Taming the Complexity of Model-Driven Systems Engineering Projects

Arvid Butting, Timo Greifenberg, Bernhard Rumpe
Software Engineering, RWTH Aachen University
www.se-rwth.de

Andreas Wortmann
University of Rennes 1, IRISA, France
andreas.wortmann@irisa.fr

***Abstract*—Model-driven development has shown to facilitate systems engineering. It employs automated transformation of heterogeneous models into source code artifacts for software products, their testing, and deployment. To this effect, model-driven processes comprise several activities, including parsing, model checking, generating, compiling, testing, and packaging. During this, a multitude of artifacts of different kinds are involved that are related to each other in various ways. The complexity and number of these relations aggravates development, maintenance, and evolution of model-driven systems engineering (MDSE). For future MDSE challenges, such as the development of collaborative cyber-physical systems for automated driving or Industry 4.0, the understanding of these relations must scale with the participating domains, stakeholders, and modeling techniques. We motivate the need for understanding these relations between artifacts of MDSE processes, sketch a vision of formalizing these, and present challenges towards it.**

I. Motivation

The complexity of future interconnected cyber-physical systems, such as the factory of the future, fleets of automated cars, or smart grids poses grand challenges to software engineering. These challenges partly arise from the number of domains, stakeholders, and modeling techniques required to successfully deploy such systems. Model-driven engineering has shown to alleviate this, but introduces the challenge of managing the multitude of different artifacts, such as configuration, models, templates, transformations, and their relations as contributed by the different domains. Considering, for instance, software engineering for the factory of the future [1], successful deployment of a virtual factory [2] requires integration of modeling techniques to describe the factory's geometry, production processes and their optimization, software architecture, production systems with their interaction, manufacturing knowledge, and, ultimately, general-purpose programming language artifacts. The artifacts contributed by the respective domain experts are required in different stages of the development process and exhibit various forms of relations, such as design temporal dependencies, run-time dependencies, or creational dependencies (*e.g.,* a model and the code generated from it).

Moreover, how artifacts interrelate not only depends on their nature, but also on the context they are used in and the tools they are used with. For instance, software architecture models may be used for communication and documentation, model checking, transformation into source code, or simulation of the system part under development. In these contexts, the relations required to understand and process such an artifact may change: whereas the pure architecture model might be sufficient for communicating its structural properties, transformation into source code relates it to transformation artifacts and to the artifacts produced by this transformation.

Understanding and explicating these artifacts and their relations facilitates traceability of artifacts, change impact analysis, and interoperability of software tools crucial to successful model-driven engineering of the future systems of systems.

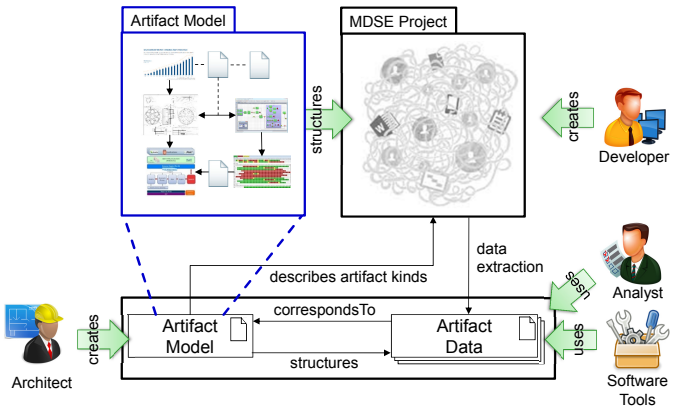

Figure 1. Handling the complexity of model-driven systems engineering projects with artifact models.

II. Modeling Artifact Relations

Typical MDSE projects require a multitude of different artifacts addressing the different domains' concerns (cf. Fig. 1). Managing the complexity of these artifacts requires understanding their relations, which entails understanding the relations between their languages as well as between the development tools producing and processing artifacts. We envision a MDSE future in which these relations are made explicit and machine-processable using modeling techniques. To this end, we desire reifying this information as first-level modeling concern in form of an explicit *artifact model* defined by the lead architect of the overall MDSE project. Such a model precisely specifies the kinds of artifacts, tools, languages,

This research has partly received funding from the German Federal Ministry for Education and Research under grant no. 01IS16043P. The responsibility for the content of this publication is with the authors.

[BGRW17] A. Butting, T. Greifenberg, B. Rumpe, A. Wortmann:
Taming the Complexity of Model-Driven Systems Engineering Projects.
In: Part of the Grand Challenges in Modeling (GRAND'17) Workshop. http://www.edusymp.org/Grand2017/, 2017.
www.se-rwth.de/publications/

and relations present in the MDSE project an thus enables representing the MDSE project in a structured way.

Such an artifact model should be capable to describe all different situations in terms of present artifacts and relations that could arise during its lifetime. The current situation of the project can be inspected by automatically extracting artifact data from the project according to the artifact models' entities and relations. This data corresponds to the artifact model ontologically, *i.e.,* represents an instance of it at a specific point in time. Analysts or specific software tools can employ this data to produce an overview of the current state, reporting issues, and identifying optimization potentials. Ultimately, this aims at enabling a more efficient development.

To this end, the artifact model comprises, among others, the organization of artifacts in the file system, the configuration of the tool chain, the trace of the last tool chain execution as well as static knowledge relations between artifacts leading to an architectural view including input models, artifacts forming specific tools or the target product, artifacts managed by the tools, output artifacts, and handcrafted artifacts.

This model depends on the technologies and tools used to develop the target product. Hence, it must be tailored specifically to each MDSE project. Globally, parts of such a model could be reused from similar projects (which might be achieved employing language engineering and composition methods on the artifact modeling language). For instance, model parts describing the interfaces of tools could be reusable as well as the types of specific artifacts and their relations might be applicable to multiple projects. Nevertheless, we assume each project will require manual artifact modeling to adjust existing structures. Ultimately, creating such an artifact model would

- ease communication, specification, and documentation of artifact, tool, and language dependencies,
- enable automated dependency analysis between artifacts and tools,
- support change impact analysis in terms of artifact tool, or language changes,
- support checking compliance of tools and proposing artifact, tool, and relation adaptations to 'glue' tool chains,
- facilitate an integrated view on the usage of tools in concrete scenarios,
- enable data-driven decision making, and
- enable computation of metrics and project reports to reveal optimization potentials within the tool chain.

III. Challenges of Artifact Modeling

There are few approaches towards such an artifact model. The approach described in [3] focusses on the integration of tools and the specification of tool chains and transformations between artifacts. Thus, artifacts managed within different tools are related to each other. The authors of [4] focus on an artifact-oriented way to describe a model-based requirements engineering process. Both approaches consider the requirement and design phases of MDSE projects only, but do not take code generation phases or implementation phases into account. Also, the tools themselves are not considered in the presented models. The authors of [5] contributed the idea of providing project data to analysts and software tools, but do not combine this idea with an explicit artifact model. Hence, there are still open challenges, which have to be overcome towards efficient and sophisticated artifact modeling.

First, the definition of a methodology on how to create artifact models tailored to the needs of a particular MDSE project. This includes:

- defining the scope of the MDSE project where artifact modeling can help taming the complexity,
- the development and selection of suitable modeling languages, tools and guidelines,
- the creation of model libraries providing reusable concepts common for system engineering projects, and
- development of reusable algorithms based on artifact models providing valuable analysis for common problems of system engineering projects.

Second, defining mechanisms, tools, and infrastructure supporting extraction and understanding of artifact data, including

- visualization capabilities, such as those proposed in [6],
- a methodology for integrating the different automated analysis tools to a given infrastructure,
- common interfaces for accessing artifact data, and the
- handling large amounts of artifact data efficiently.

Third, overcome modeling challenges, such as

- providing ways of defining and ensuring compliance between related software tools, such as editors, generators, or transformations, and
- integrating process data and historical data into such an artifact model to enable comprehending the state and changes of artifacts and their relations over time.

IV. Conclusion

Model-driven development can facilitate systems engineering. However, it introduces new challenges, of which taming the complexity of participating artifacts and their relations is an important one. We argue that investigating and reifying these is crucial to the successful deployment of future systems of systems and presented particular challenges future research should address to achieve this.

References

[1] A. Khan and K. Turowski, "A Survey of Current Challenges in Manufacturing Industry and Preparation for Industry 4.0," in *Proceedings of the First International Scientific Conference Intelligent Information Technologies for Industry (IITI 2016)*. Springer, 2016, pp. 15–26.

[2] S. Jain and D. Lechevalier, "Standards Based Generation of a Virtual Factory Model," in *Proceedings of the 2016 Winter Simulation Conference*. IEEE Press, 2016, pp. 2762–2773.

[3] P. Braun, *Metamodellbasierte Kopplung von Werkzeugen in der Softwareentwicklung*. Logos, 2004.

[4] D. Méndez Fernández and B. Penzenstadler, "Artefact-based requirements engineering: the AMDiRE approach," *Requirements Engineering*, vol. 20, no. 4, pp. 405–434, 2015.

[5] J. Czerwonka, N. Nagappan, W. Schulte, and B. Murphy, "Codemine: Building a Software Development Data Analytics Platform at Microsoft," *IEEE software*, vol. 30, no. 4, pp. 64–71, 2013.

[6] T. Greifenberg, M. Look, and B. Rumpe, "Visualizing MDD Projects," in *Software Engineering Conference (SE'17)*, ser. LNI. Bonner Köllen Verlag, 2017, pp. 101–104.