-23 DS-Q24	CR-2	Additional comments on the incremental & iterative process im-pacts: I believe it sets up a smooth transition to the technical team for delivery. There are still some opportunities to move along more quickly with the steps and perhaps even break the process into smaller components.
CS-Q9 DS-Q7	CR-1	Additional comments on overall graphical representation use: I don't want to minimize the total effort in developing a good project plan.
	CR-2	Additional comments on overall graphical representation use: Enabled a common framework for alignment of the team on over-all project direction.
	CR-4	Additional comments on overall graphical representation use: A picture that describes the process is always a good way to con-firm that all participants are understanding the transactional processes in the same way.
	DR-1	Additional comments on overall graphical representation use: It can have various effects on the planning process depending on the players and how they understand the 1-pager. It certainly im-proves planning, but can make the interactions in the meetings strained depending on how one views 1-page to equal simple.
	DR-3	Additional comments on overall graphical representation use: We found that its important that the person presenting the 1-pager have a good way of communicating it. If not, the discussion ends up focusing on what the 1-pager is and is showing rather than its content.

Improvements or future work: Outside of the positive comments, there are comments that indicate that education about the process and artifacts can be improved. There are indications that software engineer does not equal software engineer, which means that people's various strengths and experiences have an impact on picking the correct person for a task to ensure success.

Related Detailed Requirements: The detailed goals supporting ***HL-GOAL-5*** are fulfilled through the fulfillment of the high-level goal, shown in Figure 2.12, which is

in Sub-chapter 2.4.1. For convenience Figure 2.12 is reproduced below as Figure 7.5.

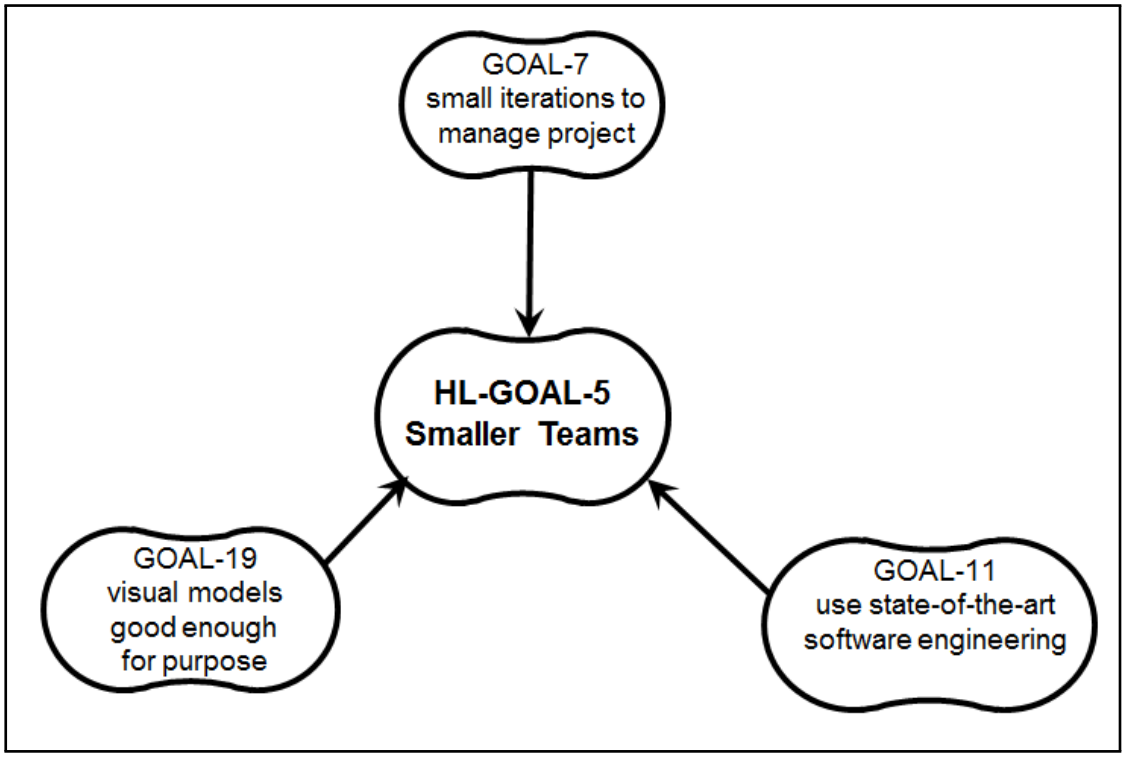


Figure 7.5: Reproduction of Figure 2.12.

HL-GOAL-6: Compatibility with Agile methods for solution (code) development.

Agile or iterative & incremental process approaches have been used with great success at the coding level. The SAMEM extends those successes to include the full project scope starting with solution concept and requirements. The SCRUM idea of Agile uses User Stories (in the general sense) to define the work tasks for the people. Compatibility with Agile methods implies that the requirements and design work can be broken into User Stories and that the requirements and design artifacts flow into development User Stories.

The mechanisms that support achieving this goal work in a supporting manner. The RM-ODP framework guides the separation of the "What" questions at one level of abstraction from the "How to" questions at the lower level, so that User Stories can be formed to produce answers. The graphical modeling produces artifacts that can be partitioned into smaller units with clearer interfaces for expression as User Stories. A

consistent project process approach for the complete project lifecycle creates a common team work environment.

Although the ***HL-GOAL-6*** is primarily oriented towards developers, stakeholders are impacted in the development of artifacts that flow into the design work and in the verification of the implementations produced. The DS-Q25 specifically asks the developers whether the requirements iterative & incremental work cleanly flows into the development work and the answer is "Yes Better" by all respondents.

Result: goal achieved

Questions: CS-Q2/DS-Q2, CS-Q9/DS-Q7, CS-Q10/DS-Q8, CS-Q20/DS-Q18, CS-Q22/DS-Q23, CS-Q23/DS-Q24, CS-Q3, DS-Q16, DS-Q17, DS-Q19, DS-Q20, and DS-Q25.

Ordinal Values Summary: Highest: **60/76**, Second Highest: **16/76** (relative values since ranges differ, denominator is the number of questions times the respondents, or (6 x 8 + 7 x 4), sometimes a question was not answered, so the sum of the numerators will not equal the denominator).

Quotes:

Table 7.11: ***HL-GOAL-6*** Supporting Survey Quotes.

Question	Response	Comment
CS-Q3 DS-Q1	CR-1	In which ways? People could understand what they were getting very early in the process.
	CR-3	In which ways? Improved understanding of project and establishment of requirements.
	DR-1	Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
CS-Q20 DS-Q18	DR-3	Additional comments on solution design activities: As we didn´t have time to make use of all the views, we had some but not complete effect.
CS-23 DS-Q24	CR-2	Additional comments on the incremental & iterative process im-pacts: I believe it sets up a smooth transition to the technical team for delivery. There are still some opportunities to move along more quickly with the steps and perhaps even break the process into smaller components.
DS-Q25	DR-1	Additional comments on the incremental & iterative process im-pacts: This concept (Paper Prototype) fits cleanly into agile.

Question	Response	Comment
CS-Q9 DS-Q7	CR-1	Additional comments on overall graphical representation use: I don't want to minimize the total effort in developing a good project plan.
	CR-2	Additional comments on overall graphical representation use: Enabled a common framework for alignment of the team on over-all project direction.
	CR-4	Additional comments on overall graphical representation use: A picture that describes the process is always a good way to con-firm that all participants are understanding the transactional processes in the same way.
	DR-1	Additional comments on overall graphical representation use: It can have various effects on the planning process depending on the players and how they understand the 1-pager. It certainly im-proves planning, but can make the interactions in the meetings strained depending on how one views 1-page to equal simple.
	DR-3	Additional comments on overall graphical representation use: We found that its important that the person presenting the 1-pager have a good way of communicating it. If not, the discussion ends up focusing on what the 1-pager is and is showing rather than its content.

Improvements or future work: From the comments and the ordinal answer values, there are not many improvements to be made on the transitioning to agile style development processes.

Related Detailed Requirements: The detailed goals supporting HL-GOAL-6 are fulfilled through the fulfillment of the high-level goal, shown in Figure 2.13, which is in Sub-chapter 2.4.1. For convenience Figure 2.13 is reproduced below as Figure 7.6.

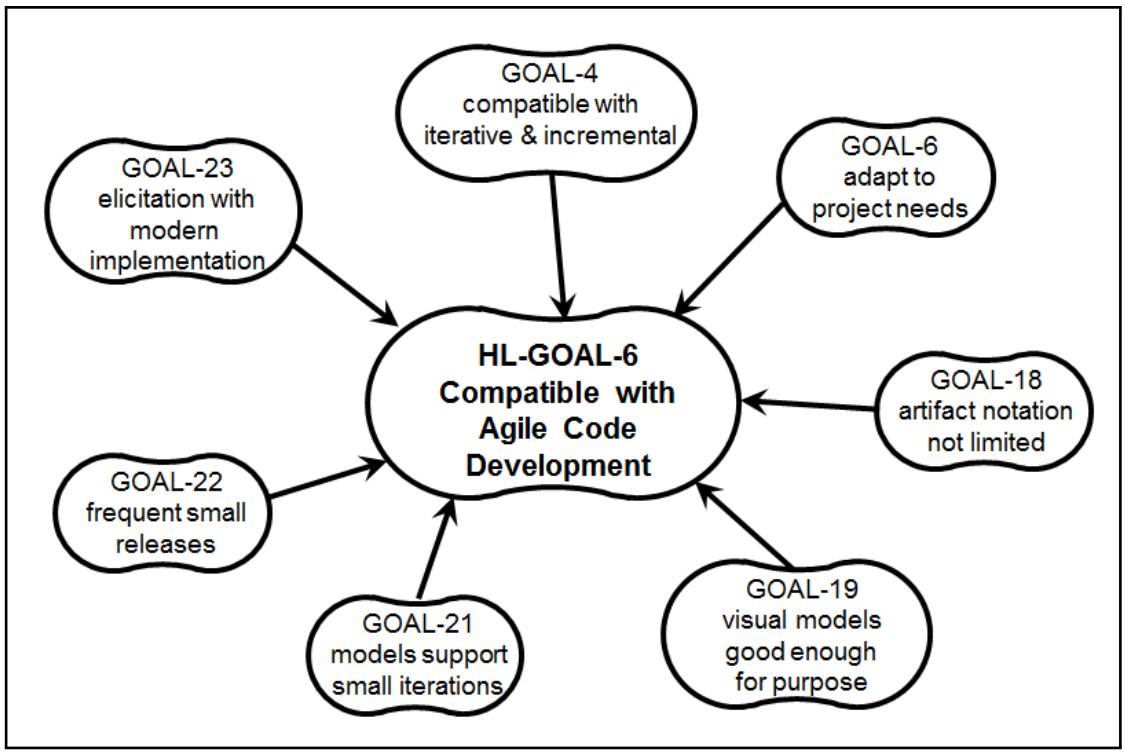


Figure 7.6: Reproduction of Figure 2.13.

7.1.4.1.1 Compatibility with Agile Code Development and Code Generation As discussed in Sub-chapter 4.3.5.4, the SAMEM is not targeted at RM-ODP Engineering and Technology Viewpoints. These are the areas where the solution architecture decisions are made, the technology choices are made, and where code development happens. As discussed in [Rum17], modeling can be used in an agile manner to generate code, tests, and help with refactoring. The modeling needed to accomplish these types of generation must be more precise than the modeling described in this thesis for requirements specification. The precise modeling language for generation is exemplified by UML/P [Rum16]. Many DSMLs can be candidates as well.

Model transformations can be envisioned that convert the SAMEM-IM and SAMEM-PM compliant models into initial *UML/P* artifacts and models. Because there is an abstraction gap between the SAMEM models and the *UML/P*, the transformation will be incomplete. Design decisions at the *UML/P* level will be needed to complete the models. The model transformations from the SAMEM to *UML/P* will generate traceability between the two levels and rules can be defined to support round-trip engineering in the case that a code generating model indicates a possible requirements weakness.

It is outside the scope of this thesis to pursue the SAMEM to *UML/P* model transformations in more detail, but it is a topic for future work.

7.1.4.2 Supporting Evidence and Evaluation for Detailed Goals

The detailed SAMEM goals from Sub-chapter 2.4.1 are listed below for convenience. Following each goal is an explanation of its fulfillment, either through the evidence collected by the survey of case study participants or through the design of the SAMEM. There can be multiple survey questions that support the fulfillment of the goal. The ordinal answer values are listed in either Table 7.3 or Table 7.4 and will not be repeated here.

GOAL-1: The methodology should provide accurate communication mechanisms.

The survey questions CS-Q7/DS-Q5, CS-Q8/DS-Q6, CS-Q18/DS-Q14, and CS-Q17 directly address the fulfillment of this requirement. The answers are overwhelmingly positive.

GOAL-2: The methodology should provide project process risk mitigation mechanisms.

The answers to survey questions CS-Q1/DS-Q1, CS-Q2/DS-Q2, CS-Q3, and CS-Q4 are all positive and are related to confidence in completing the project. From a SAMEM design perspective, the RM-ODP abstraction framework assists in ensuring the most impactful project questions are addressed first. Several of SEFP guidelines are intended to reduce project risk. Achieving the professional engineering practice of due diligence is also supported.

GOAL-3: The methodology should provide traceable artifact evolution.

There are several survey questions that address change management, CS-Q15/DS-Q12, CS-Q20/DS-Q18, CS-Q4, and DS-Q21. Although all the answers are largely positive, the management of artifact evolution is more a factor of the tools for storing the various artifact versions and project policies for controlling updates.

GOAL-4: The methodology and artifacts should be compatible with an iterative & incremental project process.

The few survey questions that address this goal in parts are CS-Q12/DS-Q10, CS-Q23/DS-Q24, and DS-Q25. The graphical nature of the requirements, engineering design, and technology design artifacts lend themselves to iteration partitioning easier than text-based representations.

GOAL-5: The artifacts should maximize the use of visual models for communication and compactness.

The primary survey questions that are directed towards the value of graphical artifacts are CS-Q5/DS-Q3, CS-Q7/DS-Q5, and CS-Q10/DS-Q8. The answers strongly support the benefits of graphical over textual representations for as many artifacts as possible.

GOAL-6: The methodology should be adaptable to individual project process needs.

The adaptability of the SAMEM to project process needs is mostly a factor of general guidelines within the SAMEM design rather than specific prescriptions. There are three areas of guidelines: the SEFP, the use of RM-ODP, and the iterative & incremental approach. The survey questions that investigate the benefits of the RM-ODP and iterative & incremental aspects are CS-Q2/DS-Q2, CS-Q6/DS-Q4, and CS-Q20/DS-Q18.

GOAL-7: The iterations should be small for project management purposes.

Small iterations have positive consequences for the project control. Small iterations make it easier to estimate the work effort, enable frequent verification that work is on track, and show progress towards the project goals. The survey questions that deal with small iteration consequences are CS-Q2/DS-Q2, CS-Q22/DS-Q23, and CS-Q3.

GOAL-8: An iteration should have a verification step of some kind to assure that the work done is correct.

The rationale for the verification step as an integral part of an iteration is a SAMEM design decision. The purpose of the verification step is to ensure that the work stays true to the solution concept at the highest level, the requirements, and any design decisions. Passing the verification step is the closure point for an iteration, so that work can progress to the next task. There are not any survey questions that deal directly with this goal.

GOAL-9: Flexibility in the visual model artifacts is necessary for communication optimization.

In order to enable the SAMEM to adapt to different domains, the model artifacts must adapt to the domain information and behavior needs. The SAMEM definition does not require any specific modeling forms for communication. In the case studies, a variety of artifacts were effectively used for communication as reflected in questions CS-Q5/DS-Q3, CS-Q8/DS-Q6, CS-Q11/DS-Q9, CS-Q13/DS-Q11, and CS-Q15/DS-Q12.

GOAL-10: The methodology should ensure completeness.

Completeness in this sense means that the definitions of the solution and the requirements specification have the necessary and sufficient artifacts to fulfill the solution concept. It does not mean that definition and specification must be complete before starting any realization work. The related survey questions are CS-Q1/DS-Q1, CS-Q17, and DS-Q16.

GOAL-11: The methodology should support the incorporation of state-of-the-art software engineering results.

The essence to achieving this goal is in avoiding prescriptive mechanisms in the definition of the SAMEM. This goal is evolutionary in that it means that the SAMEM needs to be able to incorporate new ideas and practices that arise in the future. The general guidelines of the SAMEM that assist in meeting this goal are the abstraction framework, currently the RM-ODP, the list of Software Engineering First Principles, which can be extended, and the iterative & incremental task style. The key to satisfying this goal is to design the SAMEM with as few restrictions as possible.

GOAL-12: The visual modeling and textual artifacts need to have consistent presentation to optimize communication.

This goal embodies some good cognitive effectiveness rules. When people need to relearn the symbols used in the models, then communication suffers from wasted time and confusion. Several survey questions are about the consistent use of communication mechanisms for the case studies: CS-Q8/DS-Q6, CS-Q11/DS-Q9, CS-Q12/DS-Q10, CS-Q18/DS-Q14, and CS-Q29/DS-Q31.

GOAL-13: A metamodel is needed for the artifacts and process to ensure a rigorous methodology.

With a metamodel for the SAMEM, artifact tools can be constructed. The tools can be computer-aided design and construction tools for creating and updating the SAMEM artifacts and computer-aided analysis tools for checking that the artifacts are well-formed and complete. This is a design goal and there are no empirical evidence questions for this goal.

GOAL-14: The methodology must support multiple feedback mechanisms.

The survey questions that examine the quality of feedback in the case studies are CS-Q2/DS-Q2, CS-Q7/DS-Q5, CS-Q10/DS-Q8, CS-Q11/DS-Q9, CS-Q15/DS-Q12, and CS-Q20/DS-Q18.

GOAL-15: The methodology should allow for multiple requirement artifacts to match the different communication needs of the different levels of abstraction needed during the project process.

In the case studies, several different requirements artifact forms were used. The feedback was positive on the questions that evaluated the different forms. The survey questions are CS-Q8/DS-Q6, CS-Q10/DS-Q8, CS-Q12/DS-Q10, and CS-Q13/DS-Q11.

GOAL-16: The methodology project process must ensure that progress towards a solution is accomplished at a reasonable speed.

It is obvious that *reasonable* in the statement of this goal is relative to the domain, the expectations of the stakeholders, and the complexity of the solution scope. On the

basis of the case study survey questions, the answer is yes. The following questions are pertinent: CS-Q1/DS-Q1, CS-Q22/DS-Q23, CS-Q3, and DS-Q16.

GOAL-17: The artifact evolution trace should have possibilities for recording the rationale for the artifact improvement.

Fulfilling this goal is mostly a matter of having an artifact revision control tool in place to capture the new version or increment of the artifact and the reason for the new version. For the case studies, the Apache Subversion (https://subversion.apache.org/) revision control tool was used. The use of the versioning tool is mostly an aspect of the specific project management rules. The SAMEM does not mandate any specific rules one way or the other, but good engineering practices encourage the recording of decisions and changes.

GOAL-18: The visual artifacts must not be limited by existing notations such as UML, alt-hough, when appropriate, existing notations should be preferred.

The domain and the specific solution area within the domain will guide the project decisions for the artifact forms. There are no specific survey questions for this goal. From the case studies, examples of extending UML and using non-UML notations are shown in Sub-chapters 3.5.1.1, 3.5.2.1, 3.5.3.1, 3.5.4.1, and 3.5.5.1 and discussed in 4.4.1.5.1, and 4.4.1.6.

GOAL-19: The visual models must be good enough to achieve the purpose within the current context or step of the project process.

The surveys have questions about the effectiveness of the visual models for both the stakeholders and the developers. The questions that concern the evaluation of the success of the visual models are CS-Q5/DS-Q3, CS-Q7/DS-Q5, CS-Q8/DS-Q6, CS-Q10/DS-Q8, CS-Q11/DS-Q9, CS-Q13/DS-Q11, CS-Q15/DS-Q12, CS-Q18/DS-Q14, CS-Q17, and DS-Q16.

GOAL-20: The methodology should have a set of principles that support adaptation to various domains and provide a checklist for rigor.

This goal establishes a design constraint on the SAMEM. The Software Engineering First Principles as defined in Sub-chapter 3.2 are embodiment of the design constraint specified by this goal.

GOAL-21: The models used for artifacts in the methodology should support multiple small iterations.

The evidence supporting this goal is indirect from the survey questions, as there were no direct questions on the fulfillment of this goal. The case studies used small iterations

and visual models as the artifacts. The questions that imply that this goal is fulfilled are CS-Q2/DS-Q2, CS-Q12/DS-Q10, and DS-Q25.

GOAL-22: The methodology should support frequent small releases of project deliverables to support verification of progress.

The iterative & incremental approach is the basis for producing frequent small releases. The survey questions that gathered the empirical evidence that this goal is achieved are CS-Q2/DS-Q2, CS-Q21/DS-Q22, CS-Q22/DS-Q23, CS-Q3, and DS-Q25.

GOAL-23: The requirements elicitation portion of the methodology should be consistent with modern implementation best practices.

The main modern implementation practices are those based on Agile and SCRUM methods. The Developer Survey question DS-Q25 is specifically targeted to provide the evidence and the answer is 100% yes.

GOAL-24: The methodology artifact representations should be consistent with the best practices of communication as measured through cognitive effectiveness.

While there are no questions about the cognitive effectiveness of the visual modeling techniques, there are questions in the survey about the effectiveness of the communication. As evidence for meeting this goal, Sub-chapter 7.3 will contain a detailed analysis of the graphics used and their effectiveness. The survey questions about communication effectiveness are CS-Q5/DS-Q3, CS-Q7/DS-Q5, CS-Q8/DS-Q6, CS-Q11/DS-Q9, and CS-Q15/DS-Q12.

GOAL-25: The methodology should allow for the ad-hoc creation of artifacts for domain adaption and communication improvements.

The SAMEM does not define or recommend any specific visual modeling mechanisms, such as UML. The emphasis is on effective communication; as such anything that works is acceptable. Examples of the survey questions that ask about the effectiveness of combinations of modeling techniques are CS-Q5/DS-Q3, CS-Q8/DS-Q6, CS-Q11/DS-Q9, and CS-Q18/DS-Q14. In addition, the case studies with examples of extending UML and using non-UML notations are shown in Sub-chapters 3.5.1.1, 3.5.2.1, 3.5.3.1, 3.5.4.1, and 3.5.5.1 and discussed in 4.4.1.5.1, and 4.4.1.6.

7.1.5 General Opinions from the Case Study Surveys

Three common questions appear in the case study surveys for an evaluation of the pre-SAMEM methodology from an overall perspective, somewhat independent of the solution domain. The intent of the three overall questions was an evaluation of the methodology

from a step back from the details, since the details can be adjusted for improvements. The three questions are CS-Q1/DS-Q1, CS-Q24/DS-Q26, and CS-Q25/DS-Q27.

Survey question CS-Q1/DS-Q1 concerns the overall evaluation of the methodology with respect to the quality of the solution. This can be viewed as a summation of the positive and negative aspects with the final evaluation being positive. The most positive answer on the scale is "Very Effective: good solution with a minimum number of people in a minimum time frame," which was picked by 87% of the respondents. The second most positive answer, "Moderately Effective: good solution with a reasonable number of people in a reasonable time frame," was picked by the other 13% of the respondents. Several comments were given by the respondents and are listed below with the answer value:

- Very Effective: "I can't imagine proceeding without it."
- Very Effective: "I believe the process allows for a quick verification of the process prior to investing in technical solutions and adding complex elements."
- Moderately Effective: "The effectiveness doesn't necessary have to do with the paper prototype, but rather the number of people that we had to include in the process. We are changing the way we deploy the process in the second iteration by first having a small group and then taking it to the larger group once it's stable. This way it becomes more manageable. We also doing reviews with smaller groups and rather more review meetings."
- Very Effective: "Paper prototypes were an effective way to visually communicate how a solution is intended to work, is a great tool to determine if the solution meets a customer's needs, and can be used to improve or refine a solution."

A general inquiry about any weakness or shortcoming is the purpose of question CS-Q24/DS-Q26. The possible answers are "Yes," picked by 38%, or "No," chosen by 62%, with space for comments. Comments about weaknesses given by the respondents, when given, are listed below with the answer value:

- No: "Gathering requirements is simply a matter of getting to the truth about what has to be delivered and in gaining consensus on the solution. Any means will do that if the participants are knowledgeable and engaged. With a visual graphical approach, it makes the participants understand what is happening quickly and facilitate learning as well."
- No: "I have always had some tables, flows and pictures in specification so for me it fit with and extended the model I thought worked well for spec development."
- Yes: "The Paper Prototype process is very useful in nailing down many of the requirements, there is still the need for expressing some of the requirements in a traditional fashion (e.g., nonfunctional requirements)."

- Yes: “Modelling takes time and addressing the necessary RM-ODP views requires experience in modelling to be successful within the time available. Therefore, it might be useful to have a paper prototype tool to support the process.”

- Yes: “If the Paper Prototype was not clear, it left too much room for interpretation by the customer. This could lead to a solution that might not address the customer’s expectations.”

Survey question CS-Q25/DS-Q27 is the inverse of CS-Q24/DS-Q26 and asks about strengths and benefits. The choices are “Yes,” picked by 87%, or “No,” selected by 13%, with space for comments. Comments about strengths given by the respondents, when given, are listed below with the answer value:

- Yes: “Easier to digest, manage and handoff to other teams.”

- Yes: “One major advantage was mapping processes, especially since most parts of the organization did not have those and did not have clarity on the extent of interplay and interactions with other parts of the organization. Also enabled communication to management, to ensure all parties were on the same page before committing resources.”

- Yes: “A definition to defend.”

- Yes: “Any tool that can help the user "see" and provide feedback on the proposed solution with a minimal time/investment is helpful. The Paper Prototype process is such a tool that can be effective with certain types of development efforts.”

- Yes: “The paper prototype process forces you to think about the solution in a very concrete manner and help you visualize what you are building so that you can easier align between the team members to ensure they all agree on what it is that we are building.”

- Yes: “The Paper Prototype could be used for many purposes: developing test cases, creating a documentation plan, estimating tasks, developing regression test cases, and drafting procedures.”

In summary, the early version of the SAMEM, also known as the Paper Prototype Process, is considered to be an effective project methodology. The major strengths are bringing the team together with a single vision and stimulating deeper thinking about the problem and the possible solutions. The weaknesses highlight the need for additional skills on the team, such as modeling and the addressing of requirements that are not easily modeled. The requirements not easily modeled, such as the typical non-functional requirements, are an extension in this thesis over the early version of the SAMEM used in the case studies.

7.2 Other Applications of the SAMEM

There are three additional uses of the SAMEM currently started as of the date of this thesis writing. While the project history is very limited, there are statements from participants that indicate preliminary benefits to using the SAMEM approach.

7.2.1 Start-up Company for Mobile Application Development

The start-up company has an idea for a new social connection application focused on cellular technology-based mobile device use, i.e. phones and tablets. The initiator (J.L.) of the idea has significant experience, over 20 years, in running non-software companies, but no experience in running a software-based project. A mutual acquaintance introduced the author of this thesis to J.L.

The author had several tutoring sessions with J.L. to explain the SAMEM, which was then applied with the author leading J.L. through the process. The initial focus was on the Enterprise Viewpoint to clearly express the idea and value. The second step was the elicitation of preliminary requirements of the behavior and information models. An alternative expression of the behavior of the application via the use of storyboards and user interface sketches is used as compared to the case study companies. The quote from J.L. below shows high satisfaction with the process and the beginning explorations into engineering and technology design alternatives.

> "Using the Software Agile Modeling and Engineering Methodology (SAMEM) to produce the architecture for our software development project created discipline and stability to complete an organized structure as we worked through RM-ODP viewpoints in a very short period of time. Considering the non-linear nature of most software projects and the frequent revisions in the agile or iterative methods, which encompass content and time, SAMEM exceeded its purpose, created maximum efficiency and definitely limited "momentum" in the wrong direction. The process quickly identified questions to be addressed and if outside support or input was needed. Furthermore, the use of simple visual aids and informality accelerated the ability to produce the more technical engineering aspects."

7.2.2 Company Specializing in Employee Background Checks

The Paper Prototype Process (PPP), as the early version of the SAMEM was called, was introduced in a company that specializes in employee background checks and other security offerings by a developer that worked on the Case Study 1 (CS-1) solutions. The CS-1 developer is in the position of Senior Application Developer with the Employee Background Check (EBC) company.

The motivation for introducing PPP was to mitigate the negative effects and risks on the current project caused by poor requirements, lack of design activities, and ineffective communication. While the EBC company was trying to use agile and SCRUM

practices, they were failing by their own admittance. The CS-1 developer realized that the incomplete and textual nature of the requirements used for design communication was a major cause for the project process problems. The most significant problem was the rework caused by poor quality user stories. The rework involved re-definition of the user stories, caused by redefinition of the requirements, redesign, and recoding.

The major component of the PPP that the CS-1 developer introduced was the idea of modeling the requirements and the design, then using the models to drive the agile iterations or SCRUM sprints. The introduction took place in the middle of the project, which is an enhancement to an existing internal solution, as a mitigation and correction mechanism. The future intentions are to use more components from the PPP or SAMEM as a standard part of the architectural design for new projects.

Below are two quotes from members of the EBC project team as to the benefits realized by the PPP and by extension of the SAMEM.

> "As a junior developer, I felt the paper prototype process fostered my learning better than a strictly agile process. I was able to understand all the components of a project and, in return, contribute in a meaningful way. It brought forth better collaboration and organization within the team and more accurate time assessments."

> "As a principal application developer part of my responsibilities have been to oversee the stability of our system and ensure enhancements are well thought through. When the agile process was introduced to this IT group we quickly recognized the "design" phase was lost. It was very difficult to make large system enhancements or rewrite a major portion of the system because stories weren't well defined which left developers and QA to guess at what the requirements truly were. We were seeing stories roll over and our system starting to degrade because of not having well defined stories. I am currently working on a project that took time to put the "design" phase back into the process. So far this has eliminated confusion within the team because everyone has participated in the design and fully understands our goals. This process has also helped define stories in a way that are easier to point, estimate hours and prepare for planning which gives business a better delivery date. So far putting the design phase back into the process has shown to be invaluable."

Future work within the EBC company on process improvement will be to use more modeling techniques to define solution architectures and requirements. The goal is to achieve a better understanding of the product needs to eliminate the miscommunication that is currently happening. They want to grow the success beyond the current project to make a companywide improvement.

7.2.3 Company Creating Safety and Security Solutions for the Oil and Gas Industry

Another company (referred to as SEC in this thesis) that is starting to apply the ideas of the SAMEM creates software products for safety and security needs for oil and gas companies. The products address the needs for safety and security on drilling rigs and in pipelines. There are multiple players in this domain including standardization groups and government regulatory agencies. The SEC company also sees the use of modeling as a mechanism to help define and communicate the standards and regulations

The SEC company started to introduce the SAMEM ideas during the middle of an existing development project, which focused on enhancements to a current product. The development environment has developers distributed over several continents and time zones. The earlier development process was a waterfall type. The development was starting to fracture resulting in developers creating incompatible components. The incompatibilities were in interface definitions, inconsistencies with the requirements, and inconsistent coding styles. The incompatibilities created unacceptable rework costs.

The portions of the SAMEM that are most useful in correcting the situation are the use of models for lighter weight specifications and an iterative & incremental development project process. The use of UML-like models for interfaces and defining detailed design modules are improving the communication among the developers. The interface incompatibles have essentially disappeared. Moving to a more iterative & incremental project approach has improved deviations from the requirements through more frequent checks at the end of the iteration.

Future work in SEC includes the application of more of the SAMEM ideas, especially with new projects. The possible range of new projects consists of updated versions of the current products and new products to address safety issues that are not yet covered.

A developer and the project manager jointly took part in the developer survey and combined their answers into a single response.

7.3 Communication Cognitive Effectiveness Measures of the SAMEM Artifacts

The high-level goals ***HL-GOAL-1*** and ***HL-GOAL-2*** (see Sub-chapters 1.1 and 2.4.1) of the thesis are respectively concerned with communication with the customers and the development team. There are several lower-level goals that are more specific and support the high-level thesis goals as shown with URN notation in Figure 2.8 and Figure 2.9. To achieve effective communication, the thesis emphasizes the use of graphical or visual communication techniques as much as possible, although not exclusively.

The primary measure of the effectiveness is that the participants in the project agree that the images used for communication are clear, complete with respect to their purpose, and accurate relative to the project concept [LS87]. Although many actual examples from the case studies have been presented, these remain descriptive and not prescriptive. The only mildly prescriptive aspect of the SAMEM is the use of the Solution Overview

Drawing (SOD) to communicate a larger picture. The question becomes: "What is the nature of the general guidelines needed to create an effective SOD?"

The communication cognitive effectiveness, as described in Sub-chapter 2.3.2, can be used to guide the formation of the SOD to maximize its usefulness. The cognitive effectiveness measures are described in the research from Moody, primarily [MvH08], but [Moo09] and [MHM10] show the application to UML and *i**.

7.3.1 Cognitive Effectiveness Principles and Metrics

Moody [Moo09] lists nine principles that are a prescriptive theory for designing a visual notation. With each principle, there are a number of more detailed metrics which can be applied to the notation. All the principles are listed below, but only the relevant metrics used for the SOD evaluation are included here:

1 Principle of Semiotic Clarity: There Should Be a 1:1 Correspondence between Semantic Constructs and Graphical Symbols.

2 Principle of Perceptual Discriminability: Different Symbols Should Be Clearly Distinguishable from Each Other.

3 Principle of Semantic Transparency: Use Visual Representations Whose Appearance Suggests Their Meaning.

   a) *Semantically Transparent Relationships* is the use of spatial arrangements of visual elements to predispose people toward a particular interpretation of the relationship among them even before the meaning of the elements is known.

4 Principle of Complexity Management: Include Explicit Mechanisms for Dealing with Complexity.

   a) *Modularization* is the common way of reducing complexity of large systems by dividing them into smaller parts or subsystems.

   b) *Hierarchy (Levels of Abstraction)* is one of the most effective ways of organizing complexity for human comprehension as it allows systems to be represented at different levels of detail, with complexity manageable at each level.

5 Principle of Cognitive Integration: Include Explicit Mechanisms to Support Integration of Information from Different Diagrams.

   a) *Conceptual Integration* is a summary (long shot) diagram, which provides a view of the system as a whole, which acts as an overall cognitive map.

6 Principle of Visual Expressiveness: Use the Full Range and Capacities of Visual Variables.

7 Principle of Dual Coding: Use Text to Complement Graphics.

   a) *Annotations* can improve understanding of diagrams in the same way that comments can improve understanding of programs.

8 Principle of Graphic Economy: The Number of Different Graphical Symbols Should Be Cognitively Manageable.

9 Principle of Cognitive Fit: Use Different Visual Dialects for Different Tasks and Audiences.

7.3.2 Cognitive Effectiveness Evaluation of the SOD

Some of the cognitive effectiveness principles and metrics are applicable to the SOD evaluation while some are not. The SOD is not a new visual notation; rather, it is a new mechanism to organize visual artifact entities for better communication. The principles and metrics that refer to a specific visual variable such as color or shape do not apply to the SOD, but may apply to the components assembled into the SOD. The references to principles and metrics given below use the principle number and metric letter from the list in Sub-chapter 7.3.1, for example, 3a represents the *Principle of Semantic Transparency* and the metric *Semantically Transparent Relationships*.

The SOD is an assembly of multiple models or visual artifacts that have a specific relationship to each other and together provide a more complete description, thereby easing the mental effort of remembering and integrating the models. A characteristic and lesson learned of the SODs from the case studies is a consistent selection of components and their layout, see Figure 3.5 where the components are Business Flow, Information Model, State Machine, and Notes. The choices meet the positive benefits expressed in:

- **3a** for using spatial relationships for a consistent SOD layout. The layout was developed via trial and error during the initial use of the SOD and then used for all the solutions.

- **4a** for modularization through the separation of the SOD into four areas of information. The actual model representations in the SOD were dependent on the area available and the agreed upon abstraction level of the model.

- **4b** for hierarchy or levels of abstraction by making the SOD be the top level or most abstract representation. The Information Model and Business Flow are the key components and the abstraction level in the SOD is based on communication effectiveness.

- **5a** for conceptual integration through the collection of the major aspects of the solution behavior, the solution data, and clarifying notes in one visual image. The single visual image is both memorable and communication effective by eliminating the need for a person to mentally integrate multiple models.

- **7a** for annotations with the intentional inclusion of a *Notes* section of the SOD for clarifying remarks. In practice, the Notes section evolved over the requirements elicitation work from initial lists of open questions to supporting references.

There are several aspects of cognitive effectiveness that are in strong alignment with the SOD approach. If the visual components that are assembled into the SOD are poorly done, then the SOD cannot overcome those deficiencies and might reinforce them. While the Case Study SOD examples consisted of four components, that does not mean that four components are appropriate for every domain. The SOD can consist of more or less than four, as long as the communication remains effective as agreed to by all participants.

7.4 The SAMEM Mapping to SEMAT and Essence

As described in Sub-chapter 2.1.7, the SEMAT is a standardization effort to bring some organization and foundation to software engineering. Software Engineering Method and Theory (SEMAT) is an initiative to improve the discipline of software engineering and make it more rigorous [JEJ12], [SEM98], [SEM16]. The SEMAT initiative is divided into four related areas of work: the Practice area, the Education area, the Theory area, and the Community area. The SEMAT work is supported by and has contributed to an OMG specification: Essence – Kernel and Language for Software Engineering Methods [OMG12b].

As an evaluation point, the SAMEM will be compared to the SEMAT and the Essence standard. The EssWork tool [Essb], [KM], [Essa] is a tool that is used in a project to manage the application of the SEMAT. The encoding of the SAMEM into the SEMAT framework will provide some confirmation that "GOAL-11: The methodology should support the incorporation of state-of-the-art software engineering results" is achieved.

7.4.1 The Essence Kernel Definition

This sub-chapter will provide a more detailed overview of the SEMAT as a basis for the comparison. Since many parts of this sub-chapter are definitional in nature, much of the text and Figure 7.7 through Figure 7.16 are taken directly from the SEMAT Kernel Quick Reference Guide [KM], [Essa] and sometimes slightly paraphrased. As stated in the quick reference guide, the Essence Kernel is organized into three discrete areas of concern, each focusing on a specific aspect of software engineering. These are:

- **Customer** – This area of concern contains everything to do with the actual use and exploitation of the software system to be produced.
- **Solution** – This area of concern contains everything to do with the specification and development of the software system.
- **Endeavor** – This area of concern contains everything to do with the team and the way that they approach their work.

Each area of concern contains a small number of:

- **Alphas** – representations of the essential elements of the software engineering endeavor that are relevant to an assessment of the progress and health of the

endeavor. The Alphas provide descriptions of the kind of things that a team will manage, produce, and use in the process of developing, maintaining, and supporting software.

- **Competencies** – representations of the key competencies required to do software engineering.
- **Activity Spaces** – representations of the essential things to do. The Activity Spaces identify and list generic challenges a team faces when developing, maintaining, and supporting software systems, and the kinds of things that the team will do to meet them.

7.4.1.1 The SEMAT Alpha Definition and Examples

Alpha is an acronym for an **A**bstract-**L**evel **P**rogress **H**ealth **A**ttribute. Figure 7.7 shows the seven Essence Kernel Alphas and their relationships. Each area of concern in the Essence Kernel is color-coded to assist in communicating the scope and differentiating the concern. The relationships between the Alphas show that the information and work in each Alpha depends on and impacts other Alphas, for example Stakeholders identify Opportunities. The Alphas are defined as:

- **Stakeholders**: The people, groups, or organizations who affect or are affected by a software system. The stakeholders provide the opportunity and are the source of the requirements and funding for the software system. The team members are also stakeholders.
- **Opportunity**: The set of circumstances that makes it appropriate to develop or change a software system. The opportunity articulates the reason for the creation of the new, or changed, software system. It represents the team's shared understanding of the stakeholders' needs, and helps shape the requirements for the new software system by providing justification for its development.
- **Requirements**: What the software system must do to address the opportunity and satisfy the stakeholders. It is important to discover what is needed from the software system, share this understanding among the stakeholders and the team members, and use it to drive the development and testing of the new system.
- **Software System**: A system made up of software, hardware, and data that provide its primary value by the execution of the software. The primary product of any software engineering endeavor, a software system can be part of a larger software, hardware, or business solution.
- **Work**: Activity involving mental or physical effort done in order to achieve a result. In the context of software engineering, work is everything that the team does to meet the goals of producing a software system matching the requirements, and addressing the opportunity, presented by the stakeholders. The work is guided by the practices that make up the team's way-of-working.

- **Team**: A group of people actively engaged in the development, maintenance, delivery, or support of a specific software system. One or more teams plan and perform the work needed to create, update, and/or change the software system.
- **Way-of-Working**: The tailored set of practices and tools used by a team to guide and support their work. The team evolves their way of working alongside their understanding of their mission and their working environment. As their work proceeds, they continually reflect on their way of working and adapt it as necessary to their current context.

From the Essence Kernel Quick Reference Guide:

> "The Alphas should not be viewed as a physical partitioning of your endeavor or as just abstract work products. Rather they represent critical indicators of the things that are most important to monitor and progress."

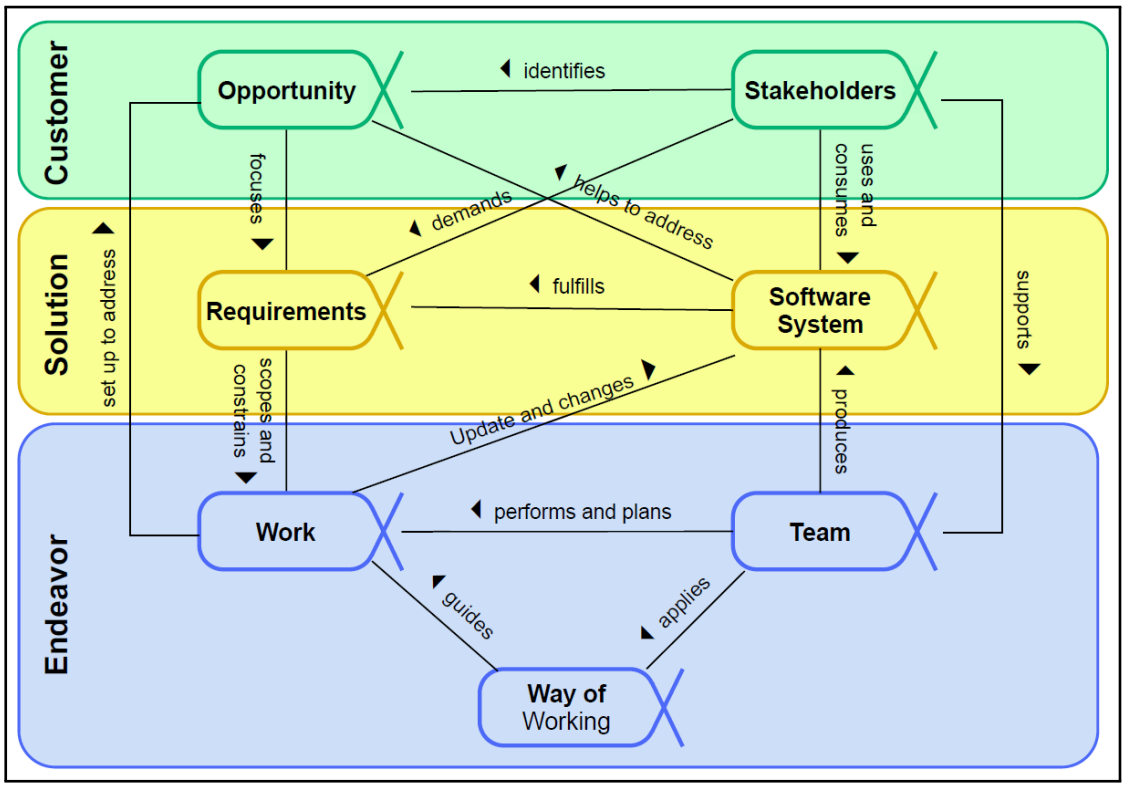


Figure 7.7: Essence Kernel Alphas.

Each Alpha is represented by an Alpha Overview Card, a set of Alpha State Cards, and an Alpha Full Checklist. An overview of the three Alpha artifacts is shown, in Figure 7.8 with details for the set of Stakeholder Alpha Cards. The set of Way of Working

Alpha State Cards is shown in Figure 7.9 and details for the Way of Working Alpha Full Checklist are shown in Figure 7.10.

The Alpha Overview Card provides a definition of the Alpha, a list of the states, and a summary of the important check points to fulfill in order to satisfy a state. The Alpha States make a simple and linear state machine that represents the progression from an unaddressed Alpha to a complete Alpha. For an Alpha to move to a more completed State, specific objectives listed in the Alpha Full Checklist must be achieved and agreed upon by the team. It is possible to regress to a previous State if project circumstances show that items from the Checklist are no longer fulfilled. The Alpha Full Checklist is a starting point that can be modified to adapt to different domains or project types (new product, functionality update, maintenance update, etc.).

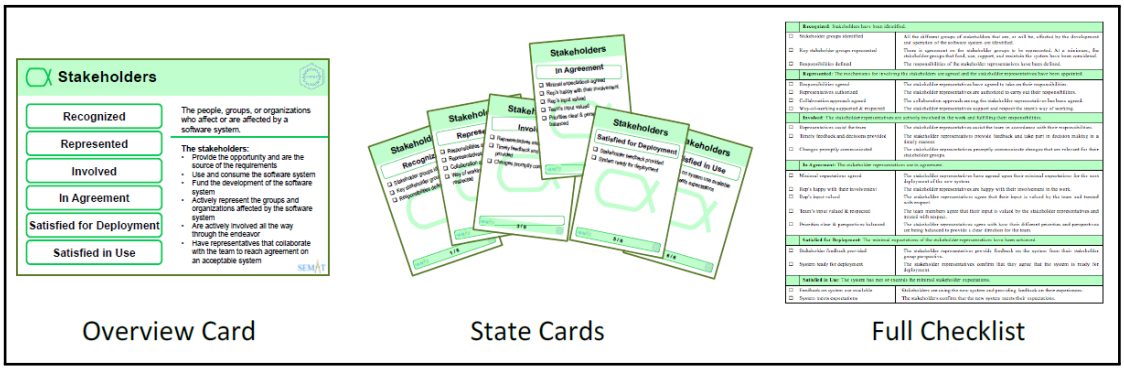


Figure 7.8: Alpha Artifact Overview.

The set of Alpha Cards, shown in Figure 7.9, consists of an Alpha Overview Card and an Alpha State Card for each state. On each Alpha State Card is a summary or label of the State Checklist Items that should be achieved in order for that state to be realized or achieved. The Alpha Full Checklist has the summary or label and an extended explanation of the Checklist Item, see Figure 7.10. The Alpha Checklist Items are the minimal suggested starting point from the Essence standard, can be modified or extended to be more domain or project relevant. The Essence standard is defined to support these adaptions.

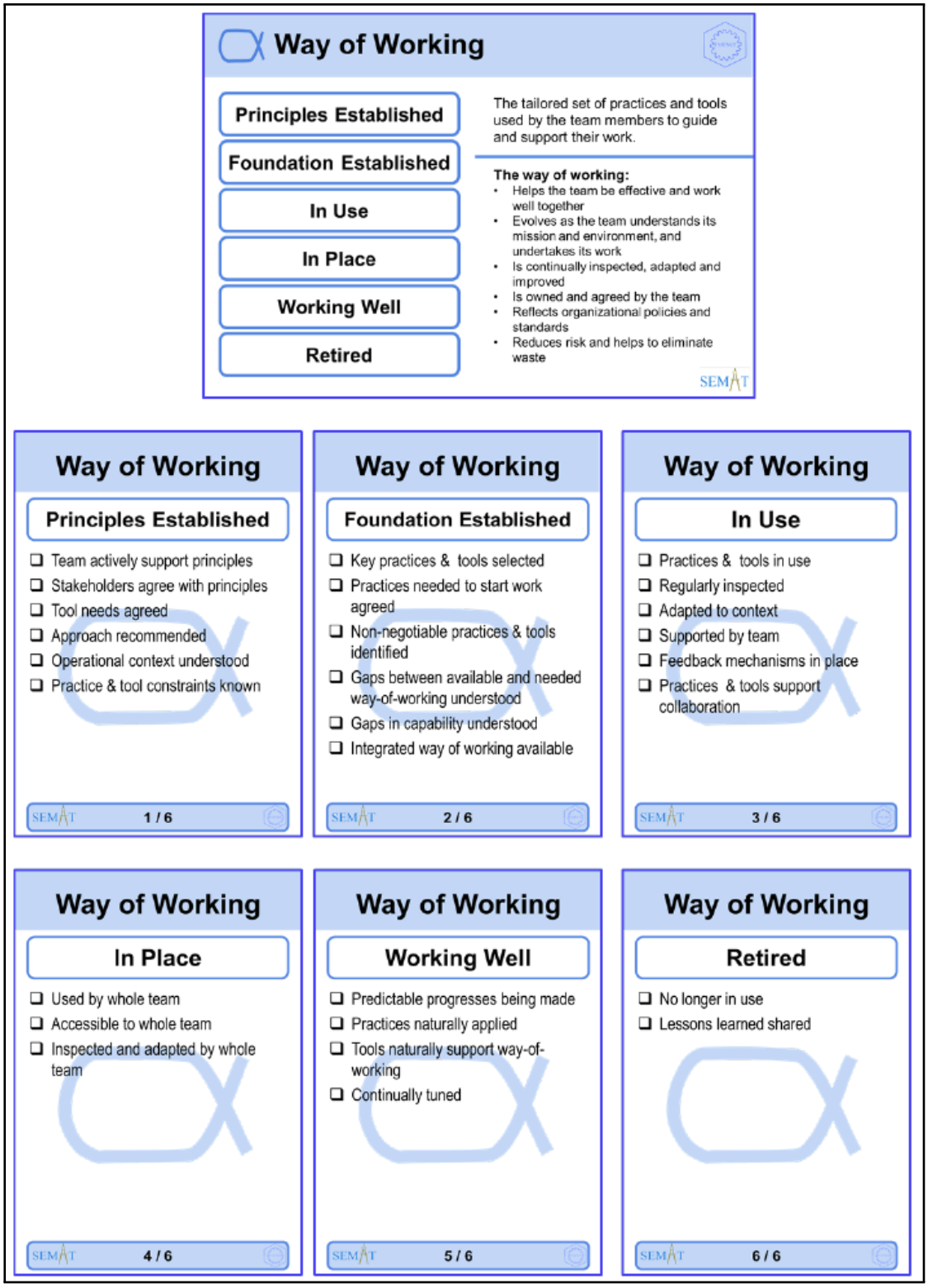


Figure 7.9: Way of Working Alpha Overview and State Cards Example.

	Principles Established: The principles, and constraints, that shape the way-of-working are established.	
☐	Team actively support principles	Principles and constraints are committed to by the team.
☐	Stakeholders agree with principles	Principles and constraints are agreed to by the stakeholders.
☐	Tool needs agreed	The tool needs of the work and its stakeholders are agreed.
☐	Approach recommended	A recommendation for the approach to be taken is available.
☐	Operational context understood	The context within which the team will operate is understood.
☐	Practice & tool constraints known	The constraints that apply to the selection, acquisition and use of practices and tools are known.
	Foundation Established: The key practices, and tools, that form the foundation of the way of working are selected and ready for use.	
☐	Key practices & tools selected	The key practices and tools that form the foundation of the way-of-working are selected.
☐	Practices needed to start work agreed	Enough practices for work to start are agreed to by the team.
☐	Non-negotiable practices & tools identified	All non-negotiable practices and tools have been identified.
☐	Gaps between available and needed way-of-working understood	The gaps that exist between the practices and tools that are needed and the practices and tools that are available have been analyzed and understood.
☐	Gaps in capability understood	The capability gaps that exist between what is needed to execute the desired way of working and the capability levels of the team have been analyzed and understood.
☐	Integrated way-of-working available	The selected practices and tools have been integrated to form a usable way-of-working.
	In Use: Some members of the team are using, and adapting, the way-of-working.	
☐	Practices & tools in use	The practices and tools are being used to do real work.
☐	Regularly inspected	The use of the practices and tools selected are regularly inspected.
☐	Adapted to context	The practices and tools are being adapted to the team's context.
☐	Supported by team	The use of the practices and tools is supported by the team.
☐	Feedback mechanisms in place	Procedures are in place to handle feedback on the team's way of working.
☐	Practices & tools support collaboration	The practices and tools support team communication and collaboration.
	In Place: All team members are using the way of working to accomplish their work.	
☐	Used by whole team	The practices and tools are being used by the whole team to perform their work.
☐	Accessible to whole team	All team members have access to the practices and tools required to do their work.
☐	Inspected and adapted by whole team	The whole team is involved in the inspection and adaptation of the way-of-working.
	Working well: The team's way of working is working well for the team.	
☐	Predictable progress being made	Team members are making progress as planned by using and adapting the way-of-working to suit their current context.
☐	Practices naturally applied	The team naturally applies the practices without thinking about them
☐	Tools naturally support way-of-working	The tools naturally support the way that the team works.
☐	Continually tuned	The team continually tunes their use of the practices and tools.
	Retired: The way of working is no longer in use by the team.	
☐	No longer in use	The team's way of working is no longer being used.
☐	Lessons learned shared	Lessons learned are shared for future use.

Figure 7.10: Way of Working Full Checklist Example.

7.4.1.2 The SEMAT Competencies Definition and Examples

The competencies area of the SEMAT provides an overview of the skills needed in each area. The SEMAT areas are Customer, Solution, and Endeavor. An overview is shown in Figure 7.11. In the Customer area, the skill competencies are those needed to demonstrate and communicate an understanding of the business needs and the domain. The skills needed in the Solution area are the abilities to analyze the requirements and opportunities, design and create the solution, and test the solution to verify it meets the requirements. In the endeavor area, the team has to be able to organize itself and manage its workload.

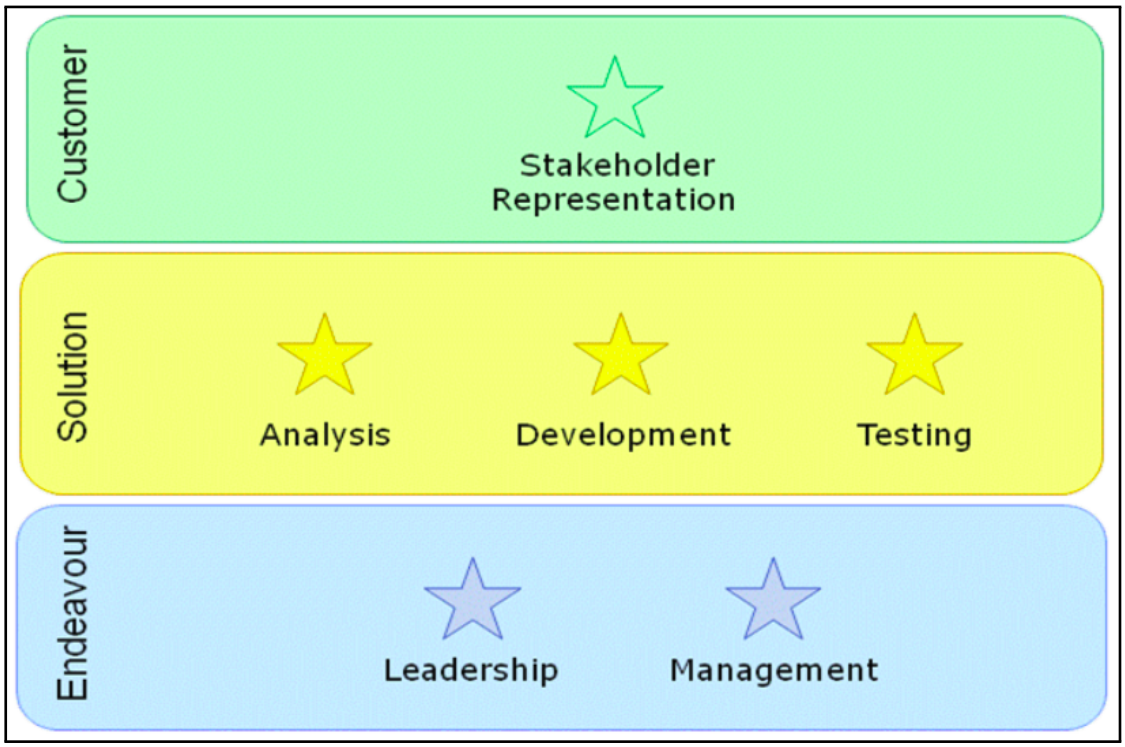


Figure 7.11: SEMAT Competencies Overview.

As defined in the standard, each competency has five levels of achievement. The higher competency levels build upon the lower ones. An individual at Level 2 has all the traits of an individual at Level 1 as well as the additional traits required at Level 2. Individuals at levels 1 and 2 have awareness or basic understanding of the knowledge, skills, and abilities associated with the competency. However, they do not possess the knowledge, skills, and abilities to perform the competency in difficult or complex situations and typically can only perform simple routine tasks without direction or other guidance. Individuals at Level 3 and above have mastered this aspect of their profession. The levels of competency are organized as a table and shown in Figure 7.12.

There are many factors that drive up the level of competency required on a project,

Level 1 Assists	Level 2 Applies	Level 3 Masters	Level 4 Adapts	Level 5 Innovates
❑ Has a basic understanding of the concepts ❑ Is able to act in a professional manner ❑ Is able to correctly respond to basic questions within his/her domain ❑ Is able to perform most basic functions within the domain ❑ Is able to follow instructions and complete basic tasks ❑ Is able to perform tasks under supervision	❑ Is able to collaborate within the team ❑ Is able to satisfy routine demands and simple work requirements ❑ Can handle simple challenges with confidence ❑ Is able to perform tasks under minimal supervision ❑ Can handle simple work requirements but needs guidance in handling any complications or difficulties ❑ Is able to reason about the context and draw sensible conclusions	❑ Is able to satisfy most demands and work requirements ❑ Is able to speak the domain language with ease and accuracy ❑ Is able to communicate and explain his/her work ❑ Is able to give and receive constructive feedback ❑ Knows the limits of his/her capability and when to call on more expert advice. ❑ Works at a professional level with little or no guidance.	❑ Is able to satisfy complex demands and work requirements ❑ Is able to communicate with others working outside the domain ❑ Can direct and help others working within the domain ❑ Is able to adapt his/her way of working to work well with others, both inside and outside their domain	❑ Has many years of experience and is currently up to date in what is happening within the domain ❑ Is recognized as an expert by peers ❑ Supports others in working on a complex professional level ❑ Knows when to innovate or do something different and when to follow normal procedure ❑ Develops innovative and effective solutions to the current challenges within the domain

Figure 7.12: SEMAT Generic Competency Levels Table.

including but not limited to:

- The size and complexity of the work.
- The size and distribution of the team.
- The size, complexity, and diversity of the stakeholder community.
- The novelty of the software system being produced.
- The technical complexity of the software system.
- The levels of risk facing the team.

Each of the areas within the competencies, as shown in Figure 7.11, has additional details. The generic competency levels listed in Figure 7.12 are refined to specific measures for each competency area. An example of the specific refinements for Way of Working is shown in Figure 7.13 and Figure 7.14.

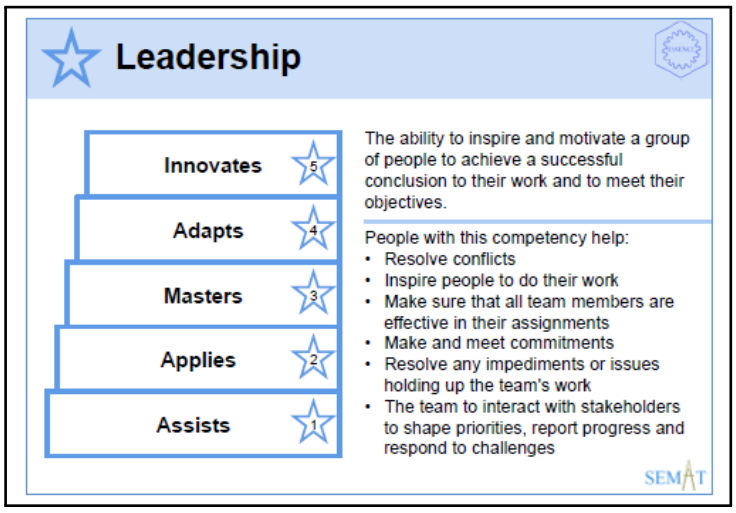


Figure 7.13: SEMAT Leadership Competency Card Example.

People with this competency help the team to:	Essential skills include:
• Inspire people to do their work • Make sure that all team members are effective in their assignments • Make and meet their commitments • Resolve any impediments or issues holding up the team's work • Interact with stakeholders to shape priorities, report progress and respond to challenges.	• Inspiration • Motivation • Negotiation • Communication • Decision making

Competency Levels

Level 1 – Assists	Demonstrates a basic understanding of the concepts and can follow instructions.
Level 2 – Applies	Able to apply the concepts in simple contexts by routinely applying the experience gained so far.
Level 3 – Masters	Able to apply the concepts in most contexts and has the experience to work without supervision.
Level 4 – Adapts	Able to apply judgment on when and how to apply the concepts to more complex contexts. Can help others in applying the concepts.
Level 5 – Innovates	A recognized expert able to extend the concepts to new contexts and inspire others.

Figure 7.14: SEMAT Leadership Competency Goals and Skills Example.

7.4.1.3 The SEMAT Activities Definition and Examples

The SEMAT kernel provides a set of Activity Spaces which complement the Alphas and provide an Activity-based view of software engineering. Within the Activity Spaces, work and specific tasks are created and accomplished. The activities that fill the spaces are derived from the Alpha Checklists and the tasks needed to move the Alphas through the States. Graphically, the Activity Spaces and Concerns are shown in Figure 7.15, while Figure 7.16 shows the relationship between Alphas, Activity Spaces, and Competencies.

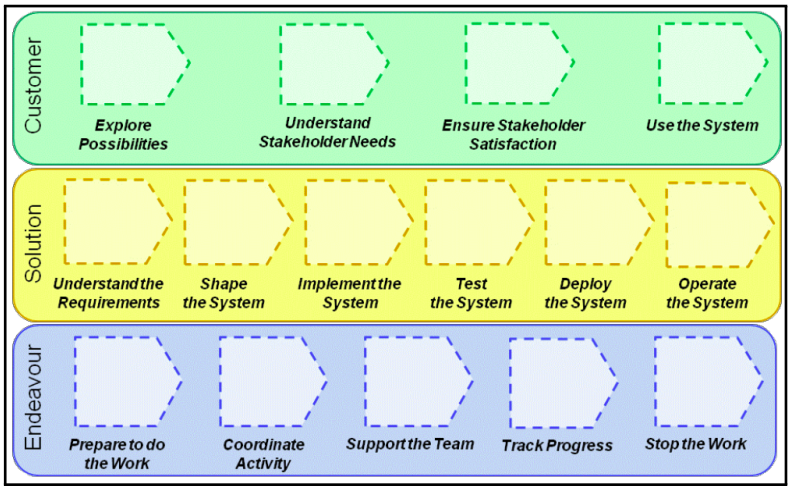


Figure 7.15: Activity Spaces and Concerns Overview.

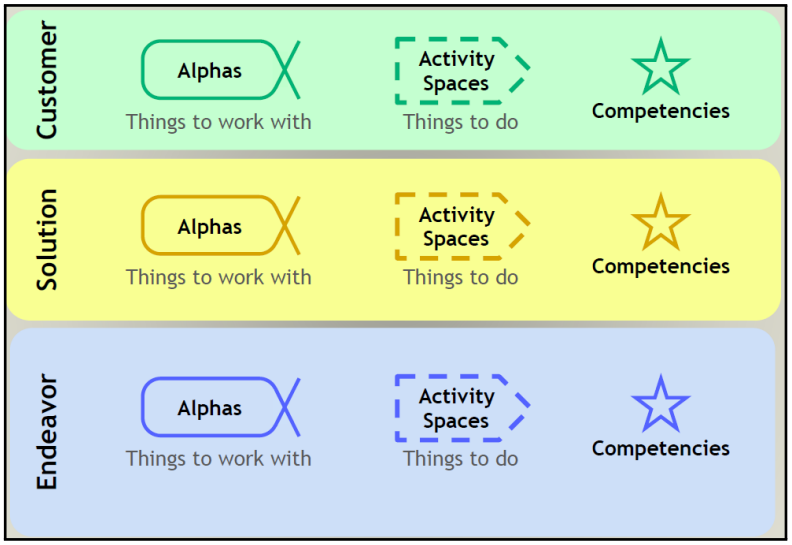


Figure 7.16: Alphas, Activity Spaces, and Competencies Relationship.

7.4.2 The SAMEM and SEMAT Comparisons

There are many areas where the SAMEM and the SEMAT have similar goals or have complementary approaches to the software development process. Although each brings a different viewpoint to dealing with software engineering projects, both emphasize ensuring project success. Both approaches provide guidelines but with flexibility designed into the methodology to enable adapting to multiple domains and project situations. The flexibility to adapt means that there are limits to detail and specifics defined in both the SAMEM and the SEMAT. The comparison will thus be focusing on the major concepts.

7.4.2.1 Framework Concepts Comparisons

The SEMAT framework is structured into three major concerns: Customer, Solution, and Endeavor, while the SAMEM is structured on the five RM-ODP viewpoints: Enterprise, Information, Computational (Behavior), Engineering, and Technology. The structuring has similarities but also different levels of abstraction. The structural comparison is organized around the RM-ODP viewpoints. In Figure 7.17, the relationships between the SAMEM and the SEMAT are shown at a high level. The correspondence is not one-to-one as the SAMEM has the two levels, the three Project Phases (see Sub-chapter 4.3.1) and the RM-ODP Viewpoints, while the SEMAT has three levels, Concern-Alpha-State. Table 7.12 shows the correspondence at a finer level of granularity.

The RM-ODP Enterprise Viewpoint, as used in the SAMEM, covers the same concerns as the Stakeholder and Opportunity Alphas in the SEMAT Customer concern. In both approaches, the identification of the goals of the solution, as stated in values to the stakeholders, are the work objectives. The values (SEMAT Opportunities) will be business or personal, such as processing an order faster, increasing work quality through automatic data checking, improving human safety by having cyber-physical entries in hazardous environments, or the pleasure of an entertaining game. The value of the solution is only relevant to a specific set of stakeholders. Someone that receives no value from the solution is by definition not a stakeholder. The explicit separation of Stakeholders and Opportunities of the SEMAT is an advantage that can be used by the SAMEM in the Enterprise Viewpoint to be more explicit. Both approaches allow flexibility as to the expression of the Opportunities or goals and the Stakeholders; however, clarity of communication is emphasized.

The SAMEM uses the Information and Computational (Behavior) Viewpoints for the representation of the SEMAT Requirements Alpha in the Solution Concern. The SAMEM approach has more granularities in the representing of the requirements than the SEMAT. The splitting of the requirements across the two viewpoints makes interaction more explicit and enables elicitation through two opposing but complementary perspectives. Although both approaches support flexibility in the expression of the requirements, the SAMEM emphasizes the use of models to maximize effective communication.

The RM-ODP Engineering Viewpoint corresponds in part to the Software System

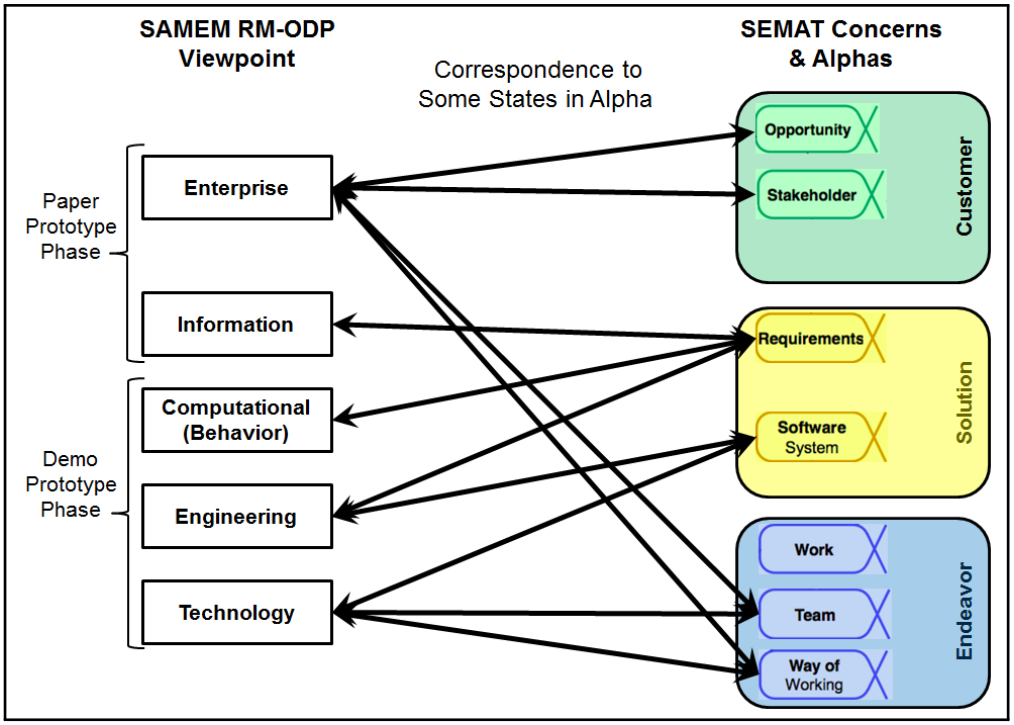


Figure 7.17: High Level SAMEM - SEMAT Concept Correspondence.

Alpha of the Solution Concern. Within the SAMEM, the Engineering Viewpoint includes the various design alternative artifacts and the evaluations of the designs in fulfilling the requirements. There is a correspondence with the *Architecture Selected* State for the Software System Alpha, as shown in the Checklist items for that State. The SAMEM is more explicit in separating the architectural design activities than the SEMAT.

The SEMAT Software Solution Alpha also includes Checklist items that are done in the RM-ODP Technology Viewpoint, such as selecting a hardware platform or programming languages. The examples of hardware platform and programming language selection are checkpoints for achieving the *Architecture Selected* State for the Software System Alpha (see Figure 7.18 [Essa]). The RM-ODP Engineering and Technology Viewpoints deliberately separate these design decisions versus the SEMAT. The separation of engineering and technology decisions in the SAMEM allows for finer grained communication of project progress than the SEMAT states.

The SAMEM does not have the States and Checklists of the SEMAT Alphas; however, it does have something similar to measure project process. The project process phases, as outlined in Figure 4.1, perform a similar function to the SEMAT states. The relationships between the SAMEM project phases and the SEMAT Alphas and States are shown in Table 7.12. The SEMAT Alpha that is not addressed in Table 7.12 is Work, since that happens everywhere and is dependent on the specific project management approach. The SAMEM Project Phase of **Initiation** is not covered in Figure 4.1, as the business

	Architecture Selected: An architecture has been selected that addresses the key technical risks and any applicable organizational constraints.	
☐	Architecture selection criteria agreed	The criteria to be used when selecting the architecture have been agreed on.
☐	HW platforms identified	Hardware platforms have been identified.
☐	Technologies selected	Programming languages and technologies to be used have been selected.
☐	System boundary known	System boundary is known.
☐	Decisions on system organization made	Significant decisions about the organization of the system have been made.
☐	Buy, build, reuse decisions made	Buy, build, and reuse decisions have been made.
☐	Key technical risks agreed to	Key technical risks agreed to.

Figure 7.18: SEMAT Software System Architecture Selected State Checklist.

decision to initiate a project is outside the scope of the SAMEM.

Table 7.12: SAMEM Project Phases and SEMAT Alpha States Mapping.

SAMEM Project Phase	**SEMAT Alpha and State**	
	Alpha	**State**
Initiation: Not a part of SAMEM as this work in the consulting company is done as a separate strategy project.	Opportunity	Identified Solution Needed
	Team	Seeded Formed
	Way of Working	Principles Established Foundation Established
Phase 1 – Paper Prototype Covers Viewpoints: Enterprise Information Computational (Behavior)	Stakeholders	Recognized Represented Involved In Agreement
	Opportunity	Value Established Viable Solution Needed
	Requirements	Conceived Bounded Coherent Acceptable
	Team	Collaborating Performing
	Way of Working	In Use In Place Working Well
Phase 2 – Demo Prototype Covers Viewpoints: Engineering Technology	Stakeholders	In Agreement
	Requirements	Addressed
	Software System	Architecture Selected Demonstrable Useable Ready
	Team	Seeded Formed Collaborating Performing
	Way of Working	In Use In Place Working Well

SAMEM Project Phase	SEMAT Alpha and State	
	Alpha	**State**
Phase 3 – CRP Verification of assembled demo prototypes	Stakeholders	Satisfied for Deployment Satisfied in Use
	Opportunity	Addressed
	Requirements	Addressed Fulfilled
	Software System	Operational
	Team	Seeded Formed Collaborating Performing
	Way of Working	In Use In Place Working Well

Within each SAMEM Project Phase, there are multiple iterations. One or more iterations can be involved in incrementing an Alpha State to the next level of completeness. The SEMAT has Alpha States that correspond to the deployment of the solution, whereas the SAMEM does not. The reason the SAMEM does not specify the deployment as part of the project is that the deployment timing is often a business decision independent of the solution creation. For example, if a solution depends on changes to existing infrastructure such as a database, then the solution deployment might be scheduled together with other database updates to minimize business disruption.

7.4.2.2 The SAMEM – SEMAT Comparison Summary

The SAMEM and the SEMAT do not conflict in any fundamental manner. Each approach has a similar goal of ensuring project success. The variations are in the details of what is needed to ensure success. While the SEMAT stays independent of the project process, the SAMEM is based on the agile ideas of iterative & incremental progress which relate to the *Way of Working* Alpha. The SEMAT approach can benefit from applying the abstraction refinement of the RM-ODP to the *Software System* Alpha. Specifically, the SEMAT *Software System* Alpha's initial State of *Architecture Selected* covers the work distributed across the RM-ODP Engineering and Technology Viewpoints, which is a large granularity mismatch (see Figure 7.18).

The SEMAT ideas of Project Alpha Metrics with States indicating completeness can be applied to the SAMEM. Specifically, applying the SEMAT State and Checklist to the SAMEM Project Process Example Activity Model in Sub-chapter 5.3 and Figure 5.8, will enhance the robustness of several tasks. The completing of the *Goal Elicitation* task can be indicated by a SEMAT *Opportunities* State of *Addressed*. The decision task of *Completeness Evaluation* maps to the SEMAT idea of monitoring the process progress

and is indicated by multiple Alphas in their final or near final State.

There is recent research extending the SEMAT and its associated OMG standard of ESSENCE [OMG12b], while the SAMEM is primarily the work of a single person. The research work described in [HSG16] of modeling the dynamic semantics of ESSENCE as a graph grammar brings a rigor that is currently lacking in the SAMEM. The graph grammar and graph transformations form the basis for tools for ESSENCE which can be applied to the SAMEM when it is defined in the SEMAT. Through the ESSENCE definitions of Alphas, States, and Checklists, visualization tools for monitoring a process can be created as described in [BSBG17]. On the other hand, the SEMAT and ESSENCE can be extended with concepts from the SAMEM, such as incorporating the SEFPs into the Checklists and suggesting sub-Alphas to support the abstraction transition guidance offered by the RM-ODP.

7.5 Consulting Business Success Measures

The SAMEM is intended to be a pragmatic methodology that benefits the people and organization using it. The early version of the SAMEM (pre-SAMEM or Paper Prototype) has been very successful over the course of the two case studies. Some aspects of the success are validated by the responses detailed in Sub-chapter 7.1.4. The business success is measured along several dimensions: team size, speed to requirements closure, quality of requirements, customer satisfaction, and business benefit of the solution. During the first four years of use of the pre-SAMEM, a parallel project was running that used the more traditional project approach of waterfall and text-based requirements. The solution intent of each project was roughly the same; however, the pre-SAMEM–based project accomplished more with fewer people. Both projects conducted After Action Reviews at the end, both internally and with the customer.

7.5.1 Team Size

The development team size of the pre-SAMEM project was smaller by a factor of four. The smaller team size reduced the team communication overhead as documented by [Bro95]. The smaller team could afford to send the same people, which covered all the significant project roles, to meet with the customer, while the larger team needed to rotate people for visits. Rotating people produced misunderstandings in the development team, as each person had a partial view of the customer needs. Also, the rotating team members created a Mythical Man-Month [Bro95] situation with higher inter-team communication overhead. For the pre-SAMEM project, the major impact was more consistent communication between the team and customer, which resulted in a better shared vision of the solution needed (conceptual integrity). This is also supported by the responses to CS-Q1 and DS-Q1

7.5.2 Speed to Requirements

Speed to requirements closure was much faster for the pre-SAMEM team versus the traditional team. The pre-SAMEM team would have a fairly stable set of business requirements in about six weeks and sometimes as little as two weeks, while the traditional team took close to eight months. There were a couple of major factors that influenced the time to closure.

It was much faster to produce the graphical models; the initial model was often done on a whiteboard with the customer in a few hours. Once a good relationship was established and the initial model was developed at a face-to-face meeting, many reviews happened via web-based meetings or through emailing model artifacts for review.

Another speed factor was the better communication within the modeling team and with the customer. There was significantly less rework. The models reduced the ambiguity that is present in text-based requirements. The traditional text-based approach had many more rework cycles because requirements were poorly expressed, often as simple declarative sentences, which demanded that the reviewer mentally combines several requirement statements to achieve a complete and hopefully correct understanding [LS87].

The third speed factor was that models allowed the project to be more agile. The models enabled easier agreement with the customer on partitioning the solution into smaller partial solutions when faced with unforeseen schedule or budget constraints. This was often achieved by going to the graphic of the information model or the behavior model and drawing a circle around the partial solution. This kind of partitioning is very difficult to do with many text-based requirements because of the linear structure.

The survey questions CS-Q16 and DS-Q13 directly address this with positive responses from all participants.

7.5.3 No Requirements Prioritization

One of the goals of modeling requirements rather than using a text-based approach was the elimination of the wasted time spent prioritizing requirements, then arguing over the priorities. The author has seen this in multiple past projects and the basic cause was the undisciplined use of brainstorming to generate possible requirements. The projects would use the technique of prioritization to filter the requirement candidates rather than delete obvious bad requirements.

Not a single minute in the case study projects was spent on traditional requirements prioritization. The model-driven approach with the graphical communication allowed clear expectation setting with the customer and within the development team.

There were times when the scope of a project was changed to respond to external changes in budget or competitive business pressures. This was always done within the scope of the business flow and information model while maintaining the conceptual integrity.

There are multiple survey questions that cover this through the small size of the requirements and the associated clarity, such as CS-Q7, DS-Q5, CS-Q10, DS-Q8, CS-

Q18, and DS-Q14.

7.5.4 Quality of Requirements

The quality of requirements produced by modeling was significantly better as indicated by the positive responses to survey questions CS-Q17, CS-Q18, and DS-Q14. In a business situation, an important quality metric is the stability of the requirements specification during implementation and verification. The specifications that used models were about one-third of the length of text-based specifications. This resulted in more thorough reviews and the possibility of a wider range of people to be involved in the reviews. While either approach can produce a high quality requirements specification, the modeling approach took about 10% of the time in final specification review and rework compared to the text-based approach.

The stability metric used was the number of issues identified during the Conference Room Pilot (CRP) verification exercise that were traceable back to a requirement. For the model-based requirements, the number of issues per project was in the 15 – 20 range. About half of those issues needed to be resolved by process improvements in the business or resetting the requirements specification because of insight gained during the CRP [Bro10]. The other issues were resolved by implementation changes that averaged about one engineering-week of effort. In the traditional project, the CRP issue list was over 200 and several engineering-months of reimplementation and requirements rewriting were needed.

7.5.5 Customer Satisfaction

Customer satisfaction with the solutions was about the same. The primary reason for equal success is the dedication to quality from all participants; however, the pre-SAMEM project was more efficient. Two aspects of the model-based approach with an iterative & incremental style garnered special praise. The model-based project had more visibility into the progress for the solution sponsors; see responses to survey questions CS-Q1, DS-Q1, CS-Q2, DS-Q2, CS-Q24, DS-Q26, CS-Q25, and DS-Q27. The graphical nature of the requirements was part of the visibility, but the feedback from the participants on how fast the project was moving, compared to expectations based on other projects, was also an important factor.

The graphical nature of the models allowed the creation of the SOD, Figure 3.5. The SOD is a software engineering solution drawing. It fit easily onto an 11x17 inch page and contained the essential models. It formed the basis of the start of all the meetings and customers could show it to colleagues and easily explain what the solution was going to be. One was posted on the wall of the R&D vice president's office, which was most unusual. The survey questions CS-Q8 and CS-Q6 specifically inquire about the effectiveness of the SOD or 1-pager.

7.5.6 Consulting Company Business Benefits

The primary business benefits of modeling were in the earlier delivery of a part of the solution, which established the ability to deliver value to the customer. Starting with the information models allowed the easy partitioning of the solution. In looking at a picture of the solution information components, lines could be drawn around sets of information units, which became the realization partitioning. The partitioning made it easy to explain to the customer why a certain realization order was best, as some of the partitions were foundational. Survey questions that cover this issue are CS-Q1, DS-Q1, CS-Q9, DS-Q7, CS-Q22, DS-Q23, CS-Q23, and DS-Q24.

Although often difficult to quantify, organization change in response to a business change is a critical factor of success. An unexpected benefit of the partitioning was the ability to evaluate whether the groups involved with the partition were more or less open to change; see survey question CS-Q4. When the flexibility existed, targeting groups that were more open to improvement in their operating procedures minimized organization resistance and created examples of success.

The smaller team was a financial benefit to the customer, as it kept the costs down. This helped with the satisfaction, as the customer felt they got a good return on the investment; covered by survey question CS-Q3.

7.6 Threats to Validity

The majority of the evidence supporting the effectiveness of the SAMEM comes from two case studies. As described in [WRH+00], there are four primary threats to the validity of the case study evidence. These threats are examined in detail below.

7.6.1 Construct Validity

Construct validity is the extent to the degree the operational measures that are studied represent the subject of this thesis. This ties back to the goals of the thesis as expressed in Sub-chapter 1.1. The survey mechanism is used to gather the experiences and evaluation from the people that used the SAMEM. If the respondents to the survey interpreted the questions in a manner different from the author's intent, then there can be construct validity concerns.

There are several mechanisms to mitigate misinterpretation of the survey questions. The terminology used in the questions is carefully crafted to be the same as used during the case study projects. Most of the questions clearly have a single objective targeted toward validation of a single thesis goal; however, questions such as CS-Q1, DS-Q1, CS-Q24, DS-Q26, CS-Q25, and DS-Q27 ask for a broader opinion across the whole SAMEM approach. The answer possibilities use a consistent ordinal range from positive to negative with a neutral middle value. There are several places in the survey for a respondent to extend the answer with comments.

It is felt that there was no misinterpretation of the survey questions based on the

experience of the respondents with the pre-SAMEM process (see survey questions CS-Q26 and DS-Q28).

7.6.2 Internal Validity

Internal validity is of concern when casual relationships are examined. The case study surveys investigate whether the pre-SAMEM project process was effective and more effective than other approaches used by the respondents. The effectiveness is based on the respondents' direct experience in using the pre-SAMEM. There is not a comparison of the pre-SAMEM to any other methodology such as RUP.

One confounding factor that might have an impact on the internal validity is that the people involved in the case studies are all exceptionally adept at modeling and agile software engineering methodologies. However, that is unlikely as each case study involved a different set of customers, who are non-software people, and different development team members. The only common person is the author.

7.6.3 External Validity

External validity is concerned with the generalizability of the findings. The case study companies manufacture physical medical device products. Their industry domain can be characterized as manufacturing, product development, and highly regulated by government authorities.

In generalizing, another domain can be created by weakening or removing the regulatory oversight. The weakening would impact the SAMEM by removing the need to produce and manage some of the project artifacts. The removal of project process work does not impact the SAMEM negatively.

An example of generalizing the SAMEM to work in other domains is shown in Subchapter 7.2.1. The domain characteristics for the cellular application development are software applications, mobile technologies, and critical user interface sensitivity. The application of the SAMEM in this domain is through the organization of the project process and the initial emphasis on determining the mobile application goals. Some of the requirement artifacts are new in comparison to the case study examples. For example, the criticality of the mobile application user interface demands sketches of the screen layouts and a function/screen navigation map.

The design of the SAMEM enables generalization through emphasizing best practices but not requiring specific modeling or visual technologies. The visual or modeling best practices are epitomized by the pioneering work of Larkin and Simon [LS87] and the principles and metrics cognitive psychology as described by Moody [Moo09].

7.6.4 Reliability

The reliability of a case study is dependent on the specific researchers. The author of this thesis is the developer of the SAMEM, guided the case study projects which used the SAMEM in the role of Solution Architect, and developed the survey questionnaire

for collecting the empirical evidence. As the developer of the SAMEM, the author has a vested interest in proving the benefits of his work. Steps were taken to ensure the neutrality of the author in collecting the supporting evidence for the value of the SAMEM. In the case of the EBC company, as described in Sub-chapter 7.2.2, the early version of the SAMEM was introduced by a member of the development team from CS-1.

The questionnaire was sent to a well-known professor experienced in empirical software engineering studies, Prof. Dr. Michel Chaudron, for independent review of the question formulations and answer possibilities. Some adjustments were made to question vocabulary and ordinal answer ranges based on his recommendations.

Several years have passed since the pre-SAMEM or Paper Prototype process was in use in the case study companies. While the relationships are still positive, there is no direct contact with or influence on the respondents through daily work contacts. The respondents were assured that all answers would be anonymous to encourage honesty and the questionnaire does not have a place for the respondent to enter identifying information. Due to the small response size, the author can make a reasonable guess as to the respondent based on their role within the project, but this information is not used to skew any results.

It is hoped that the time gap provided perspective on the process. The small sample size makes statistical analysis impossible. The lack of a statistical analysis does jeopardize the ability to generalize the results. However, in order to get an honest evaluation of the SAMEM, it must be used in real-world industrial projects. The customers need to be true domain experts that understand the problem complexities that must be modeled and the business context of the solution.

7.7 The SAMEM Evaluation Summary

The SAMEM is evaluated on five major dimensions:

1 empirical surveys of the case study users in multiple real-world projects and their evaluation of the fulfillment of the SAMEM goals,

2 a discussion of the use of the SAMEM ideas on other projects with some initial indicators of applicability,

3 the cognitive effectiveness of the modeling ideas in communicating requirements and the unique SAMEM SOD visual artifact,

4 an analytical comparison with the Software Engineering Method and Theory (SEMAT) [SEM98] standard as to project scope, differences, and similarities,

5 an analytical evaluation against the consulting business success measures of smaller team, faster project, and improved customer relationship.

The responses to the survey from SAMEM practitioners are overwhelmingly positive. All the thesis high-level goals are more than fulfilled, but there are opportunities for

improvement, which will be discussed in Chapter 8. Confirmation of the modeling of the requirements to improve clarity, compactness, communication, and fitness for further design refinement is validated through the positive answers to the survey questions. Respondent quotes such as "I can't imagine proceeding without it" provide additional assurance that the SAMEM is effective.

The SAMEM is efficient in guiding the project progress forward. The projects made reasonable progress toward the solution as confirmed by the respondents. The business management aspects of the SAMEM are as important as the development of the needed requirements specification and design solutions.

The major area for improvement, as shown by the surveys, is in better explanation of some of the SAMEM concepts. The rationale for the use of the RM-ODP to guide the project process through the design abstraction levels needs improvement. Examples of the different abstraction levels and how to proceed from one to another would help establish the understanding. The use of state machines in the information model needs better explanation. People with an engineering or physical science background understand state machines, but case study stakeholders with a biological science, medical, or business background had more difficulty.

In a manner similar to the case studies, the three other applications of the SAMEM show indications of success. There is improved communication and understanding within the team. The purpose and goals of the solution are more clearly stated, thereby focusing the team. The three smaller projects are still under development and have not employed the full SAMEM approach, so there is uncertainty on the full benefits.

The cognitive effectiveness of the SOD is positive, but there is room for improvement. Specifically, some of the components of the SOD as used in the case study companies were UML-based. It has been shown that the UML graphical artifacts have poor cognitive effectiveness [MvH08]. While the SEMAT does not specify any modeling artifacts for communication, the Checklists for the Requirements and Software System Alphas could be extended with cognitive effectiveness metrics and also applied in the SAMEM.

While basically compatible, the differences in the SAMEM and the SEMAT originate in the initial goals. The SAMEM focuses on the main solution development process. In contrast, the SEMAT considers the larger project process from initiation to deployment. Each methodology can benefit from ideas in the other. The SAMEM can benefit from including the ideas of States on the artifacts with accompanying Checklists for completeness. Currently, the SAMEM relies on team consensus that an artifact is complete. The SEMAT can benefit from the greater detail defined in the SAMEM relative to the Software Solution Alpha. Specifically, the SEMAT improvements can incorporate the RM-ODP abstraction hierarchy to separate the bundled Checklist items of the Architecture Selected State in the Software System Alpha.

A software development methodology that does not consider the benefits to a business has serious deficiencies. Most software is developed with budget and time constraints. The SAMEM was developed with business constraints in mind and many of the methodology goals relate to working effectively within those constraints. Especially within the case study projects, the SAMEM has demonstrated the needed ability to deliver success

relative to the business constraints, as shown by survey questions CS-Q1, CS-Q2, CS-Q3, and CS-Q4.

Chapter 8

Chapter 8 Conclusions and Future Work

This chapter presents a summary of the contributions of the SAMEM relative to the research questions stated in Sub-chapter 1.3. Additional contributions of the SAMEM will be presented. The final sub-chapter will discuss future work that will extend the SAMEM and strengthen it.

8.1 Contributions

The main contributions of the SAMEM are expressed through addressing of the research questions. Each research question concerns an improvement in the manner in which a software-based solution is created. The two general dimensions of improvement are in more effectively creating a solution by using fewer people over a shorter time period and reducing risks to creating a poor solution. Beyond the research questions, there are other contributions that the SAMEM provides to software solution creation.

8.1.1 Research Questions Addressed

The research questions from Sub-chapter 1.3 are restated below, each with a summary of how well the question was addressed. Table 7.1 and Table 7.2 in Sub-chapters 7.1.2 and 7.1.3 are the survey sources for all the data referenced in the research question sub-chapters below.

8.1.1.1 RQ1

Research Question 1 (RQ1): Is the visual modeling of the majority of requirements possible?

> The answer is **yes**.

The contribution is that modeling of requirements versus text-based has a positive impact. There are several pieces of evidence to support this answer. The following survey questions relative to requirements modeling all have positive responses: CS-Q5 (7 of 8 Very Effective), CS-Q7 (8 of 8 Very Effective), CS-Q10 (7 of 8 Yes Better, 1 of 8 Mostly Better), CS-Q17 (4 of 4 Yes Better), CS-Q18 (8 of 8 Yes Better), and DS-Q16 (4 of 4 Yes Better).

The invention of the Solution Overview Drawing (SOD), which is a summary or abstraction of the requirements, is a larger contribution of this work. The SOD had a positive impact on the case study projects as supported by the answers to the survey questions: CS-Q8 (6 of 8 Very Effective, 2 of 8 Somewhat Effective) and CS-Q15 (5 of 8 Very Useful, 3 of 8 Useful with Limitations). The limitations of the SOD arose in the need to explain the complexity of the drawing to first time viewers.

The existence of many visual examples of requirements from the two case studies is also proof of satisfying RQ1. The examples of the requirements models can be seen in the following Sub-chapters: 3.5.1.1, 3.5.2.1, and 3.5.3.1.

8.1.1.2 RQ2

Research Question 2 (RQ2): Is an iterative & incremental (agile) process approach effective in the elicitation of requirements?

The answer is **yes**.

The objective of this research question is to examine whether the agile techniques that have been successful in code development can be extended to requirements elicitation. The survey questions that support this conclusion are: CS-Q2 (7 of 8 Very Positive, 1 of 8 Moderately Positive) and CS-Q21 (7 of 8 Yes Better, 1 of 8 Mostly Better).

The contribution is that an iterative & incremental (agile) approach can be extended to other areas in the solution process beyond code development.

8.1.1.3 RQ3

Research Question 3 (RQ3): Does there appear to be a reduction in the accidental complexity of team communication with visual models for the requirements and design?

The answer is **yes**.

This research question challenges, in part, the premise of the book *The Mythical Man-Month* by Brooks [Bro95], which is that adding people to a project does not necessarily speed up progress because of the additional communication overhead. While the basic premise of the Mythical Man-Month is considered to be true, the question arises as to whether the communication overhead can be reduced via better mechanisms than text. The SAMEM emphasizes the use of graphical artifacts and models to improve communication by reducing miscommunication while having a denser medium in comparison to text-based requirements.

The main supporting evidence is from the following survey questions: CS-Q5 (7 of 8 Yes Better, 1 of 8 Mostly Better), CS-Q7 (8 of 8 Very Effective), CS-Q15 (5 of 8 Yes Better, 3 of 8 Mostly Better), CS-Q16 (5 of 8 Yes Better, 3 of 8 Mostly Better), and CS-Q17 (4 of 4 Yes Better).

8.1.1.4 RQ4

Research Question 4 (RQ4): Does visual modeling have a positive effect on the size (smaller) and clarity (easier to comprehend) of the requirements specification?

The answer is **yes**.

The assumption behind RQ4 is that if the requirements specification can be smaller in size through the use of visual models, then it will be faster to write, faster to review, and a smaller team is possible. A smaller team minimizes the Mythical Man-Month impacts and lowers project costs. Creating and reviewing the requirements faster results in lower project costs which improve the cost-benefit calculation.

The support is seen in the following survey questions: CS-Q7 (8 of 8 Very Effective), CS-Q8 (6 of 8 Very Effective, 2 of 8 Somewhat Effective), CS-Q10 (7 of 8 Yes Better, 1 of 8 Mostly Better), CS-Q11 (7 of 8 Yes Better, 1 of 8 Mostly Better), CS-Q15 (5 of 8 Yes Better, 3 of 8 Mostly Better), CS-Q17 (4 of 4 Yes Better), CS-Q18 (8 of 8 Yes Better), CS-Q19 (7 of 8 Yes Better, 1 of 8 Mostly Better), DS-Q16 (4 of 4 Yes Better), and DS-Q19 (4 of 4 Yes Better).

8.1.1.5 RQ5

Research Question 5 (RQ5): Does an iterative & incremental full project process contribute in a positive manner to a faster project process with fewer people?

The answer is **yes**.

The survey questions supporting this conclusion are: CS-Q1 (7 of 8 Very Effective, 1 of 8 Moderately Effective), CS-Q2 (7 of 8 Very Positive, 1 of 8 Moderately Positive), CS-Q3 (4 of 4 Yes), CS-Q9 (4 of 8 Very Effective, 4 of 8 Somewhat Effective), CS-Q21 (7 of 8 Yes Better, 1 of 8 Mostly Better), CS-Q22 (8 of 8 Yes Better), and CS-Q23 (4 of 8 Yes Better, 4 of 8 Mostly Better).

8.1.1.6 RQ6

Research Question 6 (RQ6): Do iterative & incremental (agile) approaches in the front-end of a project process more effectively feed into agile development processes?

The answer is **yes**.

The assumption behind this research question is that it is better to have a consistent project management style across the entire project. The iterative & incremental code creation project management style has been used successfully for many years, but there is little evidence of its use throughout the entire project lifetime.

The supporting survey questions are: CS-Q2 (7 of 8 Very Positive, 1 of 8 Moderately Positive), CS-Q22 (8 of 8 Yes Better), and DS-Q25 (4 of 4 Yes Better).

8.1.2 Other Contributions

The other significant contributions of this thesis consist of the invention of the Solution Overview Drawing (SOD), the application of the RM-ODP abstraction framework to guide the project process, and the establishment of a set of Software Engineering First Principles (SEFP) as a solution check and to stimulate innovation.

8.1.2.1 Solution Overview Drawing Invention

The SOD is a visual model that communicates the overall intent and scope of the solution at a high level of abstraction. The purpose of the SOD is to provide a common image of the solution to establish agreement and initiate project work partitioning. As the SOD is a summary of the solution, it evolves in an iterative & incremental manner as design decisions are made over the course of the project. Survey questions that support the benefit of the SOD are: CS-Q8 (6 of 8 Very Effective, 2 of 8 Somewhat Effective), CS-Q9 (4 of 8 Very Effective, 4 of 8 Somewhat Effective), and CS-Q15 (5 of 8 Very Useful, 3 of 8 Useful with Limitations).

8.1.2.2 Modeling Innovations

Several requirements modeling innovations are a contribution of the SAMEM, such as the ATM and the Business Flow to Use Case mapping.

The Action Transformation Matrix (ATM) (see Sub-chapter 4.3.5.3) is a compact model of the CRUD behavior pattern. The ATM acts as both a behavior checklist and a two-dimensional index to specific requirements. The ATM puts each CRUD behavior in the context of its business characteristics, such as *when* it can be invoked, *who* can invoke it, *pre-conditions* to the invocation, *post-conditions* to the invocation, and necessary data.

The graphical connection of the Business Flow abstraction of behavior to the more detailed definition of the individual task behavior in a Use Case is a contribution (See Sub-chapter 3.5.3.1).

8.1.2.3 RM-ODP Integration for Abstraction Guidance

Most of the other methodologies do not explicitly discuss the natural project flow through refining the solution from general expression of goals at a high level of abstraction to specific details of the implementation. The RM-ODP provides an abstraction guidance framework, which provides order to the design questions. The order in which the design questions are asked is important because of the consequences of the answers. A contribution of the SAMEM is to explicitly incorporate moving through the abstraction hierarchy in an orderly manner.

The use of the RM-ODP helps to achieve the professional engineering goal of due diligence. Through guiding the project from high-level abstract expressions of the solution to detailed expressions, the more impactful questions and decisions are addressed first.

8.1.2.4 Software Engineering First Principle Guidance and Checks

The main contribution of the Software Engineering First Principle idea is to have a set of domain independent metrics by which the solution and the project can be evaluated. Some of the principles are designed to provide correctness checks on the overall quality of the solution progress. Quality in this sense is much more than no defects or bugs in the implementation or code; rather it is also fitness for use and achievement of the stakeholder goals. Other principles enable or stimulate innovation in the solution design by questioning assumptions.

8.2 Future Work

There are a number of areas for the future SAMEM work. The areas can be classified into improvements of the current SAMEM shortcomings, expansion into other domains, even tighter integration with the SEMAT standard, and tools to support the SAMEM.

8.2.1 The SAMEM Improvements

Comments from the surveys indicate that for some of the team members difficulties arose in understanding some of the SAMEM concepts. The improvements needed are in the areas of better explanation of the purpose and definition of the poorly understood concepts. The concept education will need to be incorporated into introduction material at the start of a new project. As the concepts come into usage during the project, short refresher material will be beneficial.

The comments indicate that improved communication of the nature of the RM-ODP standard and its application to controlling the abstraction level of the project work is very important. More explanation of and rationale for the RM-ODP framework is essential.

The role of the state machine requires better communication. This is especially needed for stakeholders with little to no familiarity with state machines, such as those with a business background. Stakeholders with an engineering, science, or mathematical background had little to no difficulty with the state machine concepts and usage.

In general, the SAMEM helped the process move along, but in some cases the communication was less than optimal. Outside of the positive comments, there are comments that indicate that education about the process and artifacts can be improved. These comments imply a better training of the participants in the overall project process and the artifact purposes.

8.2.2 Expand the SAMEM into Other Domains

The SAMEM has been intensively applied in the domain of software solutions supporting the medical device industry. As stated in Sub-chapter 7.2, three other examples of the initial application of the SAMEM are underway. At the time of this thesis, insufficient

data has been collected to determine what improvements and enhancements could be beneficial. Continued monitoring of these projects is needed.

One area for future investigation is the application of the first principles to another solution domain. This would verify that the first principles really are first principles of software engineering and would enable the discovery of other first principles and patterns.

The SAMEM has not been applied in the domain of cyber-physical systems and would likely need enhancement.

8.2.3 UML Extensions Based on SOD Idea

As indicated in Sub-chapter 6.2, the SOD idea of composing multiple models into a single image or drawing can be extended to the UML. While some preliminary possibilities were expressed, there remains work to be done. The extension must be fully compatible with the UML. The extensions should be optional, so that a particular tool does not have to implement them to be compliant with the standard. The extensions need to be implementable in a tool. It is also likely that the initial extension expressions have some shortcomings.

8.2.4 The SAMEM and the SEMAT Integration

The mapping in Sub-chapter 7.4.2.1 shows that many of the SAMEM SEFPs relate to the SEMAT theory area. This is also a possibility for future work. The incorporation of the SEMAT ideas of States describing progress and Checklists into the SAMEM would be useful. The first area in the SAMEM to apply the States and Checklist ideas would be the three-phase framework for the general project process. The factoring of the SEMAT Software Solution Alpha into the finer granularity of the RM-ODP Viewpoints would combine the strengths of both.

The SAMEM can be strengthened by defining its concepts more rigorously utilizing the ESSENCE graph grammar work [HSG16]. This would involve the analysis of the Alphas, States, and Checklists for the aspects that applied to the SAMEM and the extensions needed to cover SAMEM concepts.

8.2.5 Tool Possibilities

Tools are another area for research and development. A more comprehensive set of mutually supporting software engineering CAD (Computer Aided Design) tools are needed. The model can be rendered in various formats, individual views can be combined into a software engineering drawing, and then the model can be passed to a CAE (Computer Aided Engineering Analysis) tool and later to a CAM (Computer Aided Manufacturing) tool. The author believes the mindset of a UML editor blocks progress in this area.

To support the CAD, CAE, and CAM tools, a repository other than a simple directory and file versioning system is needed. The project models and other artifacts have content and there are relationships between the artifact content components. These two aspects, content and relationships, need to be stored and versioned independently. The

components of the SEMAT, such as the Alphas, States, and Checklists, can be stored together in the same repository. Relationships between the solution artifacts, such as a requirement and the Checklist item it satisfies, and the associated State change can be created and controlled.

The ESSENCE visualization tool described in [BSBG17] could also be extended to help with the SAMEM process visualization.

8.2.5.1 SAMEM Model to Code Model Transformations

As briefly discussed in Sub-chapter 7.1.4.1.1, there is a possibility of transforming SAMEM-IM and SAMEM-PM models into initial models in a modeling language such as *UML/P* [Rum16]. The *UML/P* is well-defined. While the SAMEM-IM and SAMEM-PM provide a starting point for the transformation, it is unknown if they are well-defined enough at this time to support a robust model transformation process. There is more research to be done in this area.

To facilitate the possibilities for round-trip engineering, the Alpha, State, and Checklist concepts from the SEMAT could potentially be employed against the artifacts to assist in indicating rework needed to a preceding artifact.

Bibliography

[AGBA10] Farooque Azam, Hina Gull, Saeeda Bibi and Sameera Amjad. Back & forth (bnf) software process model. In *2010 Second International Conference on Computer Engineering and Applications*. IEEE, 2010.

[AIS+77] Christopher Alexander, Sara Ishikawa, Murry Silverstein, Max Jacobson, Ingrid Fiksdahl-King and Shlomo Angel. *A Pattern Language*. Oxford University Press, 1977.

[AM11] Daniel Amyot and Gunter Mussbacher. User requirements notation: The first ten years, the next ten years. *Journal of Software*, 6(5):747 – 768, 2011.

[Amb02] Scott W. Ambler. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. Wiley, 2002.

[BC12] Joy Beatty and Anthony Chen. *Visual Models for Software Requirements*. Microsoft Press, 2012.

[Bec00] Ken Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000.

[BH98] Hugh Beyer and Karen Holtzblatt. *Contextual Design*. Academic Press, 1998.

[BPKR09] Brian Berenbach, Daniel Paulish, Juergen Kazmeier and Arnold Rudorfer. *Software & Systems Requirements Engineering: In Practice*. McGraw Hill, 2009.

[BPM16] Business process model and notation (bpmn), version 2.0, last accessed March 2016.

[BR05] Michael Blaha and James Rumbaugh. *Object-Oriented Modeling and Design with UML*. Pearson Prentice Hall, 2005.

[BRJ99] Grady Booch, James Rumbaugh and Ivar Jacobson. *The Unified Modeling Language Use Guide*. Addison-Wesley, 1999.

[Bro95] Frederick P. Brooks. *The Mythical Man-Month Essays on Software Engineering*. Addison-Wesley, 1995.

[Bro10] Frederick P. Brooks. *The Design of Design: Essays from a Computer Scientist*. Addison-Wesley, 2010.

[BSBG17] Sebastian Brandt, Michael Striewe, Fabian Beck and Michael Goedicke. A dashboard for visualizing software engineering processes based on essence. In *2017 IEEE Working Conference on Software Visualization,*, Seite 134 – 138. IEEE, 2017.

[CFJ+17] Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Jim Steel and Didier Vojtisek. *Engineering Modeling Languages*. CRC Press, 2017.

[CGHM13] Patrice Caire, Nicolas Genon, Patrick Heymans and Daniel L. Moody. Visual notation design 2.0: Towards user comprehensible requirements engineering notations. In *21st IEEE Requirements Engineering Conference*, Seite 115 – 124, New York, NY, USA, 2013. IEEE.

[Coh10] Mike Cohn. *Succeeding with Agile: Software Development Using SCRUM*. Addison-Wesley, Boston, MA, 2010.

[Col05] Bob Colwell. Complexity in design. *IEEE Computer*, 38(10):10 – 12, 2005.

[Dic05] Jeremy Dick. Design traceability. *IEEE Software*, 22(6):14 – 16, 2005.

[Essa] Essence reference guides. last accessed December 2017.

[Essb] Esswork practice workbench. last accessed May 2016.

[FM15] Joao M. Fernandes and Ricardo J. Machado. *Requirements in Engineering Projects*. Springer International Publishing, 2015.

[Fow03] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, London, United KIngdom, 2003.

[Fow04] Martin Fowler. *UML Distilled*. Addison-Wesley, London, United KIngdom, third edition Edition, 2004.

[FR07] Robert B. France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *Future of Software Engineering (FOSE '07)*, Seite 37 – 54. IEEE Computer Society, 2007.

[Fra03] David S. Frankel. *Model Driven Architecture, Applying MDA to Enterprise Computing*. Wiley Publishing, 2003.

[GCH13] Olly Gotel and Jane Cleland-Huang. Requirements engineering's next top model. *IEEE Software*, 30(6):24 – 29, 2013.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.

[GHR17] Timo Greifenberg, Steffen Hillemacher and Bernhard Rumpe. *Towards a Sustainable Artifact Model: Artifacts in Generator-Based Model-Driven Projects*. Shaker Verlag, Aachen, 2017.

[Gli10] Martin Glinz. Very lightweight requirements modeling. In *18th IEEE International Requirements Engineering Conference*. IEEE, 2010.

[GPHS08] César Gonzáles-Pérez and Brian Henderson-Sellers. *Metamodelling for Software Engineering*. John Wiley & Sons,Ltd., 2008.

[GXX11] Fan Guo, Bainan Xia and Fei Xue. Analysis on software processes and enhancements for rup. 2011.

[HF68] William C. Howell and Alfred H. Fuchs. Population stereotypy in code design. *Organizational Behavior and Human Performance*, (3):310 – 339, 1968.

[HH08] Chip Heath and Dan Heath. *Made to Stick: Why Some Ideas Take Hold and Others Come Unstuck*. Arrow Books, 2008.

[Hir02] Michael Hirsch. Making rup agile. In *OOPSLA*, 2002.

[HK99] Jungpil Hahn and Jinwoo Kim. Why are some diagrams easier to work with? effects of diagrammatic representation on the cognitive integration process of system analysis and design. *ACM Transactions on Computer-Human Interaction*, 6(3):181 – 213, 1999.

[Hoc99] Dee Hock. *Birth of the Chaordic Age*. Berrett-Koehler Publishers, Inc., 1999.

[HRW11] John Hutchinson, Mark Rouncefield and Jon Whittle. Model-driven engineering practices in industry. In *ICSE '11: Proceedings of the 33rd International Conference on Software Engineering*, Seite 633 – 642, New York, NY, USA, 2011. ACM.

[HSG16] S. Holtappels, M. Striewe and Michael Goedicke. From essence to theory oriented software engineering. In *SOFSEM 2016*, Seite 43 – 50, 2016. LNCS 9587.

[HWRK11] John Hutchinson, Jon Whittle, Mark Rouncefield and Steiner Kristoffersen. Empirical assessment of mde in industry. In *ICSE '11: Proceedings of the 33rd International Conference on Software Engineering*, Seite 471 – 480, New York, NY, USA, 2011. ACM.

[IDF] Interaction design foundation. last accessed December 2017.

[ISO98] Open distributed processing - reference model (rm-odp) iso 10746-1, 1998.

[JEJ12] Pontus Johnson, Mathias Ekstedt and Ivar Jacobson. Where's the theory for software engineering? *IEEE Software*, September/October:94 – 95, 2012.

[Jur15] Ivan Jureta. *The Design of Requirements Modelling Languages*. Springer International Publishing, 2015.

[KGK07] Riaan Klopper, Stefan Gruner and Derrick G. Kourie. Assessment of a framework to compare software development methodologies. In *SAICSIT*, 2007.

[KM] Mira Kajko-Mattsson. How can you support your software development method with essence? last accessed December 2017.

[KP02] Elizabeth A. Kemp and Chris Phillips. The high level design of object-oriented user interfaces: a review of methods. In *ACM SIGCHI*, 2002.

[KWB03] Anneke G. Kleppe, Jos B. Warner and Wim Bast. *MDA Explained: the Model Driven Architecture: Practice and Promise*. Pearson Education, Inc, 2003.

[LS87] J. H. Larkin and H. A Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):36 – 44, 1987.

[Mat02] Dan Matheson. Experiences using mda for enterprise it management. In *Integrate 2002, OMG*, 2002.

[Mat04] Dan Matheson. Cad data model with design notes, 2004.

[Mat05] Dan Matheson. Innovation information management model, 2005.

[Mat09] Dan Matheson. Object model for decision and issue tracking, 2009.

[Mat11] Dan Matheson. Modeling requirements: The customer communication. In *2014 IEEE 5th International Workshop on Requirements Prioritization and Communication (RePriCo)*, Seite 471 – 480, New York, NY, USA, 2011. IEEE.

[Mat17] Dan Matheson. A proposal of practices, processes and models that enable innovation potential. In *DISE Workshop in ICSE 2017*, 2017.

[MHM10] Daniel L. Moody, Patrick Heymans and Raimundas Matulevicius. Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. *Requirements Engineering Journal*, 15(2):141 – 175, 2010.

[Moo09] Daniel L. Moody. The 'physics'of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6), 2009.

[MRRH05] Dan Matheson, Indraksi Ray, Indrjit Ray and Siv Hilde Houmb. Building security requirement patterns for increased effectiveness early in the development process. In *SREIS Workshop 13th IEEE International Requirements Engineering Conference*. IEEE, 2005.

[MvH08] Daniel L. Moody and J van Hillegersberg. Evaluating the visual syntax of uml: An analysis of the cognitive effectiveness of the uml family of diagrams. In *1st International Conference on Software Language Engineering*, Seite 16 – 34, Berlin, Germany, 2008. Springer-Verlag. LNCS 5452.

[Mye75] Glenford Myers. *Reliable Software through Composite Design*. Petrocelli/Charter, 1975.

[NAS03] NASA. Error cost escalation through the project life cycle. *NASA Johnson Space Center*, 2003.

[NK11] Juan Pablo Napoli and Kalinka Kaloyanova. An integrated approach for rup, ea, soa and bpm implementation. In *International Conference on Computer Systems and Technologies*, 2011.

[OCH11] Jorge A. Osorio, Michel R.V. Chaudron and Werner Heijstek. Moving from waterfall to iterative development – an empirical evaluation of advantages, disadvantages and risks of rup. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, Seite 453 – 460, 2011.

[OMG00] Object management group product data management enablers specification v1.3, 2000. last accessed January 2015.

[OMG08] Object management group software & systems process engineering metamodel (spem), version 2, 2008. last accessed April 2014.

[OMG12a] Object management group diagram definition specification, version 1.0, 2012. last accessed January 2015.

[OMG12b] Object management group essence - kernel and language for software engineering methods, version 1.1, 2012. last accessed May 2016.

[OMG14a] Object management group model driven architecture (mda)mda guide rev. 2.0, 2014. last accessed December 2014.

[OMG14b] Object management group meta object facility core specification version 2.4.2, 2014. last accessed January 2015.

[OMG15a] Object management group systems modeling language, version 1.4, 2015. last accessed May 2016.

[OMG15b] Object management group unified modeling language (uml), version 2.5, 2015. last accessed May 2016.

[OP97] Paul Oman and Shari Lawrence Pfleeger. *Applying Software Metrics*. IEEE Computer Society Press, 1997.

[Par72] David L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053 – 1058, December 1972.

[Pet96] Henry Petroski. *Invention by Design - How Engineers Get From Thought to Thing*. Harvard University Press, 1996.

[PK00] Jr. Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley, second edition Edition, 2000.

[RJB99] James Rumbaugh, Ivar Jacobson and Grady Booch. *The Unified Modeling Language Reference misc*. Addison-Wesley, Boston, MA, 1999.

[RM-98] Iso open distributed processing - reference model (rm-odp). iso 10746-1, 1998.

[RR99] Suzanne Robertson and James Robertson. *Mastering the Requirements Process*. Addison-Wesley, Boston, MA, 1999.

[Rum16] Bernhard Rumpe. *Modeling with UML: Language, Concepts, Methods*. Springer International, July 2016.

[Rum17] Bernhard Rumpe. *Agile Modeling with UML: Code Generation, Testing, Refactoring*. Springer International, May 2017.

[RvdHMRM04] Roshanak Roshandel, Andre van der Hoek, Marija Mikic-Rakic and Nenad Medvidovic. Mae – a system model and environment for managing architectural evolution. *ACM Transactions on Software Engineering and Methodology*, 13(2):240 – 276, 2004.

[SEM98] Software engineering method and theory (semat), 1998.

[SEM16] Wikipedia - software engineering method and theory (semat), last accessed May 2016.

[SM11] Mark Speer and Dan Matheson. Applying the tcua requirements manager application in a medical device r&d environment. In *Siemens PLM Connection Annual Users Conference*, 2011.

[SSS08] Forrest Shull, Janice Singer and Dag I. K. Sjóberg. *Guide to Advanced Empirical Software Engineering.* Springer, 2008.

[Tan15] Mohsan Tanveer. *Agile For Large Scale Projects - A Hybrid Approach.* NSEC, 2015.

[TFR05] Dan Turk, Robert France and Bernhard Rumpe. Assumptions underlying agile software development processes. *Journal of Database Management*, 16(5):62 – 87, 2005.

[TW95] Michael Treacy and Fred Wiersema. *The discipline of market leaders: Choose your customers, narrow your focus, dominate your market.* Addison-Wesley, Boston, MA, 1995.

[URN12] Itu=t z.151 user requirements notation – language definition, 10/2012, 2012.

[V-M16a] V-model, last accessed May 2016.

[V-M16b] V-model software development, last accessed May 2016.

[V-M16c] V-modell xt (extreme tailoring), last accessed May 2016.

[WF86] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition.* Addison-Wesley, 1986.

[Whi15] Jon Whittle. Tutorial: How industry uses mde. *Models Conference 2015*, 2015.

[WHR14] Jon Whittle, John Hutchinson and Mark Rouncefield. The state of practice in model-driven engineering. *IEEE Software*, 31(3):79 – 85, May/June 2014.

[Win96] Terry Winograd. *Bringing Design to Software.* ACM Press, New York, NY, USA, 1996.

[WRH+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell and Anders Wesslén. *Experimentation in Software Engineering.* Springer, 2000.

[Yu95] Eric S.-K. Yu. *Modelling strategic relationships for process reengineering.* Dissertation, University of Toronto, Canada, 1995.

[Yu97] Eric S.-K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, Seite 226 – 235, 1997.

[Yu09] Eric S. Yu. Social modeling and i*. In Alexander T. Borgida, Vinay K. Chaudhri, Paolo Giorgini and Eric S. Yu, Editoren, *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*. Springer, 2009.

[ZHG05] Wolfgang Zuser, Stefan Heil and Thomas. Grechenig. Software quality development and assurance in rup, msf and xp – a comparative study. In *WoSG*, 2005.

Glossary

Definition of terms and acronyms used in the thesis.

Agile Software Development —is a project process approach or methodology that advocates a series of small steps to develop the solution. A key component is constant contact with the solution customer. It first started with a focus on planning, developing, and testing code in small pieces as opposed to first doing all the planning, then all the development, and finally all the testing [Amb02], [Coh10], [TFR05].

Approval Flow —the set of detailed actions and decisions which changes the status, or state of the information unit, or set of units.

BPMN —Business Process Modeling Notation acronym.

Business Flow —consists of the major business steps or tasks used to create a significant part of the information model, starting with nothing and proceeding to the completed business information.

Cohesion —expresses the singularity of purpose of the module. A module with the best cohesion stands on its own and does one thing [Mye75].

Computational Equivalence —informationally equivalent and the inferences drawn from one representation can be easily drawn from the other.

Coupling —indicates the closeness of interaction between two modules with no interaction the best [Mye75].

CRUD —Create-Retrieve-Update-Delete pattern on the information model units used within the *Business Flow*. The CRUD pattern acts as a completeness check on the tasks in the *Business Flow*. Each CRUD action should appear in a *Business Flow* associated with an information model unit.

Customer —is the generic term for the people or organization paying for the product or solution. This can include the *users* who actually use and interact with the solution.

Due Diligence —is the professional behavior of investigating and honestly evaluating alternatives to achieve a successful solution.

ETL —Extract Transform Load, a common data migration process of extracting the old data from an existing repository, transforming the data to the new format which might include setting new data values, and loading the transformed data into the new repository.

Fit Criteria —is a value that can be measured in the solution via testing to verify the requirement is met.

Framework —a structure for organizing the model artifacts, set of recommendations for the next tasks, and a roadmap for systematically advancing from a simple high-level statement of the problem to a realization acceptable to the stakeholders.

Incremental —progressing towards a solution in small, controlled, and well-defined steps.

Informational Equivalence —all the information in one representation is inferable from the other.

Information Model —the information definitions and partitioning for the solution in business terms, not computer science terms.

Iteration —one cycle of a series where each cycle consists of the same general steps of work and produces a result.

Iterative & Incremental —the approach of doing a small amount of work within a bounded time limit that produces an increment in completeness in some solution or project artifacts.

Methodology —a collection of artifacts, tools, and processes related in a structured way with a general, but adaptable choreography of execution and used to create a product or solution. Methodology as defined by the Oxford English Dictionary: *a system of methods used in a particular area of study or activity.* According to Wikipedia, a methodology consists of the following concepts: paradigm, theoretical model, phases, and qualitative or quantitative techniques.

Model —a human construct representing some aspects of reality that enables better understanding and communication of a particular perspective of a problem and/or solution.

Module —a code unit of implementation, similar to a class in the Java programming language.

OOTB —Out Of The Box, using the features and capabilities of a commercial application as delivered by the vendor without any change.

Paradigm —defined in the Oxford English Dictionary: *a typical example or pattern of something.* Another definition of paradigm is a distinct set of concepts, thought patterns, and standards.

Process —a set of tasks with a defined choreography of execution that produces a specific result.

Project —a defined set of work with a start point and end point, which is limited in time, people, money, and other resources, that produces a product or solution.

RM-ODP —Open Distributed Processing – Reference Model (RM-ODP), International Standards Organization (ISO) standard 10746.

Root Cause —the core deficiency that causes the problem and might be observed through one or more symptoms.

SAMEM —Software Agile Modeling and Engineering Methodology acronym.

SCRUM —an iterative & incremental agile code development process [Cohn10] using the idea of a *sprint*, which normally is one week to one month long, to deliver a potentially shippable increment of the solution.

SEFP —Software Engineering First Principles —a set of first principles for reasoning about the software solution development process to assure high quality.

SEMAT —Software Engineering Method and Theory acronym.

SOD —Solution Overview Drawing acronym.

Stakeholder —is someone who in some way is impacted by or concerned with the solution.

SurveyMonkey —a commercial application for doing electronic surveys, used to collect empirical evidence for the thesis. The application provides for the definition of the survey questions, their organization, a web link for accessing the survey, the collection of the responses, and analysis of the responses. www.surveymonkey.com

Symptom —an observed indicator of a problem, such as excessive rework.

Visual Model —a model consisting of primarily graphical elements annotated with text as opposed to a pure text description.

Appendix A

Customer Questionnaire

Paper Prototype Customer Experiences

Survey Purpose and Overview

This questionnaire concerns your experiences with the requirements gathering and project process known as the "Paper Prototype". As part of a research effort improvements to the Paper Prototype process are being made. This questionnaire is intended to gather feedback on your experiences with the process. In order to improve the Paper Prototype process a series of questions have been devised along with the opportunity to add experiences not covered by the questions.

Efforts have been made to keep the questionnaire as short as possible, while still covering the essential areas. Please be as honest as possible with the strengths, weaknesses, advantages and disadvantages of the Paper Prototype process as you experienced it. When comparisons to other project processes will help to clarify a strength or weakness, please feel free to add that information. You do not have to answer any question that you do not want to.

In compiling and using the results anonymity of the respondents will be preserved.

1

Paper Prototype Customer Experiences

Overall Paper Prototype Evaluation

This section contains questions about your overall evaluation of the Paper Prototype process in relation to the progress and management of the project.

1. What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.

- ◯ Very Effective: good solution with a minimum number of people in minimum time frame.
- ◯ Moderately Effective: good solution with a reasonable number of people in a reasonable time frame.
- ◯ Effective: acceptable solution with an acceptable number of people in an acceptable time frame.
- ◯ Poorly Effective: minimally acceptable solution with too many people in too long a time frame.
- ◯ Ineffective: an unacceptable solution with too many people in too long a time frame.

Additional comments:

2. How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?

- ◯ Very Positive
- ◯ Moderately Positive
- ◯ No Impact
- ◯ Moderately Negative
- ◯ Very Negative

2

3. Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?

- ◯ Yes
- ◯ Sometimes
- ◯ No

In which ways?

4. In your experience did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?

- ◯ Yes
- ◯ Sometimes
- ◯ No

In which ways?

5. How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom an improvement
- ◯ Not Better

3

6. How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?

- ◯ Very Helpful
- ◯ Helpful, but with some limitations
- ◯ Occasionally Helpful
- ◯ Not Helpful

Additional comments on the project management activities:

4

Paper Prototype Customer Experiences

Communication Evaluation

A key aspect of the Paper Prototype process is the increased use of visual or graphical representations of the requirements over the more traditional text-based mechanisms. The next set of questions focuses on the use of the graphical representations and models for the communication amongst the project members.

7. What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?

- ◯ Very Effective
- ◯ Somewhat Effective
- ◯ No Difference
- ◯ Less Effective
- ◯ A Mismatch

8. What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?

- ◯ Very Effective
- ◯ Somewhat Effective
- ◯ No Difference
- ◯ Less Effective
- ◯ A Mismatch

5

9. What effect did the “1-pager” have on project iteration planning?

- ◯ Very Effective
- ◯ Somewhat Effective
- ◯ No Difference
- ◯ Less Effective
- ◯ A Mismatch

Additional comments on overall graphical representation use:

10. What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

11. Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

6

12. Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

13. Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?

- ◯ Very Helpful
- ◯ Helpful, but with some limitations
- ◯ Occasionally Helpful
- ◯ Seldom Helpful
- ◯ Not Helpful

Additional comments on business information unit modelling:

14. For a new solution did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?

- ◯ Very Helpful
- ◯ Helpful, but with some limitations
- ◯ Occasionally Helpful
- ◯ Seldom Helpful
- ◯ Not Helpful

7

15. Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?

- ◯ Very Useful
- ◯ Useful, but with some limitations
- ◯ Occasionally Useful
- ◯ Seldom Useful
- ◯ Not Useful

Additional comments on the business flow model:

8

Paper Prototype Customer Experiences

Requirements Quality

One hypothesis behind the Paper Prototype process of using graphical specification techniques is that a more compact and higher quality specification is produced.

16. Do you feel that the visual models of the business information and behavior requirements affected the **speed** of the review task for correctness and completeness?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

17. Do you feel that the visual models of the business information and behavior requirements affected the **accuracy** of the review task for correctness and completeness?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

18. How do you feel that the use of visual models for the requirements affected the**clarity** of the requirements specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

9

19. How do you feel that the use of visual models for the requirements affected the**size** of the requirements specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

Additional comments on achieving requirements quality, correctness and completeness:

20. Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

21. If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

10

22. Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?

- ○ Yes, Better
- ○ Mostly Better, but with some limitations
- ○ Occasionally Better
- ○ Seldom Better
- ○ Not Better

23. What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?

- ○ Yes, Better
- ○ Mostly Better, but with some limitations
- ○ Occasionally Better
- ○ Seldom Better
- ○ Not Better

Additional comments on the incremental & iterative process impacts:

11

Paper Prototype Customer Experiences

General Paper Prototype Experience

24. Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?

- ◯ Yes
- ◯ No

Please explain:

25. Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?

- ◯ Yes
- ◯ No

Please explain:

12

26. Experience in using the Paper Prototype process (time in months)?

- ○ Up to 3 months
- ○ 3 – 6 months
- ○ 6 – 12 months
- ○ 12 – 24 months
- ○ More than 24 months

27. Previous experience with software-based solution development projects?

- ○ None
- ○ 1 project
- ○ 2 – 4 projects
- ○ More than 4 projects

28. What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.

- ☐ Business/customer stakeholder
- ☐ Technical member of solution development team
- ☐ Solution technology technical expert
- ☐ Non-technical member of solution development team
- ☐ Project manager
- ☐ Domain technical expert
- ☐ Other (please specify)

13

29. What is your evaluation of the learning effort needed to participate in the Paper Prototype process?

- ◯ Easy: understood the concepts immediately
- ◯ Medium: some concepts were quick to learn, while others took longer
- ◯ Hard: took longer than expected
- ◯ Extremely Hard: never fully comfortable with the process

Any additional comments on the Paper Prototype process:

14

Appendix B

Developer Questionnaire

Paper Prototype Developer Experiences

Survey Purpose and Overview

This questionnaire concerns your experiences with the requirements gathering and project process known as the "Paper Prototype". As part of a research effort improvements to the Paper Prototype process are being made. This questionnaire is intended to gather feedback on your experiences with the process. In order to improve the Paper Prototype process a series of questions have been devised along with the opportunity to add experiences not covered by the questions.

Efforts have been made to keep the questionnaire as short as possible, while still covering the essential areas. Please be as honest as possible with the strengths, weaknesses, advantages and disadvantages of the Paper Prototype process as you experienced it. When comparisons to other project processes will help to clarify a strength or weakness, please feel free to add that information. You do not have to answer any question that you do not want to.

In compiling and using the results anonymity of the respondents will be preserved.

1

Paper Prototype Developer Experiences

Overall Paper Prototype Evaluation

This section contains questions about your overall evaluation of the Paper Prototype process in relation to the progress and management of the project.

1. What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.

- ◯ Very Effective: good solution with a minimum number of people in minimum time frame.
- ◯ Moderately Effective: good solution with a reasonable number of people in a reasonable time frame.
- ◯ Effective: acceptable solution with an acceptable number of people in an acceptable time frame.
- ◯ Poorly Effective: minimally acceptable solution with too many people in too long a time frame.
- ◯ Ineffective: an unacceptable solution with too many people in too long a time frame.

Additional comments:

2. How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?

- ◯ Very Positive
- ◯ Moderately Positive
- ◯ No Impact
- ◯ Moderately Negative
- ◯ Very Negative

2

3. How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom an improvement
- ◯ Not Better

4. How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?

- ◯ Very Helpful
- ◯ Helpful, but with some limitations
- ◯ Occasionally Helpful
- ◯ Not Helpful

Additional comments on the project management activities:

3

Paper Prototype Developer Experiences

Communication Evaluation

A key aspect of the Paper Prototype process is the increased use of visual or graphical representations of the requirements over the more traditional text-based mechanisms. The next set of questions focuses on the use of the graphical representations and models for the communication amongst the project members.

5. What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?

- ◯ Very Effective
- ◯ Somewhat Effective
- ◯ No Difference
- ◯ Less Effective
- ◯ A Mismatch

6. What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?

- ◯ Very Effective
- ◯ Somewhat Effective
- ◯ No Difference
- ◯ Less Effective
- ◯ A Mismatch

4

7. What effect did the "1-pager" have on project iteration planning?

- ◯ Very Effective
- ◯ Somewhat Effective
- ◯ No Difference
- ◯ Less Effective
- ◯ A Mismatch

Additional comments on overall graphical representation use:

8. What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

9. Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

5

10. Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

11. Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?

- ◯ Very Helpful
- ◯ Helpful, but with some limitations
- ◯ Occasionally Helpful
- ◯ Seldom Helpful
- ◯ Not Helpful

Additional comments on business information unit modelling:

6

12. Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?

- ◯ Very Useful
- ◯ Useful, but with some limitations
- ◯ Occasionally Useful
- ◯ Seldom Useful
- ◯ Not Useful

Additional comments on the business flow model:

7

Paper Prototype Developer Experiences

Requirements Quality

One hypothesis behind the Paper Prototype process of using graphical specification techniques is that a more compact and higher quality specification is produced.

13. Do you feel that the visual models of the business information and behavior requirements affected the **speed** of the review task for correctness and completeness?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

14. How do you feel that the use of visual models for the requirements affected the**clarity** of the requirements specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

15. How do you feel that the use of visual models for the requirements affected the**size** of the requirements specification?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

Additional comments on achieving requirements quality, correctness and completeness:

8

Paper Prototype Developer Experiences

Development and Testing Evaluation

One of the Paper Prototype goals is to ease the requirements gathering and specification transition to the development activities.

16. Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

17. Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

9

18. Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

Additional comments on solution design activities:

19. Do you feel that the visual components of the requirements specification had effects on developing tests?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

20. Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

10

21. Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

Additional comments on testing activities:

22. If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better
- ◯ Not Applicable

23. Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

11

24. What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?

- ◯ Yes, Better
- ◯ Mostly Better, but with some limitations
- ◯ Occasionally Better
- ◯ Seldom Better
- ◯ Not Better

25. Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?

- ◯ Yes
- ◯ Sometimes
- ◯ About the same
- ◯ No, it added effort

Additional comments on the incremental & iterative process impacts:

12

Paper Prototype Developer Experiences

General Paper Prototype Experience

26. Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?

- ◯ Yes
- ◯ No

Please explain:

27. Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?

- ◯ Yes
- ◯ No

Please explain:

13

28. Experience in using the Paper Prototype process (time in months)?

- ◯ Up to 3 months
- ◯ 3 – 6 months
- ◯ 6 – 12 months
- ◯ 12 – 24 months
- ◯ More than 24 months

29. Previous experience with software-based solution development projects?

- ◯ None
- ◯ 1 project
- ◯ 2 – 4 projects
- ◯ More than 4 projects

30. What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.

- ☐ Business/customer stakeholder
- ☐ Technical member of solution development team
- ☐ Solution technology technical expert
- ☐ Non-technical member of solution development team
- ☐ Project manager
- ☐ Domain technical expert
- ☐ Other (please specify)

14

31. What is your evaluation of the learning effort needed to participate in the Paper Prototype process?

- ○ Easy: understood the concepts immediately
- ○ Medium: some concepts were quick to learn, while others took longer
- ○ Hard: took longer than expected
- ○ Extremely Hard: never fully comfortable with the process

Any additional comments on the Paper Prototype process:

15

Appendix C

Customer Response 1

Paper Prototype Customer Experiences SurveyMonkey

#1

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Thursday, April 27, 2017 12:05:31 PM
Last Modified: Thursday, April 27, 2017 12:13:58 PM
Time Spent: 00:08:27
IP Address:

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Very Effective: good solution with a minimum number of people in minimum time frame. , Additional comments: I can't imagine proceeding without it
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Very Positive
Q3: **Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?**	Yes, In which ways? People could understand what they were getting very early in the process
Q4: **In your experience did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?**	Yes, In which ways? It allowed buy-in of the process very early and flushed out internal dissent or misunderstanding.
Q5: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Yes, Better
Q6: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Very Helpful

PAGE 3: Communication Evaluation

Q7: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective

1 / 4

Q8: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Very Effective
Q9: **What effect did the "1-pager" have on project iteration planning?**	Somewhat Effective, Additional comments on overall graphical representation use: I don't want to minimize the total effort in developing a good project plan.
Q10: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	Yes, Better
Q11: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Yes, Better
Q12: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	Yes, Better
Q13: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Very Helpful
Q14: **For a new solution did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?**	Very Helpful
Q15: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Very Useful

PAGE 4: Requirements Quality

Q16: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Yes, Better
Q17: **Do you feel that the visual models of the business information and behavior requirements affected the accuracy of the review task for correctness and completeness?**	Yes, Better

Q18: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better
Q19: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Yes, Better
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Yes, Better
Q21: **If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?**	Yes, Better
Q22: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q23: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Yes, Better

PAGE 5: General Paper Prototype Experience

Q24: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	No, Please explain: Gathering requirements is simply a matter of getting to the truth about what has to be delivered and in gaining consensus on the solution. Any means will do that if the participants are knowledgeable and engaged. With a visual graphical approach, it makes the participants understand what is happening quickly and facilitate learning as well.
Q25: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	Yes
Q26: **Experience in using the Paper Prototype process (time in months)?**	More than 24 months
Q27: **Previous experience with software-based solution development projects?**	More than 4 projects
Q28: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Business/customer stakeholder

Q29: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Easy: understood the concepts immediately

Appendix D

Customer Response 2

Paper Prototype Customer Experiences SurveyMonkey

#2

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Wednesday, June 07, 2017 2:11:23 PM
Last Modified: Wednesday, June 07, 2017 2:21:32 PM
Time Spent: 00:10:08
IP Address:

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Very Effective: good solution with a minimum number of people in minimum time frame. , Additional comments: I believe the process allows for a quick verification of the process prior to investing in technical solutions and adding complex elements.
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Moderately Positive
Q3: **Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?**	Yes
Q4: **In your experience did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?**	Yes, In which ways? when proper resource engagement was in place
Q5: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Mostly Better, but with some limitations
Q6: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Very Helpful

PAGE 3: Communication Evaluation

Q7: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective

1 / 4

Q8: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Somewhat Effective
Q9: **What effect did the "1-pager" have on project iteration planning?**	Somewhat Effective, Additional comments on overall graphical representation use: Enabled a common framework for alignment of the team on overall project direction
Q10: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	Mostly Better, but with some limitations
Q11: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Yes, Better
Q12: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	Mostly Better, but with some limitations
Q13: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Very Helpful, Additional comments on business information unit modelling: Once the team resources understood the state machine model, it was very helpful in the project definition process.
Q14: **For a new solution did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?**	Very Helpful
Q15: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Useful, but with some limitations, Additional comments on the business flow model: Focus is on the primary flow as it should be. The 1-pager does help in consolidating solutions as well

PAGE 4: Requirements Quality

Q16: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Mostly Better, but with some limitations

Paper Prototype Customer Experiences — SurveyMonkey

Q17: **Do you feel that the visual models of the business information and behavior requirements affected the accuracy of the review task for correctness and completeness?**	Yes, Better
Q18: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better
Q19: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Yes, Better, Additional comments on achieving requirements quality, correctness and completeness: The specifications would have been much larger if a textual only approach was used.
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Yes, Better
Q21: **If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?**	Mostly Better, but with some limitations
Q22: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q23: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Mostly Better, but with some limitations , Additional comments on the incremental & iterative process impacts: I believe it sets up a smooth transition to the technical team for delivery. There are still some opportunities to move along more quickly with the steps and perhaps even break the process into smaller components.

PAGE 5: General Paper Prototype Experience

Q24: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	No, Please explain: I have always had some tables, flows and pictures in specification so for me it fit with and extended the model I thought worked well for spec development.
Q25: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	Yes, Please explain: Easier to digest, manage and handoff to other teams

3 / 4

Q26: **Experience in using the Paper Prototype process (time in months)?**	More than 24 months
Q27: **Previous experience with software-based solution development projects?**	More than 4 projects
Q28: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Business/customer stakeholder
Q29: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Medium: some concepts were quick to learn, while others took longer , Any additional comments on the Paper Prototype process: State diagrams and new acronyms and words are a bit of a learning curve but should be easy to grasp by the average corporate technology or business worker.

Appendix E

Customer Response 3

Paper Prototype Customer Experiences SurveyMonkey

#3

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Sunday, June 11, 2017 1:25:20 PM
Last Modified: Sunday, June 11, 2017 1:35:57 PM
Time Spent: 00:10:36
IP Address:

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Very Effective: good solution with a minimum number of people in minimum time frame.
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Very Positive
Q3: **Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?**	Yes, In which ways? Improved understanding of project and establishment of requirements
Q4: **In your experience did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?**	Yes, In which ways? Allowed agreement that this was what was desired before extensive investment in program configuration.
Q5: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Yes, Better
Q6: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Very Helpful

PAGE 3: Communication Evaluation

Q7: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective
Q8: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Very Effective

1 / 3

Q9: **What effect did the "1-pager" have on project iteration planning?**	Very Effective
Q10: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	*Respondent skipped this question*
Q11: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Yes, Better
Q12: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	Yes, Better
Q13: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Helpful, but with some limitations , Additional comments on business information unit modelling: State machine model was not always clear to some end users.
Q14: **For a new solution did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?**	Very Helpful
Q15: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Very Useful

PAGE 4: Requirements Quality

Q16: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Yes, Better
Q17: **Do you feel that the visual models of the business information and behavior requirements affected the accuracy of the review task for correctness and completeness?**	Yes, Better
Q18: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better

Q19: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Yes, Better
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Yes, Better
Q21: **If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?**	Yes, Better
Q22: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q23: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Yes, Better

PAGE 5: General Paper Prototype Experience

Q24: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	No
Q25: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	Yes, Please explain: One major advantage was mapping processes, especially since most parts of the organization did not have those and did not have clarity on the extent of interplay and interactions with other parts of the organization. Also enabled communication to management, to ensure all parties were on the same page before committing resources.
Q26: **Experience in using the Paper Prototype process (time in months)?**	More than 24 months
Q27: **Previous experience with software-based solution development projects?**	2 – 4 projects
Q28: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Business/customer stakeholder
Q29: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Easy: understood the concepts immediately

Appendix F

Customer Response 4

Paper Prototype Customer Experiences SurveyMonkey

#4

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Sunday, June 11, 2017 6:24:15 PM
Last Modified: Sunday, June 11, 2017 6:37:02 PM
Time Spent: 00:12:47
IP Address:

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Very Effective: good solution with a minimum number of people in minimum time frame.
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Very Positive
Q3: **Did the Paper Prototype process approach of an iterative & incremental solution creation style establish confidence in project progress and a sense of reasonable return on the investment?**	Yes
Q4: **In your experience did the Paper Prototype process approach of an iterative & incremental solution creation style provide manageable organizational change around the new solution?**	Yes, In which ways? It allowed the people who were going to have to use the system to see that the solution was "real" (going to come into fruition) and to think about changes that they would like to see (forced good though thought in thinking thru the solution deliverables)
Q5: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Yes, Better
Q6: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Helpful, but with some limitations

PAGE 3: Communication Evaluation

Q7: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective

1 / 4

Q8: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Very Effective
Q9: **What effect did the "1-pager" have on project iteration planning?**	Very Effective, Additional comments on overall graphical representation use: A picture that describes the process is always a good way to confirm that all participants are understanding the transactional processes in the same way
Q10: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	Yes, Better
Q11: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Mostly Better, but with some limitations
Q12: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	Yes, Better
Q13: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Helpful, but with some limitations
Q14: **For a new solution did the initial focus on defining the Business Flow behavior have an effect on gathering requirements?**	Very Helpful
Q15: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Very Useful, Additional comments on the business flow model: This is an effective tool to explain to upper management stakeholders

PAGE 4: Requirements Quality

Q16: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Mostly Better, but with some limitations

Q17: **Do you feel that the visual models of the business information and behavior requirements affected the accuracy of the review task for correctness and completeness?**	Yes, Better
Q18: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better
Q19: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Mostly Better, but with some limitations
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Mostly Better, but with some limitations
Q21: If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?	*Respondent skipped this question*
Q22: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q23: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Mostly Better, but with some limitations

PAGE 5: General Paper Prototype Experience

Q24: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	No
Q25: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	No
Q26: **Experience in using the Paper Prototype process (time in months)?**	More than 24 months
Q27: **Previous experience with software-based solution development projects?**	None
Q28: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Business/customer stakeholder, Project manager
Q29: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Medium: some concepts were quick to learn, while others took longer

Appendix G

Developer Response 1

Paper Prototype Developer Experiences SurveyMonkey

#1

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Wednesday, April 26, 2017 5:12:14 PM
Last Modified: Wednesday, April 26, 2017 6:20:48 PM
Time Spent: 01:08:33
IP Address:

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Very Effective: good solution with a minimum number of people in minimum time frame.
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Very Positive
Q3: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Yes, Better
Q4: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Very Helpful, Additional comments on the project management activities: It is a model. The Paper Prototype concept would work with another model, looking abstractly. I have to admit my ignorance of some of the RM-ODP concepts, and that would make the Paper Prototype as proposed a learning curve to implement. Still, I responded that having a framework is good for the process because the process needs bones.

PAGE 3: Communication Evaluation

Q5: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective
Q6: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Very Effective

1 / 4

Q7: **What effect did the "1-pager" have on project iteration planning?**	Somewhat Effective, Additional comments on overall graphical representation use: It can have various effects on the planning process depending on the players and how they understand the 1-pager. It certainly improves planning, but can make the interactions in the meetings strained depending on how one views 1-page to equal simple.
Q8: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	Yes, Better
Q9: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Yes, Better
Q10: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	Yes, Better
Q11: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Very Helpful, Additional comments on business information unit modelling: Pictures are better for the business. And it is a bonus that the models make development easier.
Q12: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Very Useful, Additional comments on the business flow model: If you can get the buy-in from the business to model their flow.

PAGE 4: Requirements Quality

Q13: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Yes, Better
Q14: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better

Paper Prototype Developer Experiences SurveyMonkey

Q15: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Yes, Better, Additional comments on achieving requirements quality, correctness and completeness: I must say that if images/models made it bigger that would be acceptable as well. I don't like the question (15)

PAGE 5: Development and Testing Evaluation

Q16: **Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?**	Yes, Better
Q17: **Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?**	Yes, Better
Q18: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Yes, Better
Q19: **Do you feel that the visual components of the requirements specification had effects on developing tests?**	Yes, Better
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?**	Mostly Better, but with some limitations
Q21: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?**	Yes, Better
Q22: **If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?**	Yes, Better
Q23: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q24: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Yes, Better

3 / 4

Q25: **Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?**	Yes, Additional comments on the incremental & iterative process impacts: This concept (Paper Prototype) fits cleanly into agile.

PAGE 6: General Paper Prototype Experience

Q26: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	No
Q27: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	Yes, Please explain: A definition to defend.
Q28: **Experience in using the Paper Prototype process (time in months)?**	12 – 24 months
Q29: **Previous experience with software-based solution development projects?**	More than 4 projects
Q30: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Technical member of solution development team , Solution technology technical expert, Other (please specify) Contributing team member in the thinking of the "team"
Q31: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Easy: understood the concepts immediately, Any additional comments on the Paper Prototype process: Excepting the basis in RM-ODP. Still don't get that, but maybe someone will take me aside and explain.

Appendix H

Developer Response 2

Paper Prototype Developer Experiences SurveyMonkey

#2

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Thursday, June 08, 2017 3:17:41 PM
Last Modified: Thursday, June 08, 2017 3:32:44 PM
Time Spent: 00:15:02
IP Address:

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Very Effective: good solution with a minimum number of people in minimum time frame.
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Very Positive
Q3: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Yes, Better
Q4: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Very Helpful

PAGE 3: Communication Evaluation

Q5: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective
Q6: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Very Effective
Q7: **What effect did the "1-pager" have on project iteration planning?**	Very Effective
Q8: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	Yes, Better

1 / 4

Q9: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Yes, Better
Q10: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	*Respondent skipped this question*
Q11: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Helpful, but with some limitations
Q12: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Useful, but with some limitations

PAGE 4: Requirements Quality

Q13: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Yes, Better
Q14: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better
Q15: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Yes, Better

PAGE 5: Development and Testing Evaluation

Q16: **Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?**	Yes, Better
Q17: **Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?**	Yes, Better

Q18: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Yes, Better
Q19: **Do you feel that the visual components of the requirements specification had effects on developing tests?**	Yes, Better
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?**	Yes, Better
Q21: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?**	Yes, Better
Q22: **If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?**	Yes, Better
Q23: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q24: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Mostly Better, but with some limitations
Q25: **Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?**	Sometimes

PAGE 6: General Paper Prototype Experience

Q26: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	Yes, Please explain: The Paper Prototype process is very useful in nailing down many of the requirements, there is still the need for expressing some of the requirements in a traditional fashion (e.g., nonfunctional requirements).

Q27: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	Yes, Please explain: Any tool that can help the user "see" and provide feedback on the proposed solution with a minimal time/investment is helpful. The Paper Prototype process is such a tool that can be effective with certain types of development efforts.
Q28: **Experience in using the Paper Prototype process (time in months)?**	More than 24 months
Q29: **Previous experience with software-based solution development projects?**	More than 4 projects
Q30: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Project manager
Q31: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Medium: some concepts were quick to learn, while others took longer

Appendix I

Developer Response 3

Paper Prototype Developer Experiences SurveyMonkey

#3

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Wednesday, June 14, 2017 3:39:32 AM
Last Modified: Wednesday, June 14, 2017 4:01:40 AM
Time Spent: 00:22:07
IP Address: 77.106.169.166

PAGE 2: Overall Paper Prototype Evaluation

Q1: **What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.**	Moderately Effective: good solution with a reasonable number of people in a reasonable time frame. , Additional comments: The effectiveness doesn't necessary have to do with the paper prototype, but rather the number of people that we had to include in the process. We are changing the way we deploy the process in the second iteration by first having a small group and then taking it to the larger group once its stable. This way it becomes more manageable. We also doing reviews with smaller groups and rather more review meetings.
Q2: **How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?**	Very Positive
Q3: **How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?**	Yes, Better
Q4: **How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?**	Helpful, but with some limitations , Additional comments on the project management activities: It is hard to set aside enough time to go through the whole process in practice, so we only managed to do part of the process, meaning looking at some of the viewpoints but not all.

PAGE 3: Communication Evaluation

Q5: **What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?**	Very Effective
Q6: **What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?**	Somewhat Effective

1 / 4

Q7: **What effect did the "1-pager" have on project iteration planning?**	Somewhat Effective, Additional comments on overall graphical representation use: We found that its important that the person presenting the 1-pager have a good way of communicating it. If not, the discussion ends up focusing on what the 1-pager is and is showing rather than its content.
Q8: **What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?**	Yes, Better
Q9: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?**	Yes, Better
Q10: **Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?**	Yes, Better
Q11: **Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?**	Additional comments on business information unit modelling: We didn't successfully use the state machine models.
Q12: **Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?**	Useful, but with some limitations, Additional comments on the business flow model: The usefulness depends a bit on who one is communicating with. There are people that just like things to be the traditional way.

PAGE 4: Requirements Quality

Q13: **Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?**	Mostly Better, but with some limitations
Q14: **How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?**	Yes, Better
Q15: **How do you feel that the use of visual models for the requirements affected the size of the requirements specification?**	Yes, Better

Paper Prototype Developer Experiences — SurveyMonkey

PAGE 5: Development and Testing Evaluation

Q16: **Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?**	Yes, Better
Q17: **Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?**	Mostly Better, but with some limitations
Q18: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?**	Mostly Better, but with some limitations , Additional comments on solution design activities: As we didn´t have time to make use of all the views, we had some but not complete effect.
Q19: **Do you feel that the visual components of the requirements specification had effects on developing tests?**	Yes, Better
Q20: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?**	Mostly Better, but with some limitations
Q21: **Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?**	Mostly Better, but with some limitations , Additional comments on testing activities: As we didn´t manage to use all views, we didnt have a consistent process so we experienced various effect. Sometimes the discussion got diverted into what to capture and why, rather than using the information provided.
Q22: **If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?**	Yes, Better
Q23: **Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?**	Yes, Better
Q24: **What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?**	Mostly Better, but with some limitations
Q25: **Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?**	Yes

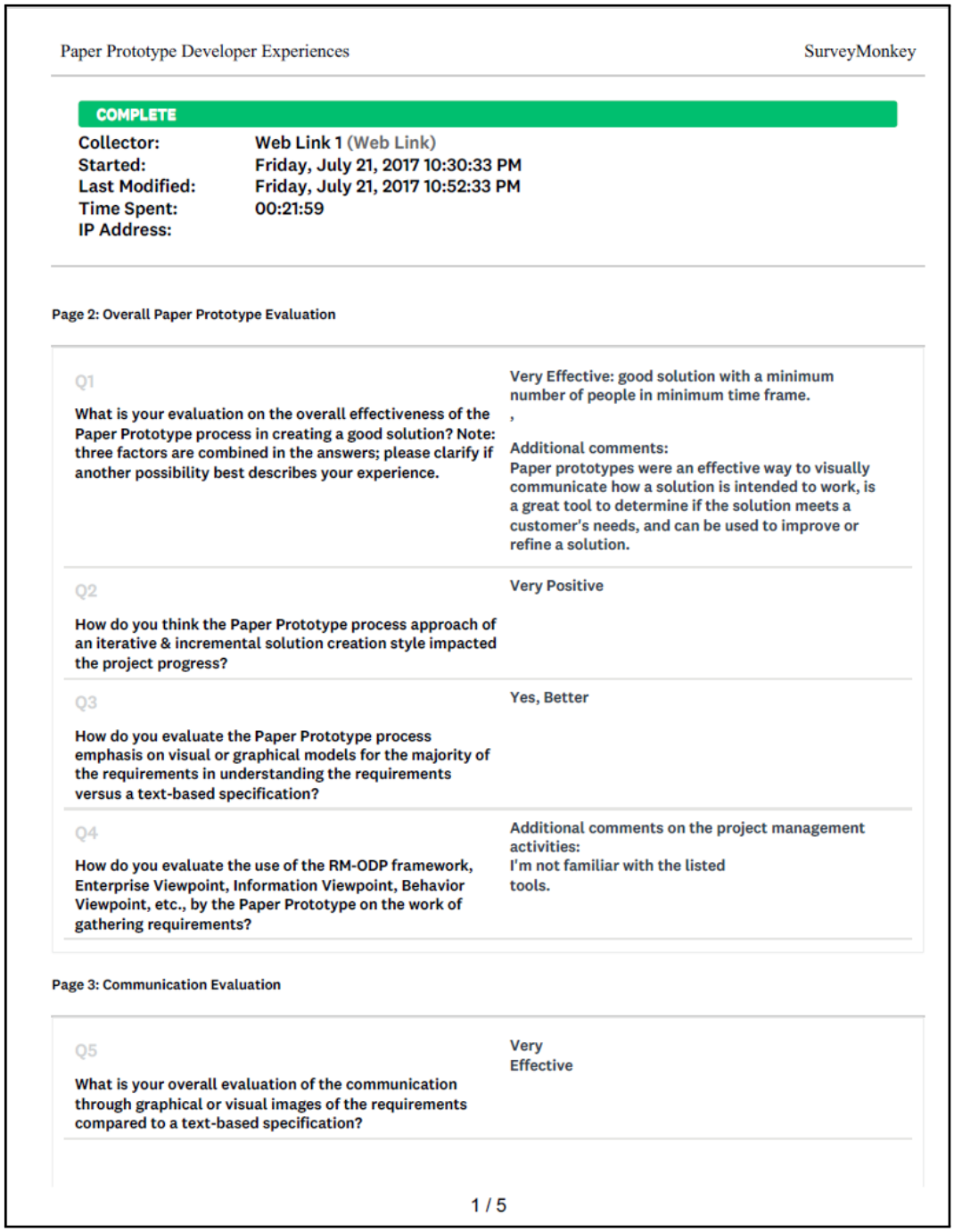
PAGE 6: General Paper Prototype Experience

3 / 4

Q26: **Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?**	Yes, Please explain: Modelling takes time and addressing the necessary RM-ODP views requires experience in modelling to be successful within the time available. Therefore, it might be useful to have a paper prototype tool to support the process.
Q27: **Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?**	Yes, Please explain: The paper prototype process forces you to think about the solution in a very concrete manner and help you visualize what you are building so that you can easier align between the team members to ensure they all agree on what it is that we are building.
Q28: **Experience in using the Paper Prototype process (time in months)?**	6 – 12 months
Q29: **Previous experience with software-based solution development projects?**	More than 4 projects
Q30: **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Solution technology technical expert, Project manager
Q31: **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Medium: some concepts were quick to learn, while others took longer , Any additional comments on the Paper Prototype process: Having experience with RM-ODP is essential to be effective in using the paper prototype process.

Appendix J

Developer Response 4



Paper Prototype Developer Experiences SurveyMonkey

COMPLETE

Collector: Web Link 1 (Web Link)
Started: Friday, July 21, 2017 10:30:33 PM
Last Modified: Friday, July 21, 2017 10:52:33 PM
Time Spent: 00:21:59
IP Address:

Page 2: Overall Paper Prototype Evaluation

Q1 What is your evaluation on the overall effectiveness of the Paper Prototype process in creating a good solution? Note: three factors are combined in the answers; please clarify if another possibility best describes your experience.	Very Effective: good solution with a minimum number of people in minimum time frame. , Additional comments: Paper prototypes were an effective way to visually communicate how a solution is intended to work, is a great tool to determine if the solution meets a customer's needs, and can be used to improve or refine a solution.
Q2 How do you think the Paper Prototype process approach of an iterative & incremental solution creation style impacted the project progress?	Very Positive
Q3 How do you evaluate the Paper Prototype process emphasis on visual or graphical models for the majority of the requirements in understanding the requirements versus a text-based specification?	Yes, Better
Q4 How do you evaluate the use of the RM-ODP framework, Enterprise Viewpoint, Information Viewpoint, Behavior Viewpoint, etc., by the Paper Prototype on the work of gathering requirements?	Additional comments on the project management activities: I'm not familiar with the listed tools.

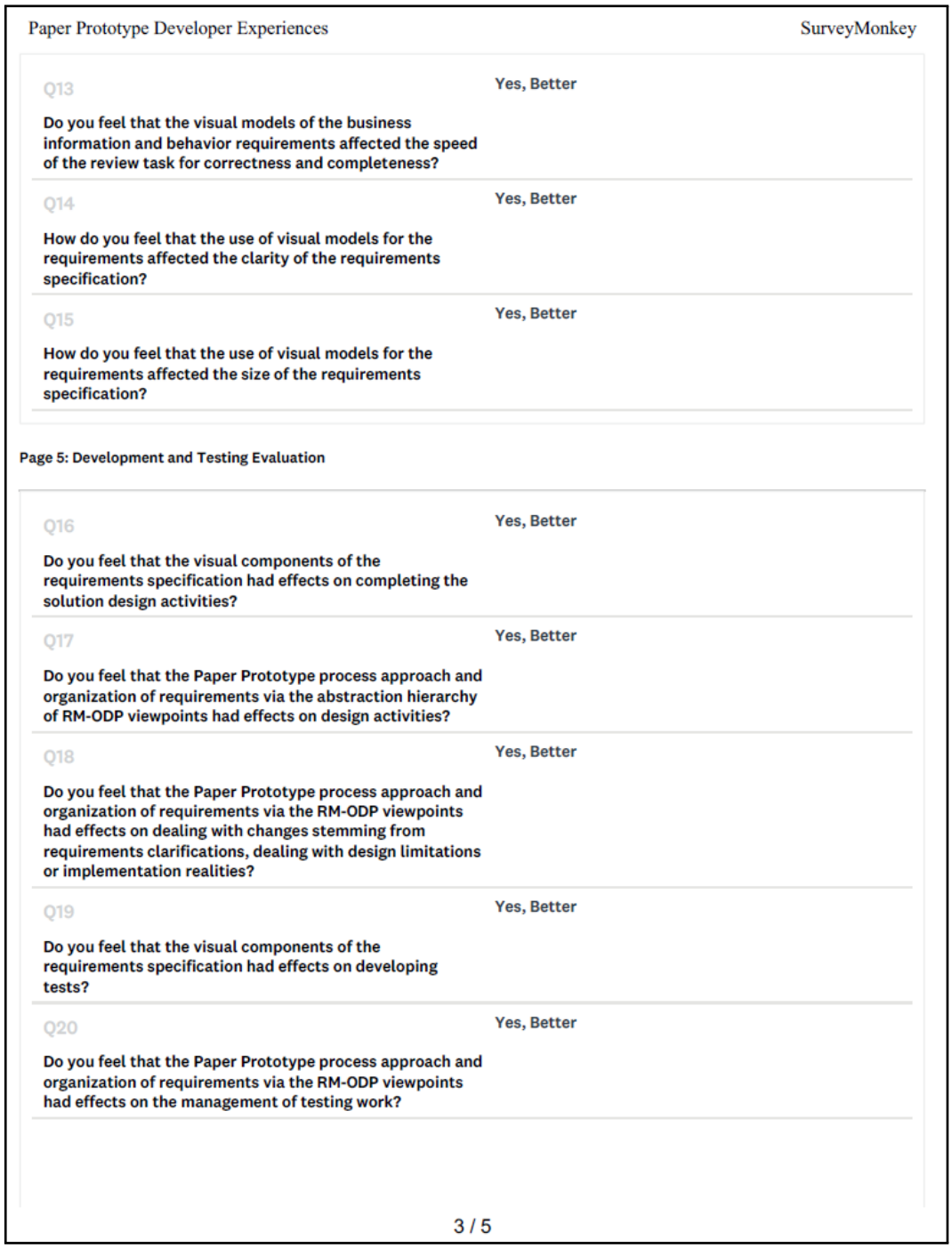
Page 3: Communication Evaluation

Q5 What is your overall evaluation of the communication through graphical or visual images of the requirements compared to a text-based specification?	Very Effective

1 / 5

Q6 What is your overall evaluation of the visual mechanism of the Paper Prototype process "1-pager" drawing in communicating the overall purpose of the solution?	Very Effective
Q7 What effect did the "1-pager" have on project iteration planning?	Very Effective
Q8 What is your opinion on the use of visual images for the business information units and their relationships or structure versus text-based descriptions of the solution business data on achieving an understanding of the specification?	Yes, Better
Q9 Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties (name, type, default value, description of business purpose, required, etc.) have an effect on communication and understanding?	Yes, Better
Q10 Did the 2-level combination of both [level-1] visual images for the business information units with a [level-2] text table for the detailed attribute properties have an effect on the process of gathering the information requirements through separating the identification of the major information units from the details of the units?	Yes, Better
Q11 Did the state machine models for the business information unit status values have an effect in understanding the behavior of the information units over time?	Very Helpful
Q12 Did the Business Flow behavior model in a single page drawing ("1-pager") have an effect on explaining to others the purpose of the solution and in obtaining useful improvement comments?	Very Useful

Page 4: Requirements Quality



Paper Prototype Developer Experiences SurveyMonkey

Q13 Do you feel that the visual models of the business information and behavior requirements affected the speed of the review task for correctness and completeness?	Yes, Better
Q14 How do you feel that the use of visual models for the requirements affected the clarity of the requirements specification?	Yes, Better
Q15 How do you feel that the use of visual models for the requirements affected the size of the requirements specification?	Yes, Better

Page 5: Development and Testing Evaluation

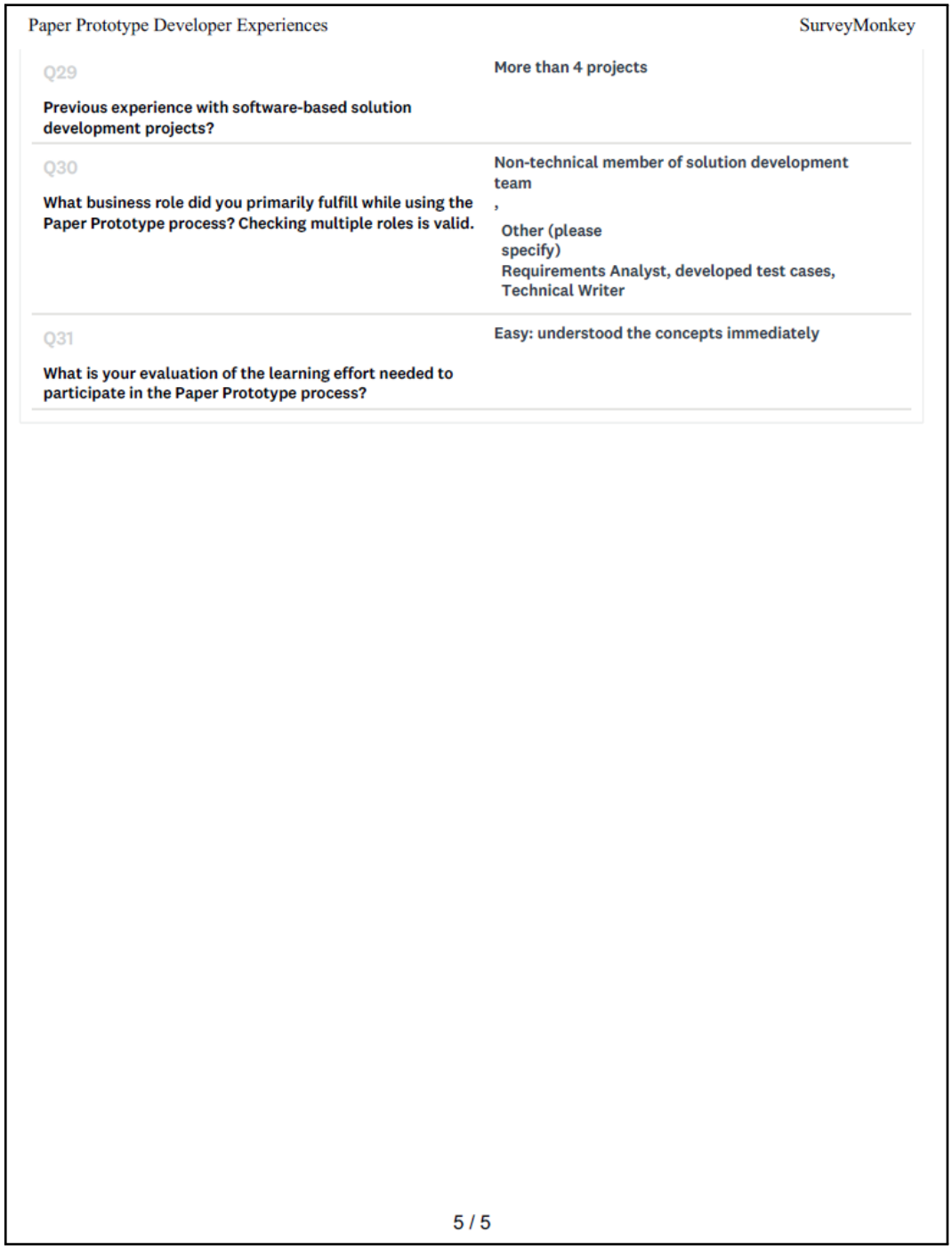
Q16 Do you feel that the visual components of the requirements specification had effects on completing the solution design activities?	Yes, Better
Q17 Do you feel that the Paper Prototype process approach and organization of requirements via the abstraction hierarchy of RM-ODP viewpoints had effects on design activities?	Yes, Better
Q18 Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on dealing with changes stemming from requirements clarifications, dealing with design limitations or implementation realities?	Yes, Better
Q19 Do you feel that the visual components of the requirements specification had effects on developing tests?	Yes, Better
Q20 Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of testing work?	Yes, Better

3 / 5

Q21 Do you feel that the Paper Prototype process approach and organization of requirements via the RM-ODP viewpoints had effects on the management of regression test work via clearer understanding of change impact?	Yes, Better
Q22 If you have had experience with other requirements gathering processes, how do you rate the incremental & iterative process approach (agile) used in the Paper Prototype for gathering requirements against the other process?	Yes, Better
Q23 Do you feel that the incremental & iterative process approach (agile) used in the Paper Prototype had an effect on the partitioning of the work of solution development?	Yes, Better
Q24 What effect do models have on the task of the partitioning of work for an incremental & iterative process (agile) and explaining why a particular partition plan was chosen?	Yes, Better
Q25 Does the incremental & iterative process (agile) for requirements gathering cleanly flow into agile design and implementation work?	Yes

Page 6: General Paper Prototype Experience

Q26 Relative to your experience with requirements specification review, did you find weaknesses or shortcomings with the Paper Prototype process?	Please explain: If the Paper Prototype was not clear, it left too much room for interpretation by the customer. This could lead to a solution that might not address the customer's expectations.
Q27 Relative to your experience with requirements specification review, did you find strengths or benefits with the Paper Prototype process?	Yes, Please explain: The Paper Prototype could be used for many purposes: developing test cases, creating a documentation plan, estimating tasks, developing regression test cases, and drafting procedures.
Q28 Experience in using the Paper Prototype process (time in months)?	12 – 24 months



Paper Prototype Developer Experiences SurveyMonkey

Q29 **Previous experience with software-based solution development projects?**	More than 4 projects
Q30 **What business role did you primarily fulfill while using the Paper Prototype process? Checking multiple roles is valid.**	Non-technical member of solution development team , Other (please specify) Requirements Analyst, developed test cases, Technical Writer
Q31 **What is your evaluation of the learning effort needed to participate in the Paper Prototype process?**	Easy: understood the concepts immediately

5 / 5

Appendix K

Curriculum Vitae

Name	Matheson
Given Names	Dan McKay
Birth Date	26. Sep 1950
Birth Place	Des Moines, Iowa
Nationality	USA

since 2014 Retired

2004 - 2014 Senior Architect, Integware, Inc.

- Architected and delivered 5 PLM business solutions based on the strategy for Edwards Lifesciences.
- Architected and delivered 13 PLM business solutions based on the strategy for J&J Vision Care.
- Conducted and delivered PLM strategy assessments for J&J Vision Care.
- MatrixOne implementations (10.5, 10.6): integration of external applications (AdLib, Myriad); developed JPOs and JSPs for custom Bill of Materials display; development on DHF, NCR and CAPA applications.

1991 - 2004 R&D Engineer and Architect, Hewlett-Packard

- Delivered consulting service (requirements through solution delivery) on WorkManager PLM for many companies, including TRW, GM, HP, Proton, Siemens, Samsung and Boeing.
- Delivered consulting (requirements through solution delivery) on WorkManager workflow for TRW, GM, EDS, HP, Samsung and Proton. Samsung cut new product development time to one third of pre-workflow and pre-PDM times.
- Member of the Software Engineering Center of Excellence team at the HP OpenView division where I delivered training and consulting to worldwide project teams in requirements management processes (Volere), solution modeling techniques (RM-ODP & UML), agile development methodologies, and enterprise and software patterns.
- Developer and lead engineer on WorkManager PDM, with concentration on MCAD integration, MCAD data management and business process automation (workflow).

• Designed data models, constructed XML schemas and coded Java on the OpenView Interconnect integration product, the next generation product for integrating OpenView into a total customer solution.
• Contributed to the further development of TeleManagement Forum standards in the areas of Process Management and Standard Data Model, which are used for telecom interoperability.
• Designed and planned the follow-on products to the Node Sentry IDS security product. Created a roadmap for a line of products for the management of security events, guided the technical decisions of the team as we worked with partners.

• Contributed to the OMG Product Data Management Enablers specification and was a member of the PDM Enablers RTF.
• Chair of the Workflow Management Coalition Working Group 5, Administration and Auditing: I brought to completion version 1.0, WfMC-TC-1015. WfMC Fellow.
• Leader of Workflow Management Facility Specification, (V1.0 - V1.2) in OMG.
• My preliminary work on a Process Definition RFP led to a complete revision of the state machine based activity modeling ideas of UML 1.2 into a directed cyclic graph approach for UML 2.0.
• Patent: Cad Data Model With Design Notes - 6,718,218
• Patent: Collaboration Session Recording Model - 6,952,660 & 7,184,940
• Patent: Innovation Information Management Model - 6,944,514 & 7,050,872
• Patent: Object Model for Decision and Issue Tracking - 7,574,329

1979 - 1991 R&D Engineer and Architect, Hewlett-Packard GmbH
• Developed several mechanical 2D & 3D CAD products, the original designer of ME10.
• Developer and architect on several OpenView network and system management products, role of security architect for all HP OpenView products.
• Developer and lead engineer for HP-UX tape and backup commands and development in the file system portion of the HP-UX kernel.
• Lead engineer on database portion of business software suite for HP-UX.
• Established the first TCP/IP network services for the Böblingen site. I managed the primary servers, developed material for and taught several European system administration courses, and authored papers for 2 HP internal UNIX system administration courses.
• Prepared and presented with a colleague several day-long internal seminars on software engineering methods. The topics included inspections, structured design, structured analysis and objects.

1978	Bachelor of Science Physics, Colorado State University
1981	Master of Science Computer Science, Colorado State University

List of Figures

1.1 SAMEM Components 4

2.1 RUP Phases and Workflows Organization 12
2.2 RUP 4+1 Architecture Model 14
2.3 REAM Example 20
2.4 Use Case Scenario Example 20
2.5 Fernandes Process Example 23
2.6 V-Model Typical Image 25
2.7 ESSENCE Four Layer Architecture 27
2.8 URN Goal Model for HL-GOAL-1 39
2.9 URN Goal Model for HL-GOAL-2 40
2.10 URN Goal Model for HL-GOAL-3 41
2.11 URN Goal Model for HL-GOAL-4 42
2.12 URN Goal Model for HL-GOAL-5 43
2.13 URN Goal Model for HL-GOAL-6 44

3.1 Solution Concept Example 53
3.2 SAMEM Components 63
3.3 RM-ODP Viewpoint Relationships 68
3.4 Design History Puzzle Piece Refinement 71
3.5 SOD Standard Layout Overview Example 73
3.6 SOD Information Model Section Detail 74
3.7 SOD State Machine Section Detail 75
3.8 SOD Notes Section Detail 75
3.9 SOD Business Flow Section Detail 77
3.10 Information Model Example of Basic Information Units 80
3.11 Information Model Example of Attribute Table 81
3.12 Information Model for a Single Specification 83
3.13 Information Model Example of a Simple Project Structure 84
3.14 Information Model Example of a Complex Structure 85
3.15 Report Example 87
3.16 Business Flow Behavior Example 91
3.17 Business Flow Task Connection to Use Case Example 93
3.18 Approval Participants Table Example 93
3.19 Approval Flow Example 95
3.20 ATM for the Regulatory Solution Example 96
3.21 ATM Reference Specification Example 97

3.22 ATM Reference Specification Example 97
3.23 Design Rationale Example 99
3.24 Design Class Model Example 100
3.25 Design Rationale for Requirement Realization 101
3.26 BMIDE Object Model Implementation Example 102
3.27 Detailed Design Approval Workflow Example 103
3.28 Detailed Design Standard Search Example 104
3.29 Detailed Design Organization Roles & Groups Example 104
3.30 Emerging Security Concepts Example 111
3.31 Emerging Domain Concepts from Authorization 112
3.32 Emerging Domain Concepts from Identification 112

4.1 Project Management Process Phases 135
4.2 Project Status Example Mapped to RM-ODP Model 138
4.3 Enterprise Viewpoint Solution Concept Model 139
4.4 Sub-solution Concept Refinement Model 141
4.5 Partial Example of an ATM 144
4.6 A Stakeholder-developed Image of the Project Process 150
4.7 Stakeholder Image of Company R&D Environment 150

5.1 The SAMEM Process Concepts 166
5.2 Process Concepts Realized as Abstract Objects 168
5.3 Process Instance States 169
5.4 The SAMEM Process Model of Concepts in UML 171
5.5 Methodology - Viewpoint Instance Relationship Example 172
5.6 Phase Goal Explanations Example 173
5.7 Viewpoint, Iteration Instance Example 174
5.8 The SAMEM Project Process Example 176

6.1 The SAMEM-IM Concepts 180
6.2 The SAMEM-IM Example, SOD Composition 182
6.3 OMG Diagram Definition Architecture for UML 186
6.4 DI Model from the OMG DD Specification 187
6.5 Merging of UML DD and SAMEM-IM Drawing Concepts 187
6.6 UML Class Operation to Behavior Link Concept 189
6.7 Reproduction of UML 2.5 Figure 9.1 for Reference 190
6.8 Reproduction of UML 2.5 Figure 9.9 for Reference 190
6.9 Reproduction of UML 2.5 Figure 9.13 for Reference 191
6.10 Composite of UML Behavior Subclasses 192
6.11 Reproduction of UML 2.5 Figure 7.17 for Reference 192
6.12 Operation Behavior Extension to UML 193
6.13 Behavior Models Extension 194

7.1 Reproduction of Figure 2.8 218

7.2 Reproduction of Figure 2.9 220
7.3 Reproduction of Figure 2.10 222
7.4 : Reproduction of Figure 2.11 225
7.5 Reproduction of Figure 2.12 229
7.6 Reproduction of Figure 2.13 232
7.7 Essence Kernel Alphas 247
7.8 Alpha Artifact Overview 248
7.9 Way of Working Alpha Overview and State Cards Example 249
7.10 Way of Working Full Checklist Example 250
7.11 SEMAT Competencies Overview 251
7.12 SEMAT Generic Competency Levels Table 252
7.13 SEMAT Leadership Competency Card Example 253
7.14 SEMAT Leadership Competency Goals and Skills Example 253
7.15 Activity Spaces and Concerns Overview 254
7.16 Alphas, Activity Spaces, and Competencies Relationship 254
7.17 High Level SAMEM - SEMAT Concept Correspondence 256
7.18 SEMAT Software System Architecture Selected State Checklist 257

List of Tables

1.1 Classification Marks 7

2.1 Volere Example Questions 17

4.1 CS-1 Sub-Solutions 118
4.2 CS-1 Solution Size Measures 120
4.3 CS-1 Participants, Roles & Numbers 121
4.4 CS-2 Workstreams 122
4.5 CS-2 Workstream Size Measures 123
4.6 CS-2 Participants, Roles & Numbers 124
4.7 Lessons Learned Summary 156

7.1 Common Survey Questions 200
7.2 Unique Survey Questions 203
7.3 Common Questions Answer Merge 204
7.4 Unique Survey Questions Response Values 209
7.5 Respondent Experience 212
7.6 ***HL-GOAL-1*** Supporting Survey Quotes 215
7.7 ***HL-GOAL-2*** Supporting Survey Quotes 219
7.8 ***HL-GOAL-3*** Supporting Survey Quotes 221
7.9 ***HL-GOAL-4*** Supporting Survey Quotes 223
7.10 ***HL-GOAL-5*** Supporting Survey Quotes 227
7.11 ***HL-GOAL-6*** Supporting Survey Quotes 230
7.12 SAMEM Project Phases and SEMAT Alpha States Mapping 258

Related Interesting Work from the SE Group, RWTH Aachen

Agile Model Based Software Engineering

Agility and modeling in the same project? This question was raised in [Rum04]: "Using an executable, yet abstract and multi-view modeling language for modeling, designing and programming still allows to use an agile development process." Modeling will be used in development projects much more, if the benefits become evident early, e.g with executable UML [Rum02] and tests [Rum03]. In [GKRS06], for example, we concentrate on the integration of models and ordinary programming code. In [Rum12] and [Rum16], the UML/P, a variant of the UML especially designed for programming, refactoring and evolution, is defined. The language workbench MontiCore [GKR+06, GKR+08] is used to realize the UML/P [Sch12]. Links to further research, e.g., include a general discussion of how to manage and evolve models [LRSS10], a precise definition for model composition as well as model languages [HKR+09] and refactoring in various modeling and programming languages [PR03]. In [FHR08] we describe a set of general requirements for model quality. Finally [KRV06] discusses the additional roles and activities necessary in a DSL-based software development project. In [CEG+14] we discuss how to improve reliability of adaptivity through models at runtime, which will allow developers to delay design decisions to runtime adaptation.

Generative Software Engineering

The UML/P language family [Rum12, Rum11, Rum16] is a simplified and semantically sound derivative of the UML designed for product and test code generation. [Sch12] describes a flexible generator for the UML/P based on the MontiCore language workbench [KRV10, GKR+06, GKR+08]. In [KRV06], we discuss additional roles necessary in a model-based software development project. In [GKRS06] we discuss mechanisms to keep generated and handwritten code separated. In [Wei12] demonstrate how to systematically derive a transformation language in concrete syntax. To understand the implications of executability for UML, we discuss needs and advantages of executable modeling with UML in agile projects in [Rum04], how to apply UML for testing in [Rum03] and the advantages and perils of using modeling languages for programming in [Rum02].

Unified Modeling Language (UML)

Starting with an early identification of challenges for the standardization of the UML in [KER99] many of our contributions build on the UML/P variant, which is described in the two books [Rum16] and [Rum12] implemented in [Sch12]. Semantic variation points of the UML are discussed in [GR11]. We discuss formal semantics for UML [BHP+98] and describe UML semantics using the "System Model" [BCGR09a], [BCGR09b], [BCR07b] and [BCR07a]. Semantic variation points have, e.g., been applied to define class diagram semantics [CGR08]. A precisely defined semantics for variations is applied, when checking variants of class diagrams [MRR11c] and objects diagrams [MRR11d] or the consistency of both kinds of diagrams [MRR11e]. We also apply these concepts to activity diagrams [MRR11b] which allows us to check for semantic differences of activity diagrams [MRR11a]. The basic semantics for ADs and their semantic variation points is given in [GRR10]. We also discuss how to ensure and identify model quality [FHR08], how models, views and the system under development correlate to each other [BGH+98] and how to use modeling in agile development projects [Rum04], [Rum02]. The question how to adapt and extend the UML is discussed in [PFR02] describing product line annotations for UML and more general discussions and insights on how to use meta-modeling for defining and adapting the UML are included in [EFLR99], [FELR98] and [SRVK10].

Domain Specific Languages (DSLs)

Computer science is about languages. Domain Specific Languages (DSLs) are better to use, but need appropriate tooling. The MontiCore language workbench [GKR+06, KRV10, Kra10, GKR+08] allows the specification of an integrated abstract and concrete syntax format [KRV07b] for easy development. New languages and tools can be defined in modular forms [KRV08, GKR+07, Völ11] and can, thus, easily be reused. [Wei12] presents a tool that allows to create transformation rules tailored to an underlying DSL. Variability in DSL definitions has been examined in [GR11]. A successful application has been carried out in the Air Traffic Management domain [ZPK+11]. Based on the concepts described above, meta modeling, model analyses and model evolution have been discussed in [LRSS10] and [SRVK10]. DSL quality [FHR08], instructions for defining views [GHK+07], guidelines to define DSLs [KKP+09] and Eclipse-based tooling for DSLs [KRV07a] complete the collection.

Software Language Engineering

For a systematic definition of languages using composition of reusable and adaptable language components, we adopt an engineering viewpoint on these techniques. General ideas on how to engineer a language can be found in the GeMoC initiative [CBCR15, CCF+15]. As said, the MontiCore language workbench provides techniques for an integrated definition of languages [KRV07b, Kra10, KRV10]. In [SRVK10] we discuss the possibilities and the challenges using metamodels for language definition. Modular composition, however, is a core concept to reuse language components like in MontiCore for the frontend [Völ11, KRV08] and the backend [RRRW15]]. Language derivation is to our believe a promising technique to develop new languages for a specific purpose that rely on existing basic languages. How to automatically derive such a transformation language using concrete syntax of the base language is described in [HRW15, Wei12] and successfully applied to various DSLs. We also applied the language derivation technique to tagging languages that decorate a base language [GLRR15] and delta languages [HHK+15a, HHK+13], where a delta language is derived from a base language to be able to constructively describe differences between model variants usable to build feature sets.

Modeling Software Architecture & the MontiArc Tool

Distributed interactive systems communicate via messages on a bus, discrete event signals, streams of telephone or video data, method invocation, or data structures passed between software services. We use streams, statemachines and components [BR07] as well as expressive forms of composition and refinement [PR99] for semantics. Furthermore, we built a concrete tooling infrastructure called MontiArc [HRR12] for architecture design and extensions for states [RRW13b]. MontiArc was extended to describe variability [HRR+11] using deltas [HRRS11, **?**] and evolution on deltas [HRRS12]. [GHK+07] and [GHK+08] close the gap between the requirements and the logical architecture and [GKPR08] extends it to model variants. [MRR14] provides a precise technique to verify consistency of architectural views [Rin14, MRR13] against a complete architecture in order to increase reusability. Co-evolution of architecture is discussed in [MMR10] and a modeling technique to describe dynamic architectures is shown in [HRR98].

Compositionality & Modularity of Models

[HKR+09] motivates the basic mechanisms for modularity and compositionality for modeling. The mechanisms for distributed systems are shown in [BR07] and algebraically underpinned in [HKR+07]. Semantic and methodical aspects of model composition [KRV08] led to the language workbench MontiCore [KRV10] that can even be used to develop modeling tools in a compositional form. A set of DSL design guidelines incorporates reuse through this form of composition [KKP+09]. [Völ11] examines the composition of context conditions respectively the underlying infrastructure of the symbol table. Modular editor generation is discussed in [KRV07a]. [RRRW15] applies compositionality to Robotics control. [CBCR15] (published in [CCF+15]) summarizes our approach to composition and remaining challenges in form of a conceptual model of the "globalized" use of DSLs. As a new form of decomposition of model information we have developed the concept of tagging languages in [GLRR15]. It allows to describe additional information for model elements in separated documents, facilitates reuse, and allows to type tags.

Semantics of Modeling Languages

The meaning of semantics and its principles like underspecification, language precision and detailedness is discussed in [HR04]. We defined a semantic domain called "System Model" by using mathematical theory in [RKB95, BHP+98] and [GKR96, KRB96]. An extended version especially suited for the UML is given in [BCGR09b] and in [BCGR09a] its rationale is discussed. [BCR07a, BCR07b] contain detailed versions that are applied to class diagrams in [CGR08]. To better understand the effect of an evolved design, detection of semantic differencing as opposed to pure syntactical differences is needed [MRR10]. [MRR11a, MRR11b] encode a part of the semantics to handle semantic differences of activity diagrams and [MRR11e] compares class and object diagrams with regard to their semantics. In [BR07], a simplified mathematical model for distributed systems based on black-box behaviors of components is defined. Meta-modeling semantics is discussed in [EFLR99]. [BGH+97] discusses potential modeling languages for the description of an exemplary object interaction, today called sequence diagram. [BGH+98] discusses the relationships between a system, a view and a complete model in the context of the UML. [GR11] and [CGR09] discuss general requirements for a framework to describe semantic and syntactic variations of a modeling language. We apply these on class and object diagrams in [MRR11e] as well as activity diagrams in [GRR10]. [Rum12] defines the semantics in a variety of code and test case generation, refactoring and evolution techniques. [LRSS10] discusses evolution and related issues in greater detail.

Evolution & Transformation of Models

Models are the central artifact in model driven development, but as code they are not initially correct and need to be changed, evolved and maintained over time. Model transformation is therefore essential to effectively deal with models. Many concrete model transformation problems are discussed: evolution [LRSS10, MMR10, Rum04], refinement [PR99, KPR97, PR94], refactoring [Rum12, PR03], translating models from one language into another [MRR11c, Rum12] and systematic model transformation language development [Wei12]. [Rum04] describes how comprehensible sets of such transformations support software development and maintenance [LRSS10], technologies for evolving models within a language and across languages, and mapping architecture descriptions to their implementation [MMR10]. Automaton refinement is discussed in [PR94, KPR97], refining pipe-and-filter architectures is explained in [PR99]. Refactorings of models are important for model driven engineering as discussed in [PR01, PR03, Rum12]. Translation between languages, e.g., from class diagrams into Alloy [MRR11c] allows for comparing class diagrams on a semantic level.

Variability & Software Product Lines (SPL)

Products often exist in various variants, for example cars or mobile phones, where one manufacturer develops several products with many similarities but also many variations. Variants are managed in a Software Product Line (SPL) that captures product commonalities as well as differences. Feature diagrams describe variability in a top down fashion, e.g., in the automotive domain [GHK+08] using 150% models. Reducing overhead and associated costs is discussed in [GRJA12]. Delta modeling is a bottom up technique starting with a small, but complete base variant. Features are additive, but also can modify the core. A set of commonly applicable deltas configures a system variant. We discuss the application of this technique to Delta-MontiArc [HRR+11, HRR+11] and to Delta-Simulink [HKM+13]. Deltas can not only describe spacial variability but also temporal variability which allows for using them for software product line evolution [HRRS12]. [HHK+13] and [HRW15] describe an approach to systematically derive delta languages. We also apply variability to modeling languages in order to describe syntactic and semantic variation points, e.g., in UML for frameworks [PFR02]. Furthermore, we specified a systematic way to define variants of modeling languages [CGR09] and applied this as a semantic language refinement on Statecharts in [GR11].

Cyber-Physical Systems (CPS)

Cyber-Physical Systems (CPS) [KRS12] are complex, distributed systems which control physical entities. Contributions for individual aspects range from requirements [GRJA12], complete product lines [HRRW12], the improvement of engineering for distributed automotive systems [HRR12] and autonomous driving [BR12a] to processes and tools to improve the development as well as the product itself [BBR07]. In the aviation domain, a modeling language for uncertainty and safety events was developed, which is of interest for the European airspace [ZPK+11]. A component and connector architecture description language suitable for the specific challenges in robotics is discussed in [RRW13b, RRW14]. Monitoring for smart and energy efficient buildings is developed as Energy Navigator toolset [KPR12, FPPR12, KLPR12].

State Based Modeling (Automata)

Today, many computer science theories are based on statemachines in various forms including Petri nets or temporal logics. Software engineering is particularly interested in using statemachines for modeling systems. Our contributions to state based modeling can currently be split into three parts: (1) understanding how to model object-oriented and distributed software using statemachines resp. Statecharts [GKR96, BCR07b, BCGR09b, BCGR09a], (2) understanding the refinement [PR94, RK96, Rum96] and composition [GR95] of statemachines, and (3) applying statemachines for modeling systems. In [Rum96] constructive transformation rules for refining automata behavior are given and proven correct. This theory is applied to features in [KPR97]. Statemachines are embedded in the composition and behavioral specification concepts of Focus [BR07]. We apply these techniques, e.g., in MontiArcAutomaton [RRW13a, RRW14] as well as in building management systems [FLP+11].

Robotics

Robotics can be considered a special field within Cyber-Physical Systems which is defined by an inherent heterogeneity of involved domains, relevant platforms, and challenges. The engineering of robotics applications requires composition and interaction of diverse distributed software modules. This usually leads to complex monolithic software solutions hardly reusable, maintainable, and comprehensible, which hampers broad propagation of robotics applications. The MontiArcAutomaton language [RRW13a] extends ADL MontiArc and integrates various implemented behavior modeling languages using MontiCore [RRW13b, RRW14, RRRW15] that perfectly fit Robotic architectural modeling. The LightRocks [THR+13] framework allows robotics experts and laymen to model robotic assembly tasks.

Automotive, Autonomic Driving & Intelligent Driver Assistance

Introducing and connecting sophisticated driver assistance, infotainment and communication systems as well as advanced active and passive safety-systems result in complex embedded systems. As these feature-driven subsystems may be arbitrarily combined by the customer, a huge amount of distinct variants needs to be managed, developed and tested. A consistent requirements management that connects requirements with features in all phases of the development for the automotive domain is described in [GRJA12]. The conceptual gap between requirements and the logical architecture of a car is closed in [GHK+07, GHK+08]. [HKM+13] describes a tool for delta modeling for Simulink [HKM+13]. [HRRW12] discusses means to extract a well-defined Software Product Line from a set of copy and paste variants. [RSW+15] describes an approach to use model checking techniques to identify behavioral differences of Simulink models. Quality assurance, especially of safety-related functions, is a highly important task. In the Carolo project [BR12a, BR12b], we developed a rigorous test infrastructure for intelligent, sensor-based functions through fully-automatic simulation [BBR07]. This technique allows a dramatic speedup in development and evolution of autonomous car functionality, and thus enables us to develop software in an agile way [BR12a]. [MMR10] gives an overview of the current state-of-the-art in development and evolution on a more general level by considering any kind of critical system that relies on architectural descriptions. As tooling infrastructure, the SSELab storage, versioning and management services [HKR12] are essential for many projects.

Energy Management

In the past years, it became more and more evident that saving energy and reducing CO2 emissions is an important challenge. Thus, energy management in buildings as well as in neighborhoods becomes equally important to efficiently use the generated energy. Within several research projects, we developed methodologies and solutions for integrating heterogeneous systems at different scales. During the design phase, the Energy Navigators Active Functional Specification (AFS) [FPPR12, KPR12] is used for technical specification of building services already. We adapted the well-known concept of statemachines to be able to describe different states of a facility and to validate it against the monitored values [FLP+11]. We show how our data model, the constraint rules and the evaluation approach to compare sensor data can be applied [KLPR12].

Cloud Computing & Enterprise Information Systems

The paradigm of Cloud Computing is arising out of a convergence of existing technologies for web-based application and service architectures with high complexity, criticality and new application domains. It promises to enable new business models, to lower the barrier for web-based innovations and to increase the efficiency and cost-effectiveness of web development [KRR14]. Application classes like Cyber-Physical Systems and their privacy [HHK+14, HHK+15b], Big Data, App and Service Ecosystems bring attention to aspects like responsiveness, privacy and open platforms. Regardless of the application domain, developers of such systems are in need for robust methods and efficient, easy-to-use languages and tools [KRS12]. We tackle these challenges by perusing a model-based, generative approach [NPR13]. The core of this approach are different modeling languages that describe different aspects of a cloud-based system in a concise and technology-agnostic way. Software architecture and infrastructure models describe the system and its physical distribution on a large scale. We apply cloud technology for the services we develop, e.g., the SSELab [HKR12] and the Energy Navigator [FPPR12, KPR12] but also for our tool demonstrators and our own development platforms. New services, e.g., collecting data from temperature, cars etc. can now easily be developed.

[BBR07] Christian Basarke, Christian Berger, and Bernhard Rumpe. Software & Systems Engineering Process and Tools for the Development of Autonomous Driving Intelligence. *Journal of Aerospace Computing, Information, and Communication (JACIC)*, 4(12):1158–1174, 2007.

[BCGR09a] Manfred Broy, María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Considerations and Rationale for a UML System Model. In K. Lano, editor, *UML 2 Semantics and Applications*, pages 43–61. John Wiley & Sons, November 2009.

[BCGR09b] Manfred Broy, María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Definition of the UML System Model. In K. Lano, editor, *UML 2 Semantics and Applications*, pages 63–93. John Wiley & Sons, November 2009.

[BCR07a] Manfred Broy, María Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 2: The Control Model. Technical Report TUM-I0710, TU Munich, Germany, February 2007.

[BCR07b] Manfred Broy, María Victoria Cengarle, and Bernhard Rumpe. Towards a System Model for UML. Part 3: The State Machine Model. Technical Report TUM-I0711, TU Munich, Germany, February 2007.

[BGH+97] Ruth Breu, Radu Grosu, Christoph Hofmann, Franz Huber, Ingolf Krüger, Bernhard Rumpe, Monika Schmidt, and Wolfgang Schwerin. Exemplary and Complete Object Interaction Descriptions. In *Object-oriented Behavioral Semantics Workshop (OOPSLA'97)*, Technical Report TUM-I9737, Germany, 1997. TU Munich.

[BGH+98] Ruth Breu, Radu Grosu, Franz Huber, Bernhard Rumpe, and Wolfgang Schwerin. Systems, Views and Models of UML. In *Proceedings of the Unified Modeling Language, Technical Aspects and Applications*, pages 93–109. Physica Verlag, Heidelberg, Germany, 1998.

[BHP+98] Manfred Broy, Franz Huber, Barbara Paech, Bernhard Rumpe, and Katharina Spies. Software and System Modeling Based on a Unified Formal Semantics. In *Workshop on Requirements Targeting Software and Systems Engineering (RTSE'97)*, LNCS 1526, pages 43–68. Springer, 1998.

[BR07] Manfred Broy and Bernhard Rumpe. Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. *Informatik-Spektrum*, 30(1):3–18, Februar 2007.

[BR12a] Christian Berger and Bernhard Rumpe. Autonomous Driving - 5 Years after the Urban Challenge: The Anticipatory Vehicle as a Cyber-Physical System. In *Automotive Software Engineering Workshop (ASE'12)*, pages 789–798, 2012.

[BR12b] Christian Berger and Bernhard Rumpe. Engineering Autonomous Driving Software. In C. Rouff and M. Hinchey, editors, *Experience from the DARPA Urban Challenge*, pages 243–271. Springer, Germany, 2012.

[CBCR15] Tony Clark, Mark van den Brand, Benoit Combemale, and Bernhard Rumpe. Conceptual Model of the Globalization for Domain-Specific Languages. In *Globalizing Domain-Specific Languages*, LNCS 9400, pages 7–20. Springer, 2015.

[CCF+15] Betty H. C. Cheng, Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, and Bernhard Rumpe, editors. *Globalizing Domain-Specific Languages*, LNCS 9400. Springer, 2015.

[CEG+14] Betty Cheng, Kerstin Eder, Martin Gogolla, Lars Grunske, Marin Litoiu, Hausi Müller, Patrizio Pelliccione, Anna Perini, Nauman Qureshi, Bernhard Rumpe, Daniel Schneider, Frank Trollmann, and Norha Villegas. Using Models at Runtime to Address Assurance for Self-Adaptive Systems. In *Models@run.time*, LNCS 8378, pages 101–136. Springer, Germany, 2014.

[CGR08] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. System Model Semantics of Class Diagrams. Informatik-Bericht 2008-05, TU Braunschweig, Germany, 2008.

[CGR09] María Victoria Cengarle, Hans Grönniger, and Bernhard Rumpe. Variability within Modeling Language Definitions. In *Conference on Model Driven Engineering Languages and Systems (MODELS'09)*, LNCS 5795, pages 670–684. Springer, 2009.

[EFLR99] Andy Evans, Robert France, Kevin Lano, and Bernhard Rumpe. Meta-Modelling Semantics of UML. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*, pages 45–60. Kluver Academic Publisher, 1999.

[FELR98] Robert France, Andy Evans, Kevin Lano, and Bernhard Rumpe. The UML as a formal modeling notation. *Computer Standards & Interfaces*, 19(7):325–334, November 1998.

[FHR08] Florian Fieber, Michaela Huhn, and Bernhard Rumpe. Modellqualität als Indikator für Softwarequalität: eine Taxonomie. *Informatik-Spektrum*, 31(5):408–424, Oktober 2008.

[FLP+11] M. Norbert Fisch, Markus Look, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. State-based Modeling of Buildings and Facilities. In *Enhanced Building Operations Conference (ICEBO'11)*, 2011.

[FPPR12] M. Norbert Fisch, Claas Pinkernell, Stefan Plesser, and Bernhard Rumpe. The Energy Navigator - A Web-Platform for Performance Design and Management. In *Energy Efficiency in Commercial Buildings Conference(IEECB'12)*, 2012.

[GHK+07] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, and Bernhard Rumpe. View-based Modeling of Function Nets. In *Object-oriented Modelling of Embedded Real-Time Systems Workshop (OMER4'07)*, 2007.

[GHK+08] Hans Grönniger, Jochen Hartmann, Holger Krahn, Stefan Kriebel, Lutz Rothhardt, and Bernhard Rumpe. Modelling Automotive Function Nets with Views for Features, Variants, and Modes. In *Proceedings of 4th European Congress ERTS - Embedded Real Time Software*, 2008.

[GKPR08] Hans Grönniger, Holger Krahn, Claas Pinkernell, and Bernhard Rumpe. Modeling Variants of Automotive Systems using Views. In *Modellbasierte Entwicklung von eingebetteten Fahrzeugfunktionen*, Informatik Bericht 2008-01, pages 76–89. TU Braunschweig, 2008.

[GKR96] Radu Grosu, Cornel Klein, and Bernhard Rumpe. Enhancing the SysLab System Model with State. Technical Report TUM-I9631, TU Munich, Germany, July 1996.

[GKR+06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. MontiCore 1.0 - Ein Framework zur Erstellung und Verarbeitung domänspezifischer Sprachen. Informatik-Bericht 2006-04, CFG-Fakultät, TU Braunschweig, August 2006.

[GKR+07] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Textbased Modeling. In *4th International Workshop on Software Language Engineering, Nashville*, Informatik-Bericht 4/2007. Johannes-Gutenberg-Universität Mainz, 2007.

[GKR+08] Hans Grönniger, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. MontiCore: A Framework for the Development of Textual Domain Specific Languages. In *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008, Companion Volume*, pages 925–926, 2008.

[GKRS06] Hans Grönniger, Holger Krahn, Bernhard Rumpe, and Martin Schindler. Integration von Modellen in einen codebasierten Softwareentwicklungsprozess. In *Modellierung 2006 Conference*, LNI 82, Seiten 67–81, 2006.

[GLRR15] Timo Greifenberg, Markus Look, Sebastian Roidl, and Bernhard Rumpe. Engineering Tagging Languages for DSLs. In *Conference on Model Driven Engineering Languages and Systems (MODELS'15)*, pages 34–43. ACM/IEEE, 2015.

[GR95] Radu Grosu and Bernhard Rumpe. Concurrent Timed Port Automata. Technical Report TUM-I9533, TU Munich, Germany, October 1995.

[GR11] Hans Grönniger and Bernhard Rumpe. Modeling Language Variability. In *Workshop on Modeling, Development and Verification of Adaptive Systems*, LNCS 6662, pages 17–32. Springer, 2011.

[GRJA12] Tim Gülke, Bernhard Rumpe, Martin Jansen, and Joachim Axmann. High-Level Requirements Management and Complexity Costs in Automotive Development Projects: A Problem Statement. In *Requirements Engineering: Foundation for Software Quality (REFSQ'12)*, 2012.

[GRR10] Hans Grönniger, Dirk Reiß, and Bernhard Rumpe. Towards a Semantics of Activity Diagrams with Semantic Variation Points. In *Conference on Model Driven Engineering Languages and Systems (MODELS'10)*, LNCS 6394, pages 331–345. Springer, 2010.

[HHK+13] Arne Haber, Katrin Hölldobler, Carsten Kolassa, Markus Look, Klaus Müller, Bernhard Rumpe, and Ina Schaefer. Engineering Delta Modeling Languages. In *Software Product Line Conference (SPLC'13)*, pages 22–31. ACM, 2013.

[HHK+14] Martin Henze, Lars Hermerschmidt, Daniel Kerpen, Roger Häußling, Bernhard Rumpe, and Klaus Wehrle. User-driven Privacy Enforcement for Cloud-based Services in the Internet of Things. In *Conference on Future Internet of Things and Cloud (FiCloud'14)*. IEEE, 2014.

[HHK+15a] Arne Haber, Katrin Hölldobler, Carsten Kolassa, Markus Look, Klaus Müller, Bernhard Rumpe, Ina Schaefer, and Christoph Schulze. Systematic Synthesis of Delta Modeling Languages. *Journal on Software Tools for Technology Transfer (STTT)*, 17(5):601–626, October 2015.

[HHK+15b] Martin Henze, Lars Hermerschmidt, Daniel Kerpen, Roger Häußling, Bernhard Rumpe, and Klaus Wehrle. A comprehensive approach to privacy in the cloud-based Internet of Things. *Future Generation Computer Systems*, 56:701–718, 2015.

[HKM+13] Arne Haber, Carsten Kolassa, Peter Manhart, Pedram Mir Seyed Nazari, Bernhard Rumpe, and Ina Schaefer. First-Class Variability Modeling in Matlab/Simulink. In *Variability Modelling of Software-intensive Systems Workshop (VaMoS'13)*, pages 11–18. ACM, 2013.

[HKR+07] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. An Algebraic View on the Semantics of Model Composition. In *Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'07)*, LNCS 4530, pages 99–113. Springer, Germany, 2007.

[HKR+09] Christoph Herrmann, Holger Krahn, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Scaling-Up Model-Based-Development for Large Heterogeneous Systems with Compositional Modeling. In *Conference on Software Engineeering in Research and Practice (SERP'09)*, pages 172–176, July 2009.

[HKR+11] Arne Haber, Thomas Kutz, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Delta-oriented Architectural Variability Using MontiCore. In *Software Architecture Conference (ECSA'11)*, pages 6:1–6:10. ACM, 2011.

[HKR12] Christoph Herrmann, Thomas Kurpick, and Bernhard Rumpe. SSELab: A Plug-In-Based Framework for Web-Based Project Portals. In *Developing Tools as Plug-Ins Workshop (TOPI'12)*, pages 61–66. IEEE, 2012.

[HR04] David Harel and Bernhard Rumpe. Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer*, 37(10):64–72, October 2004.

[HRR98] Franz Huber, Andreas Rausch, and Bernhard Rumpe. Modeling Dynamic Component Interfaces. In *Technology of Object-Oriented Languages and Systems (TOOLS 26)*, pages 58–70. IEEE, 1998.

[HRR+11] Arne Haber, Holger Rendel, Bernhard Rumpe, Ina Schaefer, and Frank van der Linden. Hierarchical Variability Modeling for Software Architectures. In *Software Product Lines Conference (SPLC'11)*, pages 150–159. IEEE, 2011.

[HRR12] Arne Haber, Jan Oliver Ringert, and Bernhard Rumpe. MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems. Technical Report AIB-2012-03, RWTH Aachen University, February 2012.

[HRRS11] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Delta Modeling for Software Architectures. In *Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteterSysteme VII*, pages 1 – 10. fortiss GmbH, 2011.

[HRRS12] Arne Haber, Holger Rendel, Bernhard Rumpe, and Ina Schaefer. Evolving Delta-oriented Software Product Line Architectures. In *Large-Scale Complex IT Systems. Development, Operation and Management, 17th Monterey Workshop 2012*, LNCS 7539, pages 183–208. Springer, 2012.

[HRRW12] Christian Hopp, Holger Rendel, Bernhard Rumpe, and Fabian Wolf. Einführung eines Produktlinienansatzes in die automotive Softwareentwicklung am Beispiel von Steuergerätesoftware. In *Software Engineering Conference (SE'12)*, LNI 198, Seiten 181–192, 2012.

[HRW15] Katrin Hölldobler, Bernhard Rumpe, and Ingo Weisemöller. Systematically Deriving Domain-Specific Transformation Languages. In *Conference on Model Driven Engineering Languages and Systems (MODELS'15)*, pages 136–145. ACM/IEEE, 2015.

[KER99] Stuart Kent, Andy Evans, and Bernhard Rumpe. UML Semantics FAQ. In A. Moreira and S. Demeyer, editors, *Object-Oriented Technology, ECOOP'99 Workshop Reader*, LNCS 1743, Berlin, 1999. Springer Verlag.

[KKP+09] Gabor Karsai, Holger Krahn, Claas Pinkernell, Bernhard Rumpe, Martin Schindler, and Steven Völkel. Design Guidelines for Domain Specific Languages. In *Domain-Specific Modeling Workshop (DSM'09)*, Techreport B-108, pages 7–13. Helsinki School of Economics, October 2009.

[KLPR12] Thomas Kurpick, Markus Look, Claas Pinkernell, and Bernhard Rumpe. Modeling Cyber-Physical Systems: Model-Driven Specification of Energy Efficient Buildings. In *Modelling of the Physical World Workshop (MOTPW'12)*, pages 2:1–2:6. ACM, October 2012.

[KPR97] Cornel Klein, Christian Prehofer, and Bernhard Rumpe. Feature Specification and Refinement with State Transition Diagrams. In *Workshop on Feature Interactions in Telecommunications Networks and Distributed Systems*, pages 284–297. IOS-Press, 1997.

[KPR12] Thomas Kurpick, Claas Pinkernell, and Bernhard Rumpe. Der Energie Navigator. In H. Lichter and B. Rumpe, Editoren, *Entwicklung und Evolution von Forschungssoftware. Tagungsband, Rolduc, 10.-11.11.2011*, Aachener Informatik-Berichte, Software Engineering, Band 14. Shaker Verlag, Aachen, Deutschland, 2012.

[Kra10] Holger Krahn. *MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering*. Aachener Informatik-Berichte, Software Engineering, Band 1. Shaker Verlag, März 2010.

[KRB96] Cornel Klein, Bernhard Rumpe, and Manfred Broy. A stream-based mathematical model for distributed information processing systems - SysLab system model. In *Workshop on Formal Methods for Open Object-based Distributed Systems*, IFIP Advances in Information and Communication Technology, pages 323–338. Chapmann & Hall, 1996.

[KRR14] Helmut Krcmar, Ralf Reussner, and Bernhard Rumpe. *Trusted Cloud Computing*. Springer, Schweiz, December 2014.

[KRS12] Stefan Kowalewski, Bernhard Rumpe, and Andre Stollenwerk. Cyber-Physical Systems - eine Herausforderung für die Automatisierungstechnik? In *Proceedings of Automation 2012, VDI Berichte 2012*, Seiten 113–116. VDI Verlag, 2012.

[KRV06] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Roles in Software Development using Domain Specific Modelling Languages. In *Domain-Specific Modeling Workshop (DSM'06)*, Technical Report TR-37, pages 150–158. Jyväskylä University, Finland, 2006.

[KRV07a] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Efficient Editor Generation for Compositional DSLs in Eclipse. In *Domain-Specific Modeling Workshop (DSM'07)*, Technical Reports TR-38. Jyväskylä University, Finland, 2007.

[KRV07b] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Integrated Definition of Abstract and Concrete Syntax for Textual Languages. In *Conference on Model Driven Engineering Languages and Systems (MODELS'07)*, LNCS 4735, pages 286–300. Springer, 2007.

[KRV08] Holger Krahn, Bernhard Rumpe, and Steven Völkel. Monticore: Modular Development of Textual Domain Specific Languages. In *Conference on Objects, Models, Components, Patterns (TOOLS-Europe'08)*, LNBIP 11, pages 297–315. Springer, 2008.

[KRV10] Holger Krahn, Bernhard Rumpe, and Stefen Völkel. MontiCore: a Framework for Compositional Development of Domain Specific Languages. *International Journal on Software Tools for Technology Transfer (STTT)*, 12(5):353–372, September 2010.

[LRSS10] Tihamer Levendovszky, Bernhard Rumpe, Bernhard Schätz, and Jonathan Sprinkle. Model Evolution and Management. In *Model-Based Engineering of Embedded Real-Time Systems Workshop (MBEERTS'10)*, LNCS 6100, pages 241–270. Springer, 2010.

[MMR10] Tom Mens, Jeff Magee, and Bernhard Rumpe. Evolving Software Architecture Descriptions of Critical Systems. *IEEE Computer*, 43(5):42–48, May 2010.

[MRR10] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. A Manifesto for Semantic Model Differencing. In *Proceedings Int. Workshop on Models and Evolution (ME'10)*, LNCS 6627, pages 194–203. Springer, 2010.

[MRR11a] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. ADDiff: Semantic Differencing for Activity Diagrams. In *Conference on Foundations of Software Engineering (ESEC/FSE '11)*, pages 179–189. ACM, 2011.

[MRR11b] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. An Operational Semantics for Activity Diagrams using SMV. Technical Report AIB-2011-07, RWTH Aachen University, Aachen, Germany, July 2011.

[MRR11c] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CD2Alloy: Class Diagrams Analysis Using Alloy Revisited. In *Conference on Model Driven Engineering Languages and Systems (MODELS'11)*, LNCS 6981, pages 592–607. Springer, 2011.

[MRR11d] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Modal Object Diagrams. In *Object-Oriented Programming Conference (ECOOP'11)*, LNCS 6813, pages 281–305. Springer, 2011.

[MRR11e] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Semantically Configurable Consistency Analysis for Class and Object Diagrams. In *Conference on Model Driven Engineering Languages and Systems (MODELS'11)*, LNCS 6981, pages 153–167. Springer, 2011.

[MRR13] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Synthesis of Component and Connector Models from Crosscutting Structural Views. In Meyer, B. and Baresi, L. and Mezini, M., editor, *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'13)*, pages 444–454. ACM New York, 2013.

[MRR14] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. Verifying Component and Connector Models against Crosscutting Structural Views. In *Software Engineering Conference (ICSE'14)*, pages 95–105. ACM, 2014.

[NPR13] Antonio Navarro Pérez and Bernhard Rumpe. Modeling Cloud Architectures as Interactive Systems. In *Model-Driven Engineering for High Performance and Cloud Computing Workshop*, CEUR Workshop Proceedings 1118, pages 15–24, 2013.

[PFR02] Wolfgang Pree, Marcus Fontoura, and Bernhard Rumpe. Product Line Annotations with UML-F. In *Software Product Lines Conference (SPLC'02)*, LNCS 2379, pages 188–197. Springer, 2002.

[PR94] Barbara Paech and Bernhard Rumpe. A new Concept of Refinement used for Behaviour Modelling with Automata. In *Proceedings of the Industrial Benefit of Formal Methods (FME'94)*, LNCS 873, pages 154–174. Springer, 1994.

[PR99] Jan Philipps and Bernhard Rumpe. Refinement of Pipe-and-Filter Architectures. In *Congress on Formal Methods in the Development of Computing System (FM'99)*, LNCS 1708, pages 96–115. Springer, 1999.

[PR01] Jan Philipps and Bernhard Rumpe. Roots of Refactoring. In Kilov, H. and Baclavski, K., editor, *Tenth OOPSLA Workshop on Behavioral Semantics. Tampa Bay, Florida, USA, October 15*. Northeastern University, 2001.

[PR03] Jan Philipps and Bernhard Rumpe. Refactoring of Programs and Specifications. In Kilov, H. and Baclavski, K., editor, *Practical Foundations of Business and System Specifications*, pages 281–297. Kluwer Academic Publishers, 2003.

[Rin14] Jan Oliver Ringert. *Analysis and Synthesis of Interactive Component and Connector Systems*. Aachener Informatik-Berichte, Software Engineering, Band 19. Shaker Verlag, 2014.

[RK96] Bernhard Rumpe and Cornel Klein. Automata Describing Object Behavior. In B. Harvey and H. Kilov, editors, *Object-Oriented Behavioral Specifications*, pages 265–286. Kluwer Academic Publishers, 1996.

[RKB95] Bernhard Rumpe, Cornel Klein, and Manfred Broy. Ein strombasiertes mathematisches Modell verteilter informationsverarbeitender Systeme - Syslab-Systemmodell. Technischer Bericht TUM-I9510, TU München, Deutschland, März 1995.

[RRRW15] Jan Oliver Ringert, Alexander Roth, Bernhard Rumpe, and Andreas Wortmann. Language and Code Generator Composition for Model-Driven Engineering of Robotics Component & Connector Systems. *Journal of Software Engineering for Robotics (JOSER)*, 6(1):33–57, 2015.

[RRW13a] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. From Software Architecture Structure and Behavior Modeling to Implementations of Cyber-Physical Systems. In *Software Engineering Workshopband (SE'13)*, LNI 215, pages 155–170, 2013.

[RRW13b] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. MontiArcAutomaton: Modeling Architecture and Behavior of Robotic Systems. In *Conference on Robotics and Automation (ICRA'13)*, pages 10–12. IEEE, 2013.

[RRW14] Jan Oliver Ringert, Bernhard Rumpe, and Andreas Wortmann. *Architecture and Behavior Modeling of Cyber-Physical Systems with MontiArcAutomaton*. Aachener Informatik-Berichte, Software Engineering, Band 20. Shaker Verlag, December 2014.

[RSW+15] Bernhard Rumpe, Christoph Schulze, Michael von Wenckstern, Jan Oliver Ringert, and Peter Manhart. Behavioral Compatibility of Simulink Models for Product Line Maintenance and Evolution. In *Software Product Line Conference (SPLC'15)*, pages 141–150. ACM, 2015.

[Rum96] Bernhard Rumpe. *Formale Methodik des Entwurfs verteilter objektorientierter Systeme*. Herbert Utz Verlag Wissenschaft, München, Deutschland, 1996.

[Rum02] Bernhard Rumpe. Executable Modeling with UML - A Vision or a Nightmare? In T. Clark and J. Warmer, editors, *Issues & Trends of Information Technology Management in Contemporary Associations, Seattle*, pages 697–701. Idea Group Publishing, London, 2002.

[Rum03] Bernhard Rumpe. Model-Based Testing of Object-Oriented Systems. In *Symposium on Formal Methods for Components and Objects (FMCO'02)*, LNCS 2852, pages 380–402. Springer, November 2003.

[Rum04] Bernhard Rumpe. Agile Modeling with the UML. In *Workshop on Radical Innovations of Software and Systems Engineering in the Future (RISSEF'02)*, LNCS 2941, pages 297–309. Springer, October 2004.

[Rum11] Bernhard Rumpe. *Modellierung mit UML, 2te Auflage*. Springer Berlin, September 2011.

[Rum12] Bernhard Rumpe. *Agile Modellierung mit UML: Codegenerierung, Testfälle, Refactoring, 2te Auflage*. Springer Berlin, Juni 2012.

[Rum16] Bernhard Rumpe. *Modeling with UML: Language, Concepts, Methods*. Springer International, July 2016.

[Sch12] Martin Schindler. *Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P*. Aachener Informatik-Berichte, Software Engineering, Band 11. Shaker Verlag, 2012.

[SRVK10] Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, and Gabor Karsai. Metamodelling: State of the Art and Research Challenges. In *Model-Based Engineering of Embedded Real-Time Systems Workshop (MBEERTS'10)*, LNCS 6100, pages 57–76. Springer, 2010.

[THR+13] Ulrike Thomas, Gerd Hirzinger, Bernhard Rumpe, Christoph Schulze, and Andreas Wortmann. A New Skill Based Robot Programming Language Using UML/P Statecharts. In *Conference on Robotics and Automation (ICRA'13)*, pages 461–466. IEEE, 2013.

[Völ11] Steven Völkel. *Kompositionale Entwicklung domänenspezifischer Sprachen*. Aachener Informatik-Berichte, Software Engineering, Band 9. Shaker Verlag, 2011.

[Wei12] Ingo Weisemöller. *Generierung domänenspezifischer Transformationssprachen*. Aachener Informatik-Berichte, Software Engineering, Band 12. Shaker Verlag, 2012.

[ZPK+11] Massimiliano Zanin, David Perez, Dimitrios S Kolovos, Richard F Paige, Kumardev Chatterjee, Andreas Horst, and Bernhard Rumpe. On Demand Data Analysis and Filtering for Inaccurate Flight Trajectories. In *Proceedings of the SESAR Innovation Days*. EUROCONTROL, 2011.