Goal and model driven design of an architecture for a care service platform

L.O. Meertens
University of Twente
P.O. Box 217
7500 AE Enschede
+31 (0) 53 489 3500
l.o.meertens@utwente.nl

M.E. Iacob
University of Twente
P.O. Box 217
7500 AE Enschede
+31 (0) 53 489 3500
m.e.iacob@utwente.nl

L.J.M. Nieuwenhuis
University of Twente
P.O. Box 217
7500 AE Enschede
+31 (0) 53 489 3500
l.j.m.nieuwenhuis@utwente.nl

ABSTRACT

Service-Oriented Architecture holds the potential of allowing the development on-the-fly of flexible applications that can adapt rapidly by combining and reusing existing services. We believe that in order to react swiftly and coherently to changes, an architecture must provide a capability to capture how services, and the more complex applications based on them, realize business motivations. This research develops a framework and a method for goal-driven, model-driven, and service-oriented design. The framework includes goal modeling in the MDA stack, from CIM to code. By using this framework, we are able to create a system that is compatible with its business goals, and thus is flexible when business demands change. A case study demonstrates how our framework can be used to combine MDA, SOA, and goal modeling with business rules as an architecture for a care service platform.

Keywords

MDA, Goals, Business Rules, SOA, Web Services, Healthcare, Architecture.

1. INTRODUCTION

Large information systems, such as care systems, are hard to implement and maintain. Important reasons for this are the ever-changing demands from the business and various and different needs from the end users. Such demands can be captured in goal models, which makes goal modeling, and analysis critical in early design phases. In this paper, we claim that in the context of model-driven and service-oriented design, goal models may become an integral part of the system design.

In this article, we present a framework that includes goal modeling in the model-driven architecture (MDA)[17][26] stack, from CIM to code. By using this framework, we are able to create a system that is compatible with its business goals, and thus is flexible when business demands change. A service-oriented architecture (SOA) supports further flexibility by distributing functionality over individual components. In addition, we apply and illustrate the framework in the context of the U-care project [29], by means of a case from the healthcare sector. Within this project, our goals are to create an architecture for a platform for integrated homecare systems, which provides tailorable, evolvable, and non-intrusive care services. In U-care, a user-centric approach is taken. Therefore, the architecture will be based on a goal model that reflects primarily the functional concerns of end users and caregivers. In a second phase, also business modeling aspects (e.g., value, profitability, etc.) will be incorporated in the goal model and in the resulting platform architecture.

With the background set, the rest of the article is organized as follows. The next section presents the problem statement, as well as the general methods. Section 3 provides the methodology. Section 4 first explains the case to which we applied the methodology and then supplies the results of this exercise, it also discusses the domain specific languages used in this research to handle the problem. Section 5 relates other research to this article. Finally, section 6 concludes this article with a discussion and some directions for further research.

2. Research background

As we said in the introduction, our goal in this research was to develop a framework and a method for *goal-driven*, *model–driven*, and *service-oriented* design that is applicable in the development of the U-care service platform. One may easily notice that three paradigms are essential in the statement above, namely goal-driven and model-driven design and service-orientation. Their choice is motivated by the nature and desired properties of the future U-care service platform in which tailorability, adaptability, composability, and loose coupling of applications are critical. The U-care vision assumes that, since each patient is different and has specific needs, it should be possible to create and adjust software applications, using atomic services that suit each end-user. Our claim is that this vision of a service-based application can be realized, if the goals of each user can be captured in goal models, which are then incorporated in design models in an (partly) automated fashion (as proposed by the MDA). Before describing the framework in Section 2.3, which is the embodiment of the claim above, we give some background information about the three paradigms on which it relies.

2.1 Model-Driven Architecture

In most traditional software application development practices, the ultimate product of the design process is the realization", deployed on available realization platforms. In several model-driven approaches, however, intermediate models are reusable and are also considered final products of the design process. These models are carefully defined so that they abstract from details in platform technologies, and are therefore called computation-independent models (CIMs) and platform-independent models (PIMs), in line with OMG's MDA [2][17][26]. MDA (Model-Driven Architecture) has emerged as a new approach for the design and realization of software, and has eventually evolved into a collection of standards that raise the level of abstraction at which software solutions are specified. Thus, MDA fosters a design process and tools, which support the specification of software in modeling languages such as UML, rather than in programming languages such as Java.

The central idea of MDA is that design models at different levels of abstraction are derived from each other through model transformations. More specifically, different platform-specific models (PSMs) can be derived (semi-) automatically from the same PIM, making use of information contained by a platform model. Thus, MDA eventually advocates the principle that models can automatically be made directly executable, instead of being delivered to programmers as merely a source for inspiration or requirements, in order for them to create the real software [4]. The complete route from business model to executable code requires model transformations that function as a bridge between business process modelers and the IT department, and actually bring us one step closer to real and (partially) automated business-IT alignment.

2.2 Goal modeling and SOA

The central idea of SOA is that a service denotes the functionality that is relevant to the user of the service, without burdening the user with irrelevant details on how the service is implemented. SOA therefore holds the potential of allowing the development on-the-fly of flexible applications that can adapt rapidly by combining and reusing existing services. However, the technological state-of-the-art with respect to SOA (i.e., Web service technology [24]) so far only partly realizes the SOA potential. Design approaches incorporating the business view and with clear architectural guidelines are to a large extent still a subject of research.

We believe that in order to react swiftly and coherently to changes in the business view, an agile SOA architecture must provide a capability to capture how services, and the more complex applications based on them, realize business motivations (vision, goals, needs, objectives, policies, regulations, etc.). One way to incorporate the business view in SOA design is to express this view formally in terms of *goals*. Goal models provide a way to communicate system requirements to different stakeholders. Furthermore, specification of business goals is regarded as a means to raise the level of abstraction at which business logic is incorporated in model driven design in the context of service-oriented architectures.

Goals can be refined and, eventually, translated (i.e., *operationalised*) into concrete *business rules* (BR) and, then, integrated in the design and composition of services. Using business rules as vehicle to achieve this has the advantage of allowing the decoupling of the business logic (expressed as goals) from business operations, such as business rules, processes and their supporting applications. Furthermore, the effects of business logic changes (e.g., different needs for different patients, new laws and regulation or change of the protocol/policy, etc.) can be thus isolated, affecting the business operations only to a limited and controllable extent (since goals, business rules, and processes can be modeled and maintained separately). In this way, it becomes possible to explicitly manage and maintain goals and business rules, which are no longer hidden and hard-coded in processes and applications [11] and to achieve higher business process and software agility.

2.3 Framework

We base this research on a framework for the integration of SOA, MDA, business rules, and goals. According to [12], business rules can be derived as the operationalisation of an organization's goals and strategies. As such, rules not only play a role in capturing business goals but also in incorporating them in the design of application services, and in the design and control of the service orchestration. This is possible through the whole stack of MDA models, from CIM to PSM. Following the MDA paradigm, we assume that model transformations will be used in order to maintain the relationships between models at different abstraction levels in the MDA model stack (see the left-column of Figure 1).

The middle column of Figure 1 (which is a "service-oriented" version of MDA) illustrates this. In Figure 1, a distinction is made between the design space (the middle column), with models expressed in design languages such as UML, business process modeling languages, or architectural description languages, and the goal & business rule space (the right column), with goals and rules expressed in special-purpose specification, such as SBVR. This framework defines our vision on how to design service-oriented applications in a model-driven and goal-based way [12][13].

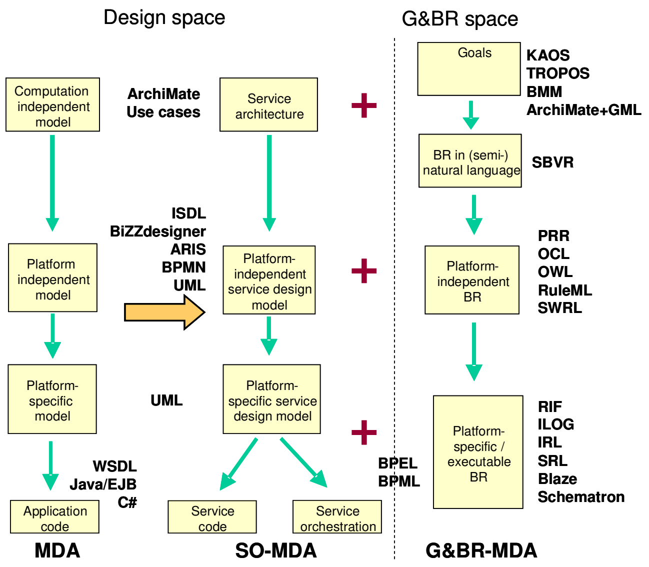

Figure 1: A model-driven view on the integration of service design enhanced with goals/business rules and their specification languages [12].

3. METHOD

As mentioned before, the framework presented in Section 2.3 for the goal-driven design of service-oriented applications serves as the basis for this research. Nevertheless, in order to use this framework in practice, we need a concrete method indicating the actual steps that should be followed during the design process.

Figure 2 illustrates this method, which is an "instantiation" of the framework in Figure 1. On the left-hand side of Figure 2, the activities (i.e., steps) we have carried out during our development process are mentioned. In the middle column, we specify the deliverables of each activity and in the right column, we present the concrete selection of specification languages and technologies used during the case study. Also an indication is given of how these columns can be mapped onto the MDA levels of models. The first two activities cover the preliminary requirements engineering steps necessary in order to obtain the CIMs. These models include high-level information, structure, and behavior models, as well as a goal model for the future application.

The first transformation step, from the CIM level to the PIM level, generates four new models. These are platform independent data, structure, and behavior (process) models, and Business Rules. These models can be enriched and, subsequently, validated. They do not only describe the future application but also identify the atomic services of which the future application is composed. For each of them, data, behavior, and interface models are to be provided. Please note that some of the business rules can also be seen and eventually implemented as (web) services. When the PIMs of all atomic services have been validated successfully, it is possible to deploy them as individual web services (application services). Furthermore, since these are merely the building blocks of the future business service (i.e., software application), in order to fulfill the initial application goals the application services still have to be orchestrated into a composite service. This step allows tailoring (i.e., customizing the application such that it fulfils the needs of a particular user) to take place. Before finally deploying the composite business service, a simulation/testing activity may also take place in order to ensure that a valid business service has been produced. Finally, the business service is ready to be deployed and consumed.

4. CASE

In this section, we focus on the U-care case of a single business service that we aim to model and implement following the framework and method presented in the previous section. Then, we describe the application of the method and the domain specific languages used in this research to handle the problem. Finally, we supply the results of this exercise.

4.1 Scenario

As part of the requirements engineering steps for the U-care service platform, usage scenarios are created (an example of which is presented in [14]). These scenarios define the main services that the care service platform has to support. Out of these services, we consider the case of a *reminder service* for end-users, specially designed for those patients who exhibit short-time memory deficiencies and have minor problems with remembering things related to their daily activities (e.g., appointments, taking medicines etc.). The main actor in our scenario, which would benefit from using this service, is Johanna. She is an elderly person with slight amnesia. The service reminds her of appointments, to take her medicines, activities in the area, birthdays, and other things to do. This helps her retain a proper

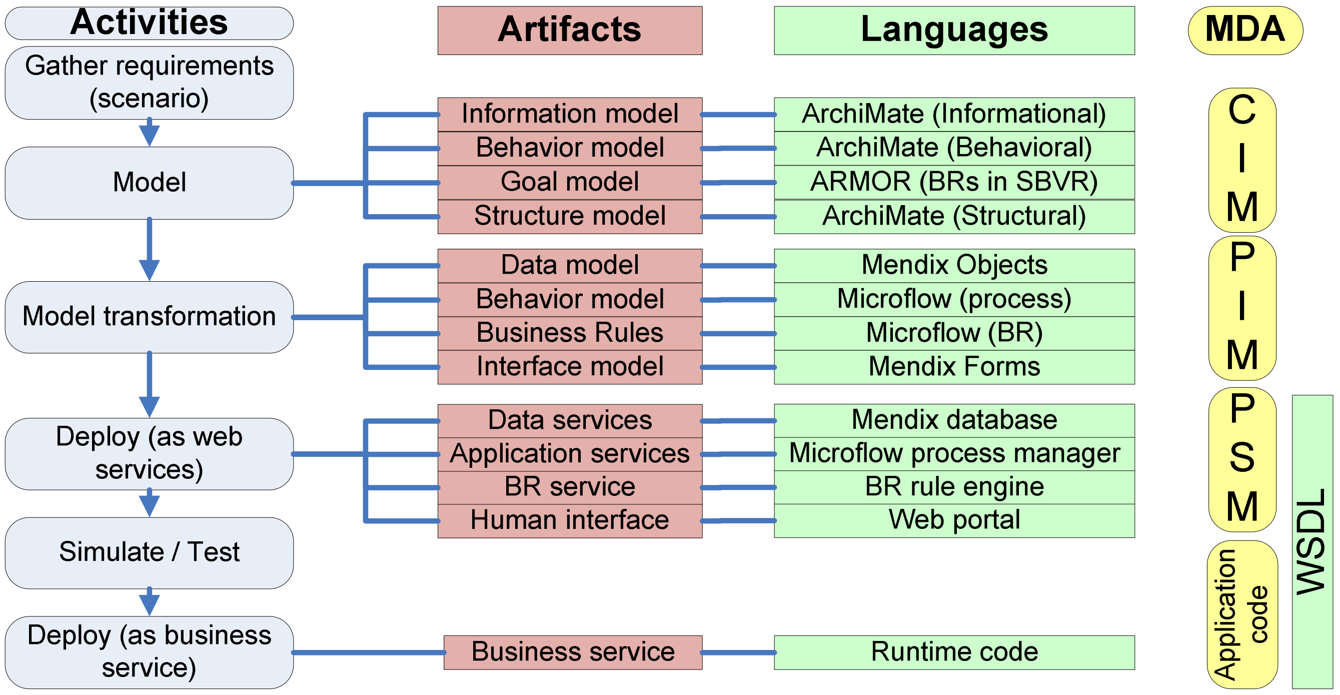

Figure 2: Method, including languages used and MDA stack.

daily routine and remember what she has to do. Reminders activate at certain moments in time or due to events in her context, and she can look at the things to do in the agenda when she wants to. Some reminders, such as taking medicines, require confirmation. She can add agenda items herself, but caregivers can also use the service to add their appointments.

4.2 Application of the method

For this case study, we apply the framework presented in section 3. The first activity, consisting of the requirements gathering and informal specification of the future application (i.e., business service), may entail the analysis of interviews with end-users and other stakeholders, scenarios, requirement documents, use cases, etc.

During the following activity, we choose ArchiMate [27][28] for the design of the CIM-level structure, behavior, and information models. ArchiMate has been developed for modeling enterprise architectures. From its philosophy, it does not model one specific architectural domain, but it focuses on a wider architecture that covers the whole organization. ArchiMate thus enables the possibility to model the global structure within a domain, but also the relationships between different domains. Just like an architectural drawing in classical building architecture describes the various aspects of the construction and use of a building, ArchiMate offers a common language for describing the construction and operation of business processes, organizational structures, information flows, IT systems, and technical infrastructure. [28]

For goal and requirements modeling, we use ARMOR [23], which is a recent extension of ArchiMate. The starting point for modeling high-level goals are stakeholders and their concerns. Goals are refined into sub-goals, by means of goal trees. Low-level goals (i.e., requirements and business rules) are then put in relation with services, processes, and applications that implement them. Consequently, ArchiMate/ARMOR models may also refer to Business Rules and show how these rules constrain the behavior of the future application. ARMOR is based on existing requirements modeling languages and is aligned with ArchiMate [23]. Thus, using ArchiMate and ARMOR in combination has the advantage of a seamless integration between the design and goal models (up to the level of modeling support).

However, ARMOR does not prescribe or contain any language for the formal specification of a Business Rules. To fill this gap, we have chosen for nearly natural language, as defined by the Semantics of Business Vocabulary and Business Rules (SBVR) standard for the Business Rules [21]. SBVR defines a structured sub-set of English vocabulary for defining business vocabularies and business rules in nearly natural language. It is underpinned with formal (first-order) logic. SBVR is an integral part of MDA. For this research, it gives us the modeling concepts to define business rules formally [21].

Together the languages above (i.e., ArchiMate, ARMOR, SBVR) provide sufficient support for the specification of a consistent, high-level architecture.

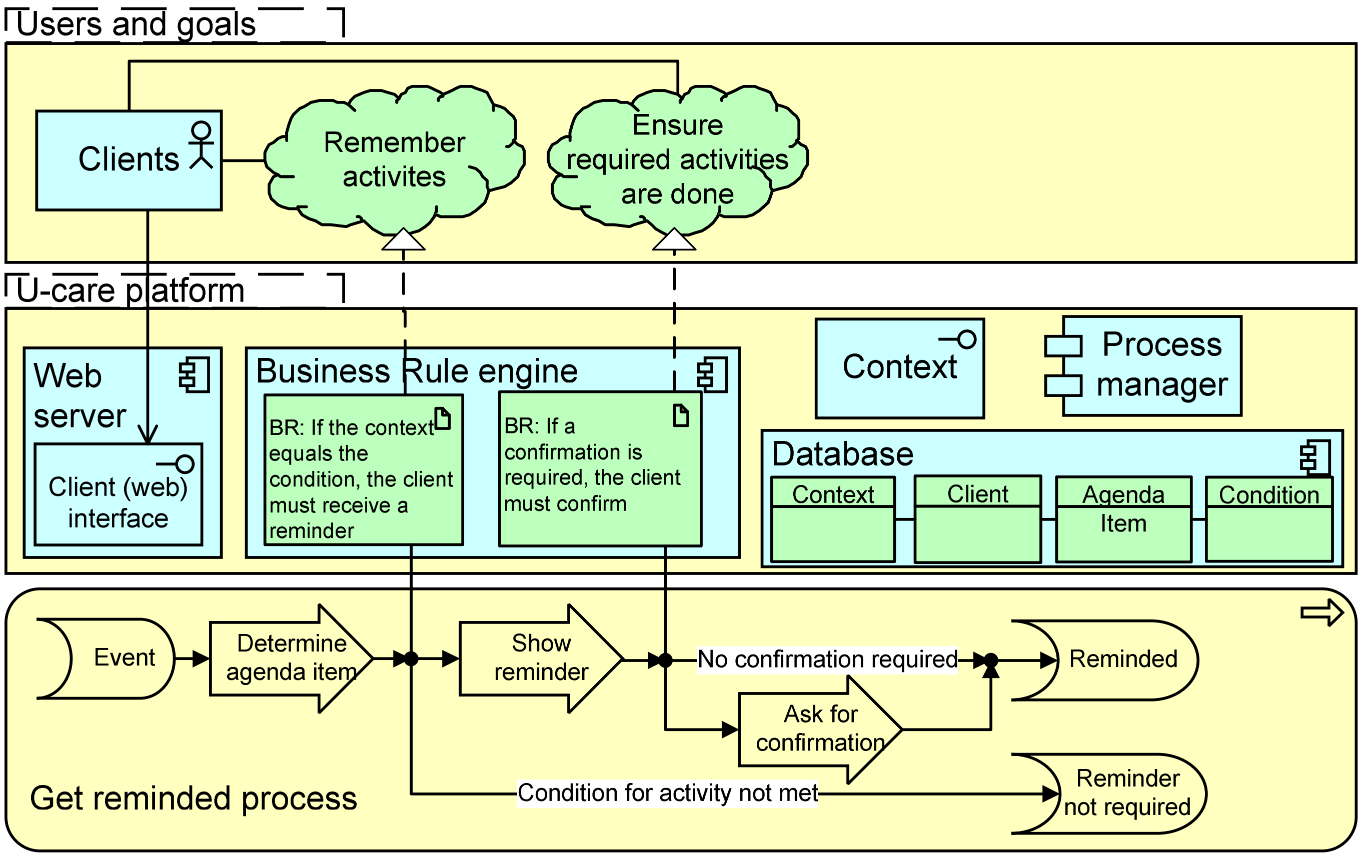

Figure 3: Compact architecture in ArchiMate and ARMOR.

The next activity is the transformation of the high-level architecture models into PIM models. Ideally, this transformation is automated. Nevertheless, the current theoretical and technological state-of-the-art does not allow us to perform such transformations (yet). Therefore, for the time being we chose to carry out this transformation manually. This shortcoming gives us the opportunity to continue our research in this area by investigating to what extent such transformations are feasible, and which configuration of standards, languages, and development platforms would make automatic transformation possible.

The PIMs, created by the manual transformation step, are modeled mainly in the DSLs of Mendix [15]. Mendix is a model-driven engineering platform that provides tools and architecture in the form of a runtime environment based on models for service-oriented applications. We use Mendix as the implementation platform for the case. This allows us to completely abstract from the actual code, although extension with Java is also possible in Mendix.

As indicated in Figure 2, four types of models are created at the PIM level: a data model, an interface model, behavior models, and business rule specifications.

The data model is specified in the Metamodel DSL of Mendix, which uses Mendix Objects. This data model is quite similar to a UML class diagram, and represents the information aspect.

An interface model, consisting of a navigable collection of Mendix Forms, represents the structure aspect on the PIM level. The business rules from the CIM level can be transformed from SBVR to Mendix Microflows [8]. Finally, it is necessary to map the behavior aspect of the high–level, CIM architecture onto corresponding models at the PIM level, both for each of the individual services, and for their orchestration (i.e., the composite business service). The behavior aspects of the architecture consist mainly of (business) processes which are mapped on low-level processes modeled as Microflows in Mendix. The Microflow DSL of Mendix is a subset of the Business Process Modeling Notation (BPMN)[15][20]. The core components of the language are start event, activities, gateways, and end event(s). It should be noted that Mendix uses (a special type of) Microflows for the structural modeling of business rules, which are meant to capture complex choices and enable easy reuse.

Deploying the Mendix models, publishing them as individual web services, orchestrating these web services and then deploying their orchestration brings us to the PSM level. Mendix takes care of most work in this transformation step. As this happens automatically, there is no need to alter the PSMs or code. Furthermore, we do not have any concerns with respect to the exact technologies and languages used for this transformation. The only aspect visible for the user from the PSM models is the human interface. This is used to handle the input and output, which the composite service requires or produce. The human interface is a simple web portal also generated automatically in Mendix.

While a final step would be to deploy the business service in an operational environment, we do not go further then testing by simulating the service. This provides a proof of concept.

4.3 Results

In the remainder of this section the concrete application of the method to the "Johanna" scenario is presented.

4.3.1 CIM

Figure 3 shows a detail of the high-level architecture at CIM level. It shows all four types of models (goal, structure, information, and behavior) in a single diagram. Due to limited space, only one of the three main services is shown, "Get reminded". The other two services are "View agenda items", and "Add/edit activities". These are simpler and their functions are apparent.

The goal model of the architecture in Figure 3 consists of the Client and its associated goals ("remember activities" and "ensure required activities are done"). These goals are operationalized by business rules (BR). These business rules are used in the process at the two gateways. There they decide which path to take. The data used by the business rules is derived from the real world. The client can enter agenda items and conditions for them (such as whether they require confirmation) through a (web) interface. Other information comes from the context of the client. This can include any data from sensors, such as blood values, or even from a video camera. For this prototype however, the context is limited to data input through a web portal by an administrator.

4.3.2 PIM

At the PIM level, a meta model exists in Mendix, which is comparable to the data model in ArchiMate. Attributes and cardinalities have been added to the objects. This transformation took place fairly smoothly, but requires manual intervention, as information is added, which could not be derived from the previous informational model.

The forms in Mendix are mainly transformed from the structural model, as they represent the client interface. In this case, Mendix automatically deploys the forms in a web portal. Besides the structural model, the forms also use the informational model (as fields in the forms must be bound to data objects), and the behavior model. The forms derived from the informational model are those to create, read, update, and delete (CRUD) the objects in the model. The behavioral model contributes with forms for each action where something needs to be shown to the client. While the CRUD forms are easy to derive from the meta model, it is harder to (automatically) find the forms for actions that need to show something.

The behavioral model transforms to Mendix Microflows in a straightforward fashion. However, as with the meta model, several things have to be added, which could not be done totally automatically. These things mainly include setting the types of activities, and the connections between objects, activities, and forms. However, the Mendix modeler allows to assign these fairly easily, by providing a point-and-click interface that makes invalid assignments impossible.

The final model to be transformed to the PIM level is the goal model. The business rules in the goal model are transformed from SBVR to limited Mendix Microflows. While in theory this could be done automatically [8][25], we could not find a tool to do so. Therefore, we did it manually, though according to the given approach. The limitation of Microflows for business rules compared to standard Microflows is that business rules are not allowed to change objects in the database.

Figure 4 shows the behavior model at the PIM level. The first two gateways connect to the business rules. The first actions retrieve data from the data model. The last action shows a form to the client, asking for confirmation if that is required.

4.3.3 PSM

The PSM and application code is not handled, as Mendix takes care of the complete steps from PIM to application code automatically. With a single mouse click the PIM deploys, and all the forms and actions are accessible through a web portal.

5. RELATED WORK

This work builds on several of the Freeband program's projects [9], most notably the A-MUSE project [1] and its lower-level project AWARENESS[3]. These respectively handle model-driven methodology of (mobile) services and context-aware infrastructure of such services. As our research illustrates, the U-care project extends this with goal-driven architecture. Besides this, context-aware tailoring and operability is handled. One of the results from the A-MUSE project is COSMO [22], a conceptual framework for service modeling and refinement. This can be combined with our work to facilitate the reuse of existing (legacy) technology in the developed architecture.

The MATCH project [16] has a similar purpose as our research. They develop individual home care systems for independent living, care in the community, and improved quality of life, while focusing on sensor technologies and their integration by means of a middleware platform. MATCH also considers the needs of users. Our research tries to achieve this by including the goals of the users from the start of the design process, while MATCH resolves this issue at the infrastructure level. The scenario-based approach is comparable in U-care and MATCH.

Including goals from the CIM level of the MDA-stack requires a way to model and transform these goals. The BServed project [5] has provided us with ARMOR, an extension to ArchiMate for this purpose. Its conceptual model and concrete are aligned with and adopted from existing languages BMM[6], i*[30], and KAOS[7]. However, the transformation from goal models to other (formal) specification languages is still a subject for future research.

Applications for the U-care platform can be derived from project such as MobiHealth[18], HealthService24[10], and MyoTel[19]. They researched several mobile applications for healthcare. These applications could be supported by the developed U-care architecture in the future.

6. CONCLUSION

The above case study demonstrates how our framework can be used to combine MDA, SOA, and goal modeling with business rules as an architecture for a care services platform. In principle, this framework applies to other domains as well. Our choice for the specific languages used is based on convenience (i.e., availability of modeling software), although the level in the MDA model and the technical feasibility of model transformation significant limit the possibilities. The presented methodology can be applied to other tools and other languages, if they support the main concepts such as business rules and processes.

Automatic transformation from the PIM to PSM level seems to occur correctly and successfully in Mendix. Transforming from CIMs to PIMs is not automated yet. Whether this will ever be fully possible is debatable. At the moment, this step seems to require additional information, which is not (and should not?) be captured in the CIMs. Therefore, it has to be added manually. Mendix is working on providing model transformation from ArchiMate to Mendix though.

The framework provides several opportunities for further research. The first thing that comes to mind is applying it in different, more complex, contexts and scenarios. Within the U-care project, for example, we plan to create an overall architecture for the combination of all requirements. Three usage scenarios capture these requirements currently. They cover the areas of monitoring and virtual communities [14]. Secondly, the framework leaves opportunities for extension. Round-trip engineering, or at least backward engineering, would help to improve interoperability. It could allow existing and newly developed services, as well as input and output methods, to

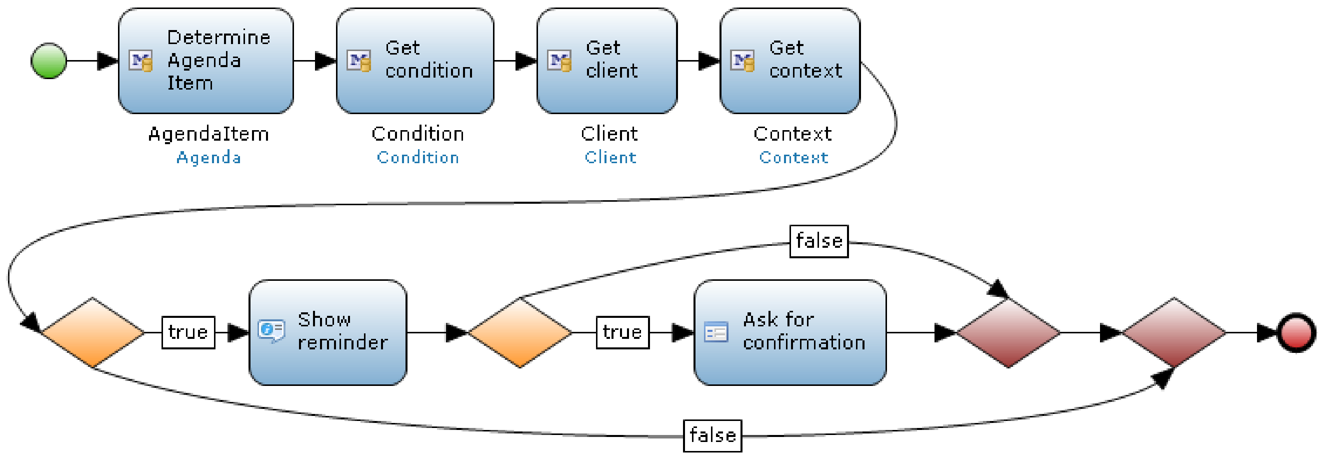

Figure 4: Behavior model at the PIM level. The gateways connect to business rules.

interface with the system easily. The information, behavior, and structure aspects could be abstracted from WSDL to the PIM level, as supported by the COSMO [22] framework for example.

7. ACKNOWLEDGMENTS

This work is part of the IOP GenCom U-CARE project, which the Dutch Ministry of Economic Affairs sponsors under contract IGC0816.

8. REFERENCES

[1] A-MUSE project, "*A-MUSE website*" http://www.freeband.nl/project.cfm?language=en&id=489 Accessed on September 14, 2009

[2] J.P.A. Almeida, M.E. Iacob, H. Jonkers, and D. Quartel, "Model-Driven Development of Context-Aware Services," *Distributed Applications and Interoperable Systems*, Heidelberg: Springer Berlin, 2006, pp. 213-227.

[3] AWARENESS project, "*AWARENESS website*" http://www.freeband.nl/project.cfm?id=494&language=en Accessed on September 14, 2009.

[4] J. Bézivin, "In Search of a Basic Principle for Model Driven Engineering," *Novatica Journal, Special Issue, March-April*, 2004.

[5] BServed project, "*BServed website*" http://www.novay.nl/okb/projects/bserved/4520 Accessed on September 14, 2009.

[6] Business Rules Group, *The Business Motivation Model – Business Governance in a Volatile World. Release 1.3*, The Business Rules Group, 2007.

[7] A. Dardenne, A.V. Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of computer programming*, 1993.

[8] R. Eder, A. Filieri, T. Kurz, T. Heistracher, and M. Pezzuto, "Model-transformation-based Software Generation utilizing Natural language notations," *2nd IEEE International Conference on Digital Ecosystems and Technologie. (DEST2008).* , 2008, pp. 306-312.

[9] Freeband program, "*Freeband website*" http://www.freeband.nl/index.cfm?language=en Accessed on September 14, 2009.

[10] HealthService24 project, "*HealthService24 eTEN-517352*" http://www.healthservice24.com/ Accessed on September 14, 2009

[11] L. Hermans, W. Lemahieu, J.Vanthienen, "Real agility and transparency requires a combination of BPM/SOA, EDA and BRA", In *Proceedings of the 6th European Business Rules Conference*, Düsseldorf (Germany), Jun. 18-19, 2007.

[12] M.E. Iacob, H. Jonkers, A Model-driven Perspective on the Rule-based Specification of Services, *Enterprise Information Systems*, Volume 3, Issue 3 August 2009, pages 279 – 298.

[13] M.E. Iacob, D.C.F. Rothengatter, and J. van Hillegersberg, "A Health-care Application of Goal-driven Software Design," *Applied Medical Informatics*, vol. 24, 2009, pp. 12-33/

[14] J-W. van 't Klooster, B-J. van Beijnum, P. Pawar, K. Sikkel, L. Meertens, and H. Hermens, "What Do Elderly Desire? A Case For Virtual Communities," *Proceedings of the International Workshop on Web Intelligence and Virtual Enterprises (WIVE09)*, Thessaloniki, Greece: 2009.

[15] R. Kruit, D. Roos, Modeling the Agile Enterprise: From Software Engineering to Business Engineering, Mendix Whitepaper, available at http://www.mendix.com/site/files/whitepapers/Enterprise_Whitepaper_Mendix.pdf Accessed on September 14, 2009.

[16] MATCH project, "MATCH - Mobilising Advanced Technologies for Care at Home" http://www.match-project.org.uk/ Accessed on September 14, 2009.

[17] J. Miller and J. Mukerji, *MDA Guide Version 1.0.1*, Object Management Group, 2003.

[18] Mobihealth project, "*MobiHealth - Shaping The Future Of Healthcare*" http://mobihealth.org/ Accessed on September 14, 2009

[19] Myotel project, "*Myotel website*" http://www.myotel.eu/ Accessed on September 14, 2009

[20] OMG, *BPMN 1.0: OMG Final Adopted Specification*, Object Management Group, 2006.

[21] OMG, *Semantics of Business Vocabulary and Business Rules (SBVR), v1.0*, Object Management Group, 2008.

[22] D.A. Quartel, M.W. Steen, S. Pokraev, and M.J. van Sinderen, "COSMO: A conceptual framework for service modelling and refinement," *Information Systems Frontiers*, vol. 9, 2007, pp. 225–244.

[23] D. Quartel, W. Engelsman, and H. Jonkers, "Modelling requirements in enterprise architectures," *Proceedings of EDOC 2009.*

[24] M. P. Papazoglou, *Web services: principles and technology*. Harlow: Pearson Prentice Hall, 2008.

[25] A. Raj, T.V. Prabhakar, and S. Hendryx, "Transformation of SBVR business design to UML models," *Proceedings of the 1st India software engineering conference (ISEC 2008)*, Hyderabad, India: ACM, 2008, pp. 29-38.

[26] R. Soley and the OMG Staff Strategy group, *Model Driven Architecture*, Object Management Group, 2000.

[27] The Open Group, *ArchiMate 1.0 Specification*, The Open Group, 2009.

[28] The Open Group, "*ArchiMate*" http://www.archimate.org. Accessed on September 14, 2009.

[29] U-care Project, "*U-care website*" http://ucare.ewi.utwente.nl. Accessed on September 14, 2009.

[30] E.S.K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," *Proceedings of the Third IEEE International Symposium on Requirements Engineering* , 1997, pp. 226–235.