Goal Modeling and MDSE for Behavior Assistance

Judith Michael, Bernhard Rumpe, Lukas Tim Zimmermann
Software Engineering
RWTH Aachen University
Aachen, Germany
{michael, rumpe}@se-rwth.de, lukas.tim.zimmermann@rwth-aachen.de

***Abstract*—Systems providing their end-users behavior assistance must be customized to meet those users' needs and behavior goals. We investigate how it is possible to improve the engineering process of such systems and provide humans better support by using human behavior goals not only for analysis but also in the design and run-time of a system. Current research focuses either on the analysis phase of a system or uses goals at the run-time of an application. They do not consider the mapping of human behavior goals from one phase to the other one needed for generative approaches. Within this paper, we present our vision towards the use of goals and goal modeling for human behavior assistance in generated systems. We show its application by adding assistive functionalities to an existing, full-size real-world information system. For the engineering of this system, we follow a model-driven, generative software engineering approach. Our prototypical implementation towards the vision of using goal models for human behavior assistance shows the feasibility of the approach and provides first insights into how it could improve the assistive abilities of systems towards end-user goals.**

***Index Terms*—Model-Driven Software Engineering, Goal Modeling, Human Behavior Goals, Assistive System, Human-Centered Engineering**

I. Introduction

Motivation. Improving user support within assistive systems requires looking at both, the behavior to be supported and the underlying goals of each individual user to better tailor the support to specific needs. Assistive systems and services *provide situational support for human behavior based on information from previously stored and real-time monitored structural context and behavior data at the moment a person needs or asks for it* [1]. They use human behavior at run-time [2], [3] to provide support as step-by-step instructions. We assume that using goals of each individual user helps to better tailor the support to specific needs. We refer to such goals as *human behavior goals* and use this term to refer to desired end states of activities, e.g., a person is ready to leave the house. Systems providing behavior assistance have to be tailored to human behavior goals of their users which means they have to be reflected in the implementation of the system.

To use Model-Driven Software Engineering (MDSE) helps to reduce the resources and the development effort needed for engineering systems [4], aspects which are crucial for creating assistive systems. This is important because assistive systems often target niche application areas, with only a few potential application users with very specific needs. Delivering such systems for niche areas is of high importance, as they provide critical functionality such as supporting people with impairments, or ensuring safety in potentially dangerous work environments. Models used within model-based assistive systems are, e.g., behavior models to describe human behavior [5]–[7], context models [8], [9] or human emotion models [10].

Research gap. To the best of our knowledge no approach exists which maps goals in Requirements Engineering (RE) to human behavior goals used during support at run-time. We have to consider human behavior goals from analysis to run-time of an assistive system within a human-centered engineering approach. Following Grundy, Khalajzadeh, and Mcintosh [11], the mapping from higher-level abstract requirements models into lower-level system models (a set of models used to design the system) is still a challenging aspect.

Current research focuses either on identification of goals in the analysis phase or uses goals at run-time of an application specifically in self-adaptive systems. A challenge for the use of *goals detected within RE* in the analysis phase of the engineering of assistive systems is that they are typically defined on a *higher level of abstraction*, e.g., on the level of main functionalities [12] or non-functional requirements. Assistive systems require goals on a fine-grained level of detail, just as with actions and activities [13]. Another open issue for assistive systems is that *goals have to be manually mapped into system behavior*. MDSE aims at increasing the degree of automation which requires automated transformations. Agent-based and self-adaptive systems use *goals within the run-time of a system*. Agent-based systems use goals to define what an agent should pursue [14]. Some of these approaches consider human agents [15] but do not provide the fine-grained level of assistance needed for step-by-step support of assistive systems. Self-adaptive systems [16] use goals during run-time to set specific goal states which should be reached by a planner [17]. Dalpiaz et al. [18] distinguish between design-time goal models and run-time goal models which are used to analyze the run-time behavior of a system with respect to its requirements. These approaches do not consider human behavior goals but basic ideas from these approaches such as how goals are formulated and handled at run-time can be adapted for the use in assistive systems. The connection between requirements models and software models is still not a matter of research within these areas.

[MRZ21] J. Michael, B. Rumpe, L. T. Zimmermann:
Goal Modeling and MDSE for Behavior Assistance.
In: Int. Conf. on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 370-379, ACM/IEEE, Oktober 2021.
www.se-rwth.de/publications/

Research question. This papers discusses how it is possible to improve the engineering process of assistive systems by using human behavior goals in the analysis, design and run-time of a generated assistive system.

Contribution. We present our vision towards the use of goals and goal modeling for human behavior assistance in generated systems and show a first prototype towards this vision in a full-size real-world application. We have identified relevant human behavior in the analysis phase, modeled them as UML Use Case and Activity Diagrams (ADs) and identified goals. Within the design phase, we have mapped these goals and activities to according Domain-Specific Languages (DSLs) [19] and introduce a goal modeling DSL. To incorporate these models within the generation process of an assistive system, we use the goal models as additional input for the MontiGem generator framework [20] within a full-size real-world application. The goals are used to show users at run-time which options they have in using the system for two selected use cases. Our lessons learned from this prototypical implementation are how goals should be formulated, what main goal concepts might be used in and together with system models and how these concepts can be translated to code within the generation process of assistive systems as well as how goal modeling can be used during run-time of an assistive system.

Structure. The paper is structured as follows: Section II describes relevant concepts for this paper such as human behavior goals, assistive systems, goal modeling languages and the code generator used for the prototypical implementation. Section III shows our vision from real world processes and goals to requirements and system models used for code synthesis. Section IV presents it's application by adding assistive functionalities in an existing, full-size real-world information system. Section V relates our work to other approaches and Section VI discusses lessons learned from this first realization. The last section concludes.

II. Preliminaries

We introduce the relationship between human activities and goals as we believe that their use improves the assistive capabilities of systems. We provide an overview of different goal modeling languages and their used concepts and introduce the generator framework MontiGem which is used for the prototypical implementation towards our vision.

A. Human Behavior Goals in Assistive Systems

We understand *human behavior goals are desired end states of activities*. This follows the ideas in the following three definitions: Leon'tev [21] describes that human actions (concrete steps) are directed towards certain conscious goals. In philosophy, 'teleology' [22] (from greek telos, "end," and logos, "reason") states that activities or processes always have a reference to some purpose, end, goal or function. Eccles and Wigfield [23] define goals as desired end states which people try to attain. This happens through the cognitive, affective, and biochemical regulation of their behavior. We focus on individual goals [24] (in contrast to system goals often used in goal modeling approaches within the RE research area).

To use human behavior goals for the MDSE of assistive systems requires that goals as well as activities are modeled in machine understandable form either in general purpose languages or as DSLs. Within [25], we have introduced an idea on how behavior goals reduce the solution space for reasoning in assistive systems during run-time. That approach describes the process how to use sensory observations of human behavior together with domain-specific and general knowledge. *Within this paper, we discuss the implications on the engineering process of assistive systems and suggest to use human behavior goals in the analysis, design and run-time of a generated assistive system.*

B. Goal modeling languages

Goal modeling is used in a broad variety of research directions of Informatics, such as RE [26], [27] and specifically Goal-Oriented RE [9] or RE for adaptive systems [28], robotics [29], artificial intelligence [30], human-computer interaction [31], [32] information systems [33], or enterprise modeling [34], [35]. For a better understanding of the used concepts in goal modeling languages, we have analyzed some of the most common representatives.

The goal modeling approach i* [36] is a goal and actor oriented goal modeling language which is used in RE and other areas of Informatics. It contains two models to define a scenario. The Strategic Dependency (SD) model defines actors, their dependencies and dependencies between tasks, resources, goals and softgoals. Actors are differentiated by type and goals are differentiated by more or less essential goals. For dependencies three different degrees of strength exist. The Strategic Rationale (SR) model describes the processes in the scenario based on the the entities namely goals, soft goals, tasks and resources. Tasks can be decomposed and due to relations between goals and tasks it can be described which goal can be achieved by a task. A newer version of i* is called iStar 2.0 and it, e.g., introduces elements, rules and constraints on social dependencies [37], as well as types for actors [38].

Another prominent goal modeling approach is *Knowledge Acquisition in automated specification (KAOS)* [39], [40]. It uses four models related to each other to represent a scenario. A goal model defines goals, obstacle models define obstacles that can occur, an object model describes the agents, their goals and their relations and the operational model represents which action an agent has to take to reach a certain goal. Goals are differentiated by more or less essential goals.

In the *Goal-Based Requirements Analysis Method (GBRAM)*, goals capture the reasons why the system must exist [41]. Goals are divided into subgoals and similar to KAOS obstacles exist. Requirements specify how a goal should be achieved and constraints are formulated to specify which conditions have to hold to reach a goal. The agents are the participants in the system and so called scenarios describe the behavior of the system and its environment depending on conditions and the situation.

A goal modeling approach from agent-based systems is the Belief, Desire and Intentions (BDI) approach [42]. *Beliefs*, namely the world knowledge as facts about the environment and the state of the agent, are stored in a knowledge database. *Desires* are the goals of the agent, divisible into sub goals. *Intentions* are plans to achieving a goal. *Plans* can be differentiated by plans for sub goals, where the lowest level of the plan consists of elementary actions the agent can take.

	i* (iStar 2.0)	KAOS	GBRAM	BDI
Goals	x	x	x	x
Diff. into strength/type	x	x	x	-
Diff. into subgoals	-	-	x	x
Obstacles	-	x	x	-
Relations to goals	x	-	-	-
Constraints	x	-	x	-
Actions	x	x	-	x
Diff. into subactions	x	-	-	-
Relations to goals	x	x	-	x
Subjects/Agents	x	x	x	x
Roles	x	-	x	-
Relations between subjects	x	x	-	-
Relations to goals	x	x	x	x
Scenarios/World knowledge	-	-	x	x

Tab. 1. Different goal modeling languages in comparison

This comparison shows that different types of goals, and actions and subjects in relationship with goals are quite common. Table 1 provides an overview of the concepts of these goal modeling languages. There are several concepts that are shared throughout different goal modeling approaches. All goal modeling approaches differentiate between different types of goals in a certain way. One way is to differentiate goals by strength or type in for example hard and soft goals as done in [38]. Another way is to differentiate goals and sub goals which have to be reached to reach the parent goal as in [42]. Another concept shared by all evaluated goal modeling approaches is the concept of subjects or agents. Every goal modeling language offers the option to define stakeholders in the system. i* and GBRAM enhance this by introducing roles as types for subjects and agents [38], [41]. The subjects and agents are tightly bound to the goals. Except for GBRAM, all goal modeling languages introduce the concept of actions which are also related to the goals.

We have decided to create an own DSL based on these common concepts as (1) it can be tailored to specific needs [19] for assistive systems, (2) it can be extended if we realize the need for changes in an agile engineering process and (3) we need a textual DSL to be handled within our generative approach.

C. The generator framework MontiGem and its application

The generator framework MontiGem [20] generates a running information system from a set of models in different DSLs. The used DSLs are based on MontiCore [43], a language workbench and development tool framework for defining DSLs. Currently, the generator handles the following set of models (in different modeling languages):

- the domain model (UML Class Diagram (CD) language) including the main concepts relevant for data storage,
- several Graphical User Interface (GUI) models (GUI-DSL, a DSL based on [44]) defining each user interface of the application,
- data models (UML CDs) for each GUI page describing a subset of the data displayed on a certain page, and
- optional Object Constraint Language (OCL) models for defining data input validators, and
- tagging models (a DSL based on [45]) to add platform specific information to the domain model.

The generated application is realized as a client-server architecture using Java in the backend and Typescript/HTML within the Angular 6 framework in the frontend. Following a MDSE approach, data structures, the database connection as well as the communication between backend and frontend are generated. The backend contains the required business logic and a connection to a MYSQL database managed through Hibernate. The frontend visualizes data and handles user interaction. The generated Java and Typescript code can be extended by handwritten code using the TOP-mechanism [43]. This ensures that the handwritten code remains stable during re-generation and allows for agile development processes.

We are using MontiGem in a series of projects and application domains, e.g., energy management systems, platforms to support the engineering of wind turbines, cockpits for self-adaptive digital twins [46], or a platform for improving the understandablity of privacy policies of smart watches within the InviDas project[1].

We use the application developed in the *Management Cockpit for University Chair Controlling* project[2] to show a first prototypical implementation of our vision. It is the largest application developed with MontiGem with a size of app. 9000 lines of code (LOC) in models, 115.000 LOC hand-written and 390.000 LOC generated. MaCoCo allows university chairs to manage and control their finances including public budgets and third party funding, staff with, e.g., contracts and salary bookings and projects with, e.g., work packages and effort planning [47]. App. 160 out of 400 chairs at RWTH Aachen university use it. We have extended this application by adding assistive functionalities.

III. The Vision: Using Goal Modeling for Behavior Assistance

As already discussed in [25], we assume that the use of goals and goal modeling improves human behavior assistance in generated systems. Our vision covers real world processes and human goals which are transformed into analysis models, have to be mapped to system models and are then transformed into software code in an automated way within agile MDSE processes. Figure 1 provides an overview of this idea.

[1]https://www.se-rwth.de/projects/#InviDas
[2]https://www.se-rwth.de/projects/#MaCoCo

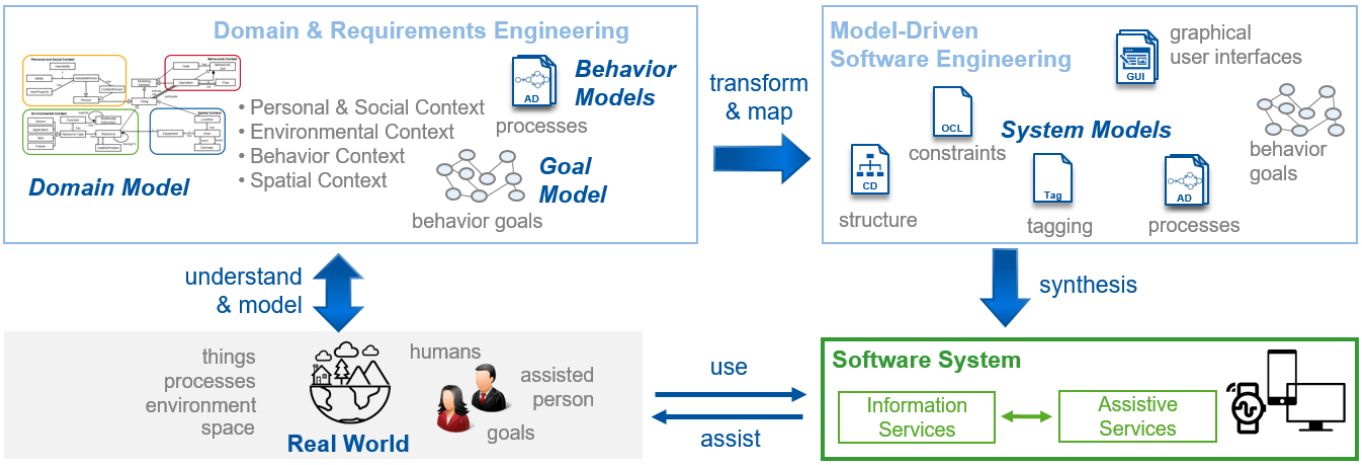

Fig. 1. From real world processes and goals to requirements and system models used for code synthesis

Starting from the real world, we want to create a software system that aims at supporting human processes i.e. we need to include assistive services. Engineering such systems requires to move from the problem domain (Figure 1 on the left) to the solution space (Figure 1 on the right). As methods to reduce the gap between the problem domain and the software implementation domain, we use agile MDSE approaches [48].

Within the *analysis phase* knowledge of domain experts from the real world needs to be transformed into scenarios, domain, behavior, and goal models. The analysis phase includes the elicitation of relevant domain concepts and context concepts. These can be collected as glossaries and further described in domain models using underspecified UML CDs.

According to Rolland and Salinesi [49], it is difficult for domain experts to deal with the fuzzy concept of goals. The use of scenarios supports their understanding. Thus, they propose to use goals to support scenario discovery and scenarios to help in goal discovery [49]. As their approach uses activities as goals and our approach requires states as goals, their approach might be used in an adapted way. We suggest to define specific scenarios as high level use cases and further detail it in behavior models such as UML ADs or BPMN models to get a better idea of the related processes which an assistive service should support.

Within the *design phase*, we have to transform and map RE models to system models. This requires, e.g., that UML CDs are further specified to be useable for code synthesis, exemplary UML Object Diagrams (ODs) can be defined to provide realistic test and dummy data, GUI models have to be created using information from scenarios, or input validators have to be specified for input forms. For goals identified in the analysis phase three possibilities exist (see Figure 2):

1) Goal concepts from the RE phase are transformed into concepts of a goal modeling language which combines goals and processes: This requires less DSL maintenance as there exists only one language. This risks that goal and process concepts are mixed and it reduces the reuseability of the DSL as it is purpose-specific.
2) Goal concepts from the RE phase are transformed into models in an own goal modeling language with connections to concepts in models of a process modeling language: This requires the maintenance of two different DSLs and a tagging language able to connect concepts from models of two different languages. Positive aspects are the reuseability of the DSLs and a clear assignment of language concepts.
3) Goal concepts from the RE phase are added as tags to concepts in models of a process modeling language: This requires a DSL for process modeling and a tagging DSL which allows to add statements to concepts of models in one DSL. This approach is not feasible if the structure of the concept goal is more complex, e.g., several attributes and types, and relationships between goals have to be modeled as well.

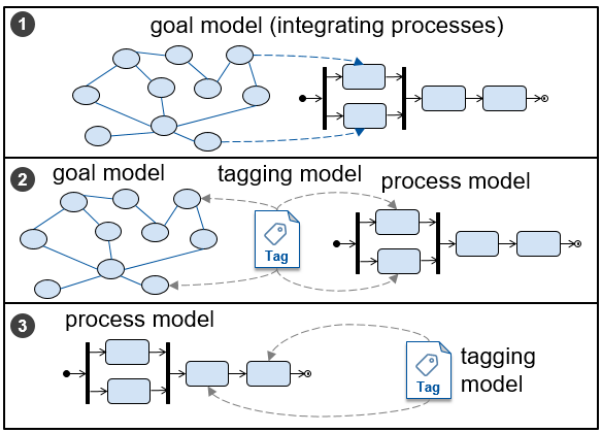

Fig. 2. Variants to use goals in MDSE

All three possibilities allow for MDSE approaches to take these system models and synthesize code for assistive services or whole assistive systems.

During *run-time of the software system*, goals and their connected activities in system processes can be tracked. Some goal-related functionalities of an assistive system are:

1) Continuously monitor and evaluate goals: This requires an intelligent monitoring component being able to handle models at run-time or at least a component which checks the current states of a set of goals. For activities, components such as process engines fulfill this functionality.
2) Provide structures to store a set of goals: This internal structure is needed to handle monitored goals.
3) Provide UI for goal overviews: This allows to show users unobtrusively what goals they could reach within a system.
4) Provide UI for support steps on each affected functionality: Assistance for end-users has to be integrated within the UIs providing functionalities such as to fill in a timesheet. This could be realized in various ways, e.g., own boxes, overlays or tooltip texts in GUIs, or additional acoustic information.
5) Provide UI to set, change or reject goals (optional): This depends on the degree of freedom of the processes within a generated application. It might be necessary that a user has to pursue certain goals to reach next ones.

This vision allows various design decisions which have to be further investigated and compared. The following section shows one possible implementation and Section VI discusses lessons learned.

IV. Prototypical Implementation Towards our Vision

We show the prototypical realization towards our vision within the full-size real-world application developed in the MaCoCo project, where assistive functionalities are added.

A. Considered Use Cases

The MaCoCo project uses an agile, MDSE approach which includes iterative requirements elicitation. Starting with a rough concept of the functionality to be realized, interviews with various end-users from different chairs as well as involved departments from the central university administration lead to better specifications and more detailed use cases in every iteration. The scenarios are described as use cases including textual descriptions, sketches of the graphical user interface, detailed descriptions of displayed and calculated data and process models for more complex functionalities with different participating roles. There is no explicit focus on workflows or process descriptions for all aspects as a main system aim is to provide users as much freedom as possible.

For the prototypical realization towards our vision, two use cases in MaCoCo were chosen: to match bookings between an external (SAP) account and an internal account and to keep timesheets for funded projects. To match bookings between these two systems is relevant because it is possible to *plan* bookings within the MaCoCo application which are realized at a later stage. To update this internal plans with the actual bookings within the SAP system requires comparison abilities and manual effort.

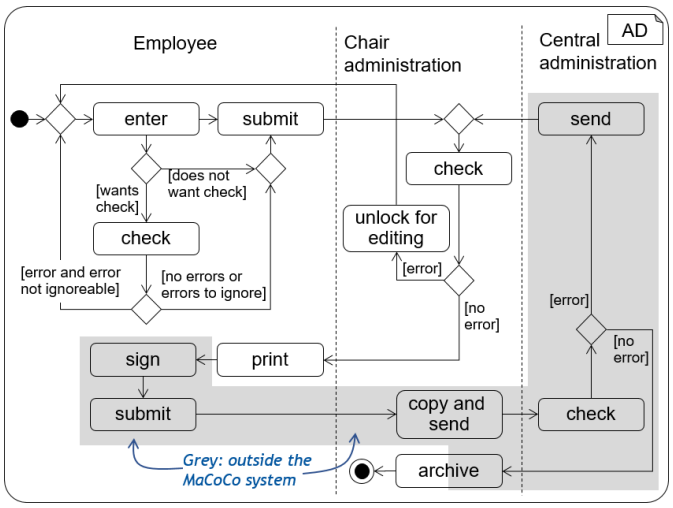

Fig. 3. Managing project timesheets in the MaCoCo application presented as an UML AD

The second use case is the timesheet process (see the UML AD in Figure 3). For universities, ensuring the correctness of timesheets is highly relevant, as errors can lead to less project funding being paid out than approved, funds having to be repaid, or in the worst case, the entire university being excluded from entire research funding programs.

Within this process, the employee has the option to enter his or her worked hours for one or more third party funded projects into a digital timesheet. After doing so, the timesheet can either be automatically checked or directly submitted to the chair administration. The chair administration examines the timesheet and returns it to the employee either for editing if it is incorrect or for printing and signing it. Incorrectness can be, e.g., caused by including more than the allowed working hours or work time entries during vacations or illnesses. The chair administration sends the printed and signed version to the central administration, which is in case of RWTH Aachen university the third party funding department. After the examination of the timesheet, the central administration returns it to the chair administration. If the timesheet was approved by the central administration, it is archived at the chair in two ways, namely physically and digitally within the MaCoCo application. If the timesheet is not approved, it can be returned to the employee for editing.

Considering these two use cases, we have identified possible user goals as states to reach for each activity, e.g., 'no errors in the check' for the activity 'check'.

B. System Design and the newly developed Goal DSL

Using the information from the scenario descriptions, processes, and human behavior goals, we define system models used as part of the generation process. As the described use cases are an addition to existing functionalities, the domain model already exists as UML CD and has to be extended, GUI

models have to be extended to represent support functionalities and goals have to be included. We have decided to realize variant 1 described in the vision, namely to use a goal modeling language which combines goals and processes.

This goal modeling language (in short GoalDSL) was newly designed for its use in this prototypical realization. A goal model consists of three different components: The *stakeholders*, the *goals* and the *actions*. To describe relationships in-between them, *links* connect goals and *relations* connect actions. Actions contain *pre-conditions* which have to be met before action execution, as well as *post-conditions*, which apply after action execution. Conditions may relate to *stakeholder properties*. A stakeholder is an actor in the system, for example represented by a class in MaCoCo defined in the domain UML CD. These concepts can be imported and addressed in the goal model by its relative names. For defining a sequence of actions for a scenario the goal model must include *start* and *end points* for the action relations.

To show text for user assistance in the frontend, goals and actions have to contain a text field with a specified text shown (1) if a goal is reachable, (2) an action is executable or (3) after a certain goal was reached, or (4) a certain action was executed. Depending on the use case, it might be necessary to not only use variables from classes in the backend in a goal model, but also frontend variables to assist the user regarding user input. Therefore, the GoalDSL offers the possibility to import frontend variables and use them in the conditions.

Now that we have this DSL, a goal model can be derived from the UML AD. Listing 1 shows an excerpt of the GoalDSL timesheet goal model which was derived from the AD in Figure 3. For every activity in the AD, an action in the goal model is derived. In this example the actions `enter` (ll. 21-24), `check` (ll. 25-28) and `submit` (ll. 29-32) are derived from the activity diagram. Every action has a goal which is reached when the action is executed. The stakeholders in Listing 1 are the `TimeSheet` (l. 8) and the `TimeSheetStatus` (l. 9) classes from the MaCoCo system. The order of events from the AD is represented through the action relations (ll. 36-39) with a start (l. 34) and an end point (l. 35).

```
1  package de.monticore.lang.goaldsl.timesheet;
2  import de.macoco.be.domain.cdmodelhwc.classes
3           .timesheet.TimeSheet;
4  import de.macoco.be.domain.cdmodelhwc.classes
5           .timesheetstatus.TimeSheetStatus;
6  goalmodel TimeSheet {
7    stakeholder{
8      TimeSheet z;
9      TimeSheetStatus TimeSheetStatus;
10   }
11   goal TimeSheetEntered{
12     action: enter
13   }
14   goal TimeSheetAutomaticallyChecked{
15     action: check
16   }
17   goal TimeSheetSubmitted{
18     action: submit
19   }
20   ...
21   action enter{
22     pre: true
23     post: z.getStatus == TimeSheetStatus.IN_ACQUISITION
24   }
25   action check{
26     pre: z.getStatus == TimeSheetStatus.IN_ACQUISITION
27     post: z.getStatus == TimeSheetStatus.IN_ACQUISITION
28   }
29   action submit{
30     pre: z.getStatus == TimeSheetStatus.IN_ACQUISITION
31     post: z.getStatus == TimeSheetStatus.INTERNAL_REVIEW
32   }
33   ...
34   start Start
35   end End
36   relation Start -> enter
37   relation enter -> enter ^ check ^ submit
38   relation check -> enter ^ check ^ submit
39   relation submit -> ...
40 }
```

Listing 1. Excerpt of the timesheet goal model derived from Fig. 3

C. Code Generation and Handling Goals during Run-Time

The MaCoCo application had to be extended to be able to use goal models for user assistance. We had to adapt the generator, the domain, data and GUI models and some handwritten parts of the system. Figure 4 shows the abstract data structure of the goal model logic in the backend.

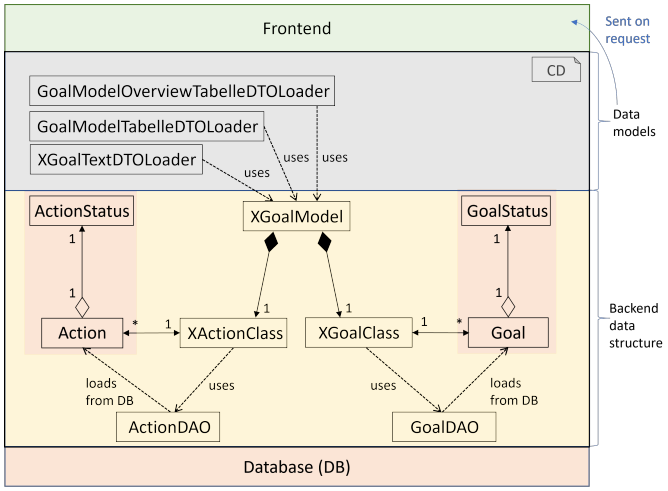

Fig. 4. Goal related system classes of MaCoCo (inspired by [47])

Domain Model. We extend the domain CD for the classes *Goal* and *Action* with the attributes name, type, status and an instanceID as well as the enumerations *GoalStatus* and *ActionStatus*. The status of goals or actions contains the information whether they have been (last) reached or (last) executed. Domain CD classes result in the database schema during the generation process and generated Data Access Object (DAO) classes allow to access instances of goals and actions during run-time. Handwritten code for specifying the methods extend the generated `GoalDAO` class.

Generator. We extend the main generator to create an `ActionClass` and a `GoalClass` for each goal model to maintain the stored goals for each instance. The `GoalClasses` contain methods to create, delete, load and mark goals as reached for an instance, as well as getter methods for the goal links, goal texts and other static information related to the goals independent of their instance. The `ActionClasses` contain methods to maintain the actions

for an instance and get, e.g., action texts or relations. The `ActionClass` also contains the logic to check the conditions for an action to determine whether it is executable or not, which the user can be informed about.

GUI Generator and frontend variables. Additionally, we adapt the generator for the GUI to be able to handle frontend variables, as they are not yet included in models. E.g., for each variable an attribute was generated into the `ActionClass`. This requires also to extend the model defining the commands between frontend and backend, where a command for each frontend variable was defined.

Data Transport. For data to be shown in the frontend, we create data models defining which data is loaded out of the database. For example on request of the frontend, the `XGoalTextDTOLoader` (see Figure 4) loads specific goal text (i.e. executable actions and last reached goals) for an instance of a goal model `X` from the database. The Data Transfer Object (DTO), which is filled with the data, is then sent to the frontend, where the data will be displayed.

Additionally, existing GUI models and pages were adapted to show support information, e.g., see help texts in Figure 6.

D. Assistance in the GUI

We use goal models as additional input for the generator and generate user assistance within the GUI of the MaCoCo application. Users get support for specific functionalities of the application: We show additional information about possible process steps and goals to reach.

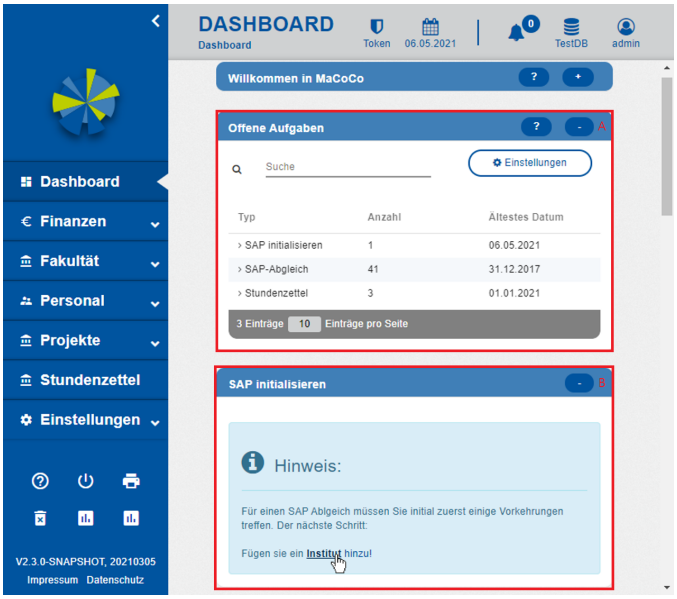

Fig. 5. MaCoCo dashboard excerpt with user assistance

Figure 5 shows an excerpt of the MaCoCo dashboard. We show the user all open assignments, represented as goal model instances in the system, that can or have to be fullfilled by the user. Area A in Figure 5 shows an example table which contains an overview over all the open goal instances in the system. There are 41 accounts where bookings have to be matched and three timesheets that have not yet been submitted within a specific month. Area B shows the help text for an open goal, which guides the user to another page where he has to initialize the connection to the external accounts.

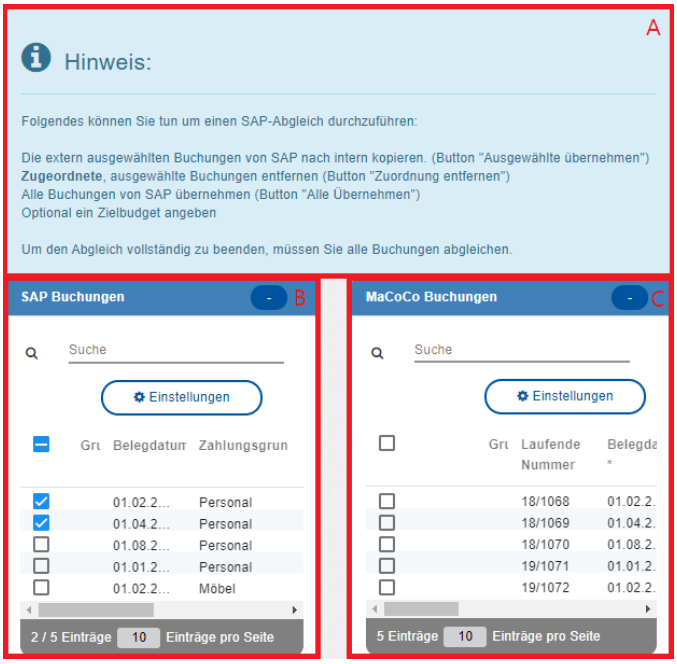

Fig. 6. Match bookings between an external (SAP) and an internal account with user assistance

Figure 6 area A shows a help text which assists the user to match bookings between an external (SAP) account (area B) and an internal account (area C). Depending on the bookings that the user has chosen, different options for the matching exist which influence the list shown in area A. In the help text, the user is informed about these options, as each option is represented by a goal in the goal model. In this concrete example in Figure 6, the user has selected external bookings in area B and in area C no booking is selected. Thus, the help text in area A informs the user, e.g., about the option of copying selected external bookings to the internal bookings because the conditions for executing this action are met.

V. Related Work

To the best of our knowledge, there is no approach that uses goal modeling to generate assistive systems, and there is no approach that maps goals in RE to human behavioral goals used for support at run-time.

Goal modeling in requirements engineering. Goal modeling and goal modeling languages are especially used in RE but the mapping from higher-level abstract requirements models into lower-level system models is hardly considered up to now.

Sutcliffe and Sawyer [33] introduce a theoretical framework for the conceptual modeling of personalized and context-aware systems and demonstrate it in a healthcare case study. They propose a user characteristics layer to model needs of individual users and generic user characteristics as well as

a personal goals layer including attitudes and preferences by individuals. Aspects of the approach can be considered for our vision as goals are represented as states and could be used as a base for a model-driven approach. The relationship between our approach and a Bayesian Network representing motivations, emotions and values is not considered yet.

There exist approaches to map goal models to BPMN processes [50] or align goals and business processes [51]. However, their focus is on organizational goals, which differ from our understanding of human behavior goals. Ghasemi and Amyot [52] investigate approaches for goal-oriented process mining. Whereas one of the identified categories of goal modeling and requirements elicitation seems to be promising to consider, the level of used goals is again too far away from defining goals as reachable states for activities.

Goal modeling for assistive systems. Baska et al. [15] implement the BDI framework [53] and introduce a multi-purpose goal model for a team of assistive agents in a digital coaching system. They assess user information via forms in seven categories of motives, e.g., maintaining physical strength and physical health, or having fun and being entertained. Additionally three questions are asked for each motive, namely the degree of importance for a user, to which extend the activity is currently performed in a satisfactory way and if an intervention by an agent is wanted (which results in defining a goal relating to this activity). This results in one goal with a certain rank which is related to each upper-level activity. In contrast to our approach, the activities are defined too general and such system would not be able to provide a fine-grained step-by step support based on human behavior goals.

Rafferty, Chen and Nugent [54] propose a goal recognition system together with an action planning mechanism as part of an assistive living solution to support inhabitants. Whereas their idea to use a goal repository, a goal recognition component, a specific goal generation mechanism, an activity planning component and an assistance provisioning component seems to be promising, they define goals as if they are activities, e.g., "get cup" or "make tea", and do consider goals as states which would be, e.g., "user has a cup in his hand" or "user has a cup or tea in his hand". To consider goals as reachable states is an important aspect of our approach.

Goal modeling in generative approaches. There exist several approaches for goal-oriented generation, e.g., [18], [55]–[58], touching other system architectures and application domains, however, they do not focus on human behavior assistance.

[55] and [56] describe a goal-oriented approach where goal models are used to generate code for a self-adaptive system. It uses and extends the software engineering methodology TROPOS to model goals and an own tool to analyze and translate the models into Java code, in form of a BDI-agent. In difference to our approach, they differentiate between hard- and softgoals and by goal types and do not support end-users in own behavioral steps.

Anda and Amyot [57] present an approach where goal models are used to generate arithmetic functions in a cyber-physical system. Similarly to our paper, all entities from the goal model are used for generation. The authors state that goal models often allow invalid combinations for alternatives and, thus, the goal models are not directly used for generation but combined with feature models. What they generate highly differs from what we aim to generate for end-users.

[58] describes an approach for the goal and knowledge-driven generation of neural dialogues which should allow machines to have conversations which are similar to human conversations. The machine creates an answer in a dialog for a specific topic and a given course of the conversation which keeps the conversation running and which is informative towards a specified goal. Unlike in our paper, goals have to be achieved in a limited number of steps. Goals are modeled as paths which connect two topics of a conversation that are then used to continue the conversation based on a knowledge graph.

Dalpiaz et al. [18] present an approach that uses goal models during the run-time of a self-adaptive system instead of using them for generation. The purpose of using the goal models during run-time is the diagnosis and monitoring of run-time behavior of the system. Similarly to our approach, the goal models are also presented by goal model instances during run-time. The authors state that common goal modeling approaches are too abstract for the use in a system during run-time and thus the goal modeling approach was extended to control the course of actions during run-time. In difference to our paper, the approach differentiates between design-time and run-time goal models. Design-time goal models are processed and refined to run-time goal models. This for example includes adding restrictions on run-time behavior to the design-time goal model.

VI. Discussion and Lessons Learned

The prototypical implementation has provided us first insights on how it is possible to improve the engineering process of assistive systems by using human behavior goals in the analysis, design and run-time of a generated information system. Our implementation is just one possible way of realization, as indicated in the description of the vision (Section III), but it gives us a better idea of which ways of realization work better than others.

Define goals as end states of activities. In the analysis phase, we have also realized, that goals are not an easy concept to describe (cf. [49]). Approaches such as [14] distinguish between procedural (action-based) goals and declarative (state-based) goals, which seemed to be promising. However, their definition mixes the concepts goal and activity [2]. Especially such approaches, which do not differentiate between activities and goals, provide no improvement in comparison to systems handling only behavior models at run-time.

What helped us the most was to clearly distinguish between activities and goals and to understand goals as desired end states of activities (following [23]). This approach helps to identify if a goal is reached. Such identifiable states are, e.g., concrete sensor values or for "virtual sensors" values of attributes in the frontend of an application. We use frontend

and backend variables of the system as conditions together with a concrete "name" for a goal and/or text which is understandable for the users. Depending on user input and system state during run-time of the assistive system, different *actions* are possible and displayed to the user.

Considering further goal-related concepts and goal attributes. Currently, we do not consider other concepts from existing goal modeling approaches for translation into software code or system behavior during run-time. This includes, e.g., attributes for describing goals in BDI approaches [59], or domain assumptions, obstacles or soft goals [33]. Our approach still lacks further investigation on the relevance of connections (positive and negative influence, hierarchies) between goals and how this influence should be reflected within the implementation. A first idea for negative influences on goals could be to show warnings for users, that other activities are not fulfillable if a certain activity was performed.

Relation between goals and subjects. To support subjects/agents in the assistive system, our GoalDSL offers the stakeholder concept. Stakeholders can be defined in the GoalDSL by importing classes from other models. The stakeholders in the generated MaCoCo system are in the current implementation objects of classes such as the *TimeSheet*-class in the previous example. Depending on the current state of an object, it is determined which actions can be executed in the system. We consider this approach as useful and suggest it for other realizations of the vision as well.

Relation between goals and actions. Our current implementation of the GoalDSL follows variant 1 in Figure 2 which still has the risk to mix goals and action concepts within the visualization. This leaves the impression that a process model and process engine would have also been sufficient. This impression would not occur, if end-users are, e.g., able to select certain goals in later stages of a process (not only the first reachable ones), see goals of the whole process or can further navigate through goals. In a further implementation we would rather use variant 2 in Figure 2 (a separate goal and behavior DSL with tagging inbetween concepts), as reuseable languages are better useable in future applications.

Ideas from the application perspective. The realized prototype could be further improved: (1) Consider the existing role system: It might be interesting for users to share goals with other users of that role or switch perspectives. (2) What to show users: There may be goals or sub-goals that are not relevant to a user and therefore should not be displayed to them. Users might decide this or it might be already captured in the goal model. (3) Formulating help texts: The displayed textual information must be formulated more clearly towards goals as reachable states and should exist in addition to task lists known from workflow systems. (4) Cover specific user needs: The prototype and used examples does not yet show the full potential of our vision.

When to use goal modeling at all. Our prototypical implementation has shown, that a realization is interesting if users are able to select and deselect certain goals. It makes less sense, if they have to reach a certain goal without alternative, as in this case a process/workflow model with a process engine would be a more feasible realization. The other aspect where realization makes sense is when there are *different alternatives* on how to reach a certain goal, i.e. the user can choose between alternative ways.

VII. Conclusion

The use of goal modeling enables person-specific support in assistive systems and services. The prototypical implementation has helped us to move one step forward towards the vision of using goal models for human behavior assistance, show its feasibility and provide first insights. It improved our understanding of how to formulate goals, what are the necessary concepts within a goal modeling language, and how these concepts can be translated to code within the generation process of assistive systems.

Future research in MDSE requires (1) to reconsider which parts of current the implementation might be realized as a part of the run-time environment of the application and not generated, (2) what further concepts might be relevant in existing goal modeling approaches and could be used for code generation and (3) how goals of different users and user groups can be efficiently handled during run-time. In a broader perspective, it would be interesting to cooperate with RE researchers to further investigate the relationship between models used within the analysis phase and system models to reduce the conceptual gap between the problem domain and the software implementation domain [48].

References

[1] K. Hölldobler, J. Michael, J. O. Ringert, B. Rumpe, and A. Wortmann, "Innovations in Model-based Software and Systems Engineering," *The Journal of Object Technology*, vol. 18, no. 1, pp. 1–60, 2019.

[2] J. Rafferty, C. D. Nugent, J. Liu, and L. Chen, "From Activity Recognition to Intention Recognition for Assisted Living Within Smart Homes," *IEEE Trans. on Human-Machine Systems*, vol. 47, no. 3, 2017.

[3] F. Al Machot, H. C. Mayr, and J. Michael, "Behavior Modeling and Reasoning for Ambient Support: HCM-L Modeler," in *Int. Conf. on Industrial, Engineering & Other Applications of Applied Intelligent Sys. (IEA-AIE 2014)*, ser. Lecture Notes in Artificial Intelligence, 2014.

[4] M. Völter, T. Stahl, J. Bettin, A. Haase, S. Helsen, and K. Czarnecki, *Model-Driven Software Development: Technology, Engineering, Management*, ser. Wiley Software Patterns Series. Wiley, 2013.

[5] J. Michael and H. C. Mayr, "Conceptual modeling for ambient assistance," in *Conceptual Modeling - ER'13*, ser. LNCS, vol. 8217. Springer, 2013, pp. 403–413.

[6] M. Giersich, P. Forbrig, G. Fuchs, T. Kirste, D. Reichart, and H. Schumann, "Towards an Integrated Approach for Task Modeling and Human Behavior Recognition," in *Human-Computer Interaction. Interaction Design and Usability*, J. A. Jacko, Ed. Springer, 2007, pp. 1109–1118.

[7] P. Parvin, F. Paternò, and S. Chessa, "Anomaly Detection in the Elderly Daily Behavior," in *14th Int. Conf. on Intelligent Environments (IE'18)*, 2018, pp. 103–106.

[8] J. Michael and C. Steinberger, "Context modeling for active assistance," in *Proc. of the ER Forum 2017 and the ER 2017 Demo Track at ER'17*. CEUR, 2017, pp. 221–234.

[9] R. Ali, F. Dalpiaz, and P. Giorgini, "A goal-based framework for contextual requirements modeling and analysis," *Requirements Engineering*, vol. 15, no. 4, pp. 439–458, 2010.

[10] M. R. Elkobaisi, H. C. Mayr, and V. A. Shekhovtsov, "Conceptual Human Emotion Modeling (HEM)," in *Advances in Conceptual Modeling*, G. Grossmann and S. Ram, Eds. Springer, 2020, pp. 71–81.

[11] J. Grundy., H. Khalajzadeh., and J. Mcintosh., "Towards Human-centric Model-driven Software Engineering," in *15th Int. Conf. on Evaluation of Novel Approaches to Software Engineering (ENASE'20)*, INSTICC. SciTePress, 2020, pp. 229–238.

[12] M. K. Curumsing, N. Fernando, M. Abdelrazek, R. Vasa, K. Mouzakis, and J. Grundy, "Emotion-oriented requirements engineering: A case study in developing a smart home system for the elderly," *Journal of Systems and Software*, vol. 147, pp. 215–229, 2019.

[13] C. Steinberger and J. Michael, "Towards Cognitive Assisted Living 3.0," in *Int. Conf. on Pervasive Computing and Communications Workshops (PerCom Workshops 2018)*. IEEE, 2018, pp. 687–692.

[14] M. B. van Riemsdijk, M. Dastani, and M. Winikoff, "Goals in Agent Systems: A Unifying Framework," in *7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS '08)*, 2008, p. 713–720.

[15] J. Baskar, R. Janols, E. Guerrero, J. C. Nieves, and H. Lindgren, "A Multipurpose Goal Model for Personalised Digital Coaching," in *Agents and Multi-Agent Systems for Health Care*. Springer, 2017, pp. 94–116.

[16] T. Bolender, G. Bürvenich, M. Dalibor, B. Rumpe, and A. Wortmann, "Self-Adaptive Manufacturing with Digital Twins," in *2021 Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2021, pp. 156–166.

[17] M. Fox and D. Long, "pddl2.1 : An Extension to pddl for Expressing Temporal Planning Domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 61–124, 2003.

[18] F. Dalpiaz, A. Borgida, J. Horkoff, and J. Mylopoulos, "Runtime goal models: Keynote," in *IEEE 7th Int. Conf. on Research Challenges in Information Science*, 2013, pp. 1–11.

[19] M. Mernik, J. Heering, and A. M. Sloane, "When and how to develop domain-specific languages," *ACM Comput. Surv.*, vol. 37, no. 4, 2005.

[20] K. Adam, J. Michael, L. Netz, B. Rumpe, and S. Varga, "Enterprise Information Systems in Academia and Practice: Lessons learned from a MBSE Project," in *40 Years EMISA (EMISA'19)*, ser. LNI, vol. P-304. GI e.V., 2020, pp. 59–66.

[21] A. N. Leont'ev, *Activity, Consciousness, and Personality*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[22] Teleology, *Encyclopaedia Britannica*, 2016, last access: 15.7.2021. [Online]. Available: https://www.britannica.com/topic/teleology

[23] J. S. Eccles and A. Wigfield, "Motivational beliefs, values, and goals," *Annual review of psychology*, vol. 53, pp. 109–132, 2002.

[24] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in *5th IEEE Int. Symposium on Requ. Eng.*, 2001, pp. 249–262.

[25] J. Michael, B. Rumpe, and S. Varga, "Human behavior, goals and model-driven software engineering for assistive systems," in *Enterprise Modeling and Information Systems Architectures (EMSIA 2020)*, vol. 2628. CEUR Workshop Proceedings, 2020, pp. 11–18.

[26] C. Rolland and C. Salinesi, "Modeling Goals and Reasoning with Them," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. Springer, 2005, pp. 189–217.

[27] J. Horkoff, N. Maiden, and J. Lockerbie, "Creativity and Goal Modeling for Software Requirements Engineering," in *Proc. ACM SIGCHI Conf. on Creativity and Cognition*, ser. C&C '15. ACM, 2015, pp. 165–168.

[28] M. Morandini, L. Penserini, A. Perini, and A. Marchetto, "Engineering requirements for adaptive systems," *Requirements Engineering*, vol. 22, no. 1, pp. 77–103, 2017.

[29] R. Heim, P. Mir Seyed Nazari, J. O. Ringert, B. Rumpe, and A. Wortmann, "Modeling Robot and World Interfaces for Reusable Tasks," in *Intelligent Robots and Systems Conference (IROS'15)*. IEEE, 2015.

[30] L. Braubach, A. Pokahr, D. Moldt, and W. Lamersdorf, "Goal Representation for BDI Agent Systems," in *Programming Multi-Agent Systems*. Springer, 2005, pp. 44–65.

[31] V. Kaptelinin, "Activity theory: implications for human-computer interaction," in *Context and consciousness. Activity theory and human-computer interaction*. MIT Press, 2001, pp. 103–116.

[32] D. S. McCrickard, C. M. Chewar, J. P. Somervell, and A. Ndiwalana, "A model for notification systems evaluation—assessing user goals for multitasking activity," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 10, no. 4, pp. 312–338, 2003.

[33] A. Sutcliffe and P. Sawyer, "Modeling Personalized Adaptive Systems," in *Advanced Information Systems Engineering*. Springer, 2013.

[34] S. Overbeek, U. Frank, and C. Köhling, "A language for multi-perspective goal modelling: Challenges, requirements and solutions," *Computer Standards & Interfaces*, vol. 38, pp. 1–16, 2015.

[35] A. Bock and U. Frank, "MEMO GoalML: A Context-Enriched Modeling Language to Support Reflective Organizational Goal Planning and Decision Processes," in *Conceptual Modeling*, ser. LNCS. Springer Int., 2016, vol. 9974, pp. 515–529.

[36] E. Yu, "Modeling Strategic Relationships for Process Reengineering," in *Social Modeling for Requirements Engineering*, 2011.

[37] L. Liu, "From i* to istar 2.0: An evolving social modelling."

[38] F. Dalpiaz, X. Franch, and J. Horkoff, "iStar 2.0 Language Guide," 05 2016.

[39] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of Computer Programming*, vol. 20, no. 1, pp. 3 – 50, 1993.

[40] O. Respect-IT, "A KAOS Tutorial," 2007.

[41] A. I. Antón, "Goal-Based Requirements Analysis," in *Int. Conf. on Requirements Engineering (ICRE '96)*, 1996, pp. 136–144.

[42] F. Schönmann, "BDI-Architektur (Beliefs – Desires – Intentions)," 2003.

[43] K. Hölldobler and B. Rumpe, *MontiCore 5 Language Workbench Edition 2017*, ser. Aachener Informatik-Berichte, Software Engineering, Band 32. Shaker Verlag, December 2017.

[44] A. Gerasimov, J. Michael, L. Netz, and B. Rumpe, "Agile Generator-Based GUI Modeling for Information Systems," in *Modelling to Program (M2P)*, A. Dahanayake, O. Pastor, and B. Thalheim, Eds. Springer, March 2021, pp. 113–126.

[45] T. Greifenberg, M. Look, S. Roidl, and B. Rumpe, "Engineering Tagging Languages for DSLs," in *Conf. on Model-Driven Engineering Languages and Systems (MODELS'15)*. ACM/IEEE, 2015, pp. 34–43.

[46] M. Dalibor, J. Michael, B. Rumpe, S. Varga, and A. Wortmann, "Towards a Model-Driven Architecture for Interactive Digital Twin Cockpits," in *Conceptual Modeling*, G. Dobbie, U. Frank, G. Kappel, S. W. Liddle, and H. C. Mayr, Eds. Springer, 2020, pp. 377–387.

[47] A. Gerasimov, P. Heuser, H. Ketteniß, P. Letmathe, J. Michael, L. Netz, B. Rumpe, and S. Varga, "Generated Enterprise Information Systems: MDSE for Maintainable Co-Development of Frontend and Backend," in *Comp. Proc. of Modellierung 2020 Short, Workshop and Tools & Demo Papers*. CEUR Workshop Proceedings, 2020, pp. 22–30.

[48] R. France and B. Rumpe, "Model-driven Development of Complex Software: A Research Roadmap," *Future of Software Engineering (FOSE '07)*, pp. 37–54, May 2007.

[49] C. Rolland and C. Salinesi, *Supporting Requirements Elicitation through Goal/Scenario Coupling*. Springer, 2009, pp. 398–416.

[50] N. Kraiem, H. Kaffela, J. Dimassi, and Z. Al Khanjar, "Mapping from MAP Models to BPMN Processes," *Journal of Software Engineering*, vol. 8, no. 4, pp. 252–264, 2014.

[51] R. Guizzardi and A. N. Reis, "A Method to Align Goals and Business Processes," in *Conceptual Modeling*, ser. LNCS, P. Johannesson, M. L. Lee, S. W. Liddle, A. L. Opdahl, and Ó. Pastor López, Eds. Springer International Publishing, 2015, vol. 9381, pp. 79–93.

[52] M. Ghasemi and D. Amyot, "From event logs to goals: a systematic literature review of goal-oriented process mining," *Requirements Engineering*, vol. 25, no. 1, pp. 67–93, 2020.

[53] C. Castelfranchi and R. Falcone, "Founding Autonomy: The Dialectics Between (Social) Environment and Agent's Architecture and Powers," in *Agents and Computational Autonomy*, ser. LNCS. Springer, 2004, vol. 2969, pp. 40–54.

[54] J. Rafferty, L. Chen, and C. Nugent, "Ontological Goal Modelling for Proactive Assistive Living in Smart Environments," in *Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction*, ser. LNCS. Springer, 2013, vol. 8276, pp. 262–269.

[55] M. Morandini, L. Penserini, and A. Perini, "Automated Mapping from Goal Models to Self-Adaptive Systems," in *23rd IEEE/ACM Int. Conf. on Automated Software Engineering*, 2008, pp. 485–486.

[56] L. Penserini, A. Perini, A. Susi, M. Morandini, and J. Mylopoulos, "A Design Framework for Generating BDI-Agents from Goal Models," in *6th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS '07)*. ACM, 2007.

[57] A. A. Anda and D. Amyot, "Arithmetic Semantics of Feature and Goal Models for Adaptive Cyber-Physical Systems," in *IEEE 27th Int. Requirements Engineering Conference (RE'19)*, 2019, pp. 245–256.

[58] A. Luo, C. Su, and S. Pan, "Goal-Oriented Knowledge-Driven Neural Dialog Generation System," in *Natural Language Processing and Chinese Computing*. Springer, 2019, pp. 701–712.

[59] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI Reasoning Engine," in *Multi-Agent Programming*, ser. Multiagent Systems, Artificial Societies, and Simulated Organizations. Boston, MA: Springer US, 2005, vol. 15, pp. 149–174.