Model-Driven Development of Human Tasks for Workflows

Stefan Link[1], Philip Hoyer[1], Thomas Schuster[2], Sebastian Abeck[1]
[1]*Cooperation & Management, Universität Karlsruhe (TH), Germany*
[2]*FZI Forschungszentrum Informatik, Germany*
{ link | hoyer | abeck } @ cm-tm.uka.de, schuster@fzi.de

Abstract

In order to increase efficiency, enterprises support their business processes by information technology (IT). The majority of business processes requires human interaction. By means of human interaction the complexity of the supporting IT grows. Model-driven approaches to software development are a promising solution to be able to cope with this complexity. According to these approaches all aspects of the developed software are captured in models and automatically transformed to source code of the desired platform. Currently there is still a lack of precise models for capturing necessary aspects of human interaction. Hence there is still a lot of manual development and configuration work to do to enable humans to perform a task within IT supported business processes. In this article we demonstrate an approach to model human tasks for business processes and propose an extension to Service-Oriented Architecture (SOA) to support the execution of human tasks. A case study fortifies the applicability of this approach.

1. Introduction

In order to stay competitive, enterprises try to align their business processes with IT. Business processes which can be completely supported by IT are focused by this article and short referred to as workflows. As during the execution of workflows an interaction by a human performing a task may be necessary, the human interaction part has attracted interest of both researchers and industry recently. Integrating humans to a workflow is accompanied by additional requirements not only concerning the definition of the workflow itself but also the execution environment [1]. If a workflow comprises human tasks, a user interface is needed. Also the human tasks have to be controlled to ensure proper execution. Consequently, the underlying software architecture has to provide the means to support human tasks [2]. As complexity of workflow development grows by integrating humans, complexity of the supporting IT rises as well [3]. The great variety in platforms, operating systems and devices are just a few aspects that cause the present complexity. Hence, new approaches to software development and software architectures arise to cope with that complexity.

With Model-Driven Architecture (MDA) [4], the modeling of a software system gains additional values. The abstraction through models allows for a better overview to the whole software system and to overcome heterogeneity and complexity [5]. Models are used as basis for transformation to source code [3]. To provide automatic transformations to source code of any desired platform is one of the main goals of MDA. Therewith, a high flexibility in software development, a shortened software development time and an increased software quality can be achieved [7].

Human tasks have to be dealt with as integral part of workflows [1]. Yet integrating human tasks to workflows still demands a high manual development and configuration effort. Although many details concerning human tasks, like the role which is qualified to execute a task, are available in the early stages of a software development process, due to insufficient means, these details are unfortunately only captured in an informal manner [8]. Thus, instead of an automated transformation as aimed by MDA, manual transformation and configuration steps have to be executed. Besides the effort, these manual steps often lead to error-prone software with significant quality losses [5, 6].

With Service-Oriented Architecture (SOA), a promising approach to overcome heterogeneity of existing software systems and for a flexible alignment of business and IT has evolved. Hence, SOA additionally has to cope with the execution of human tasks. For instance, there has to be some kind of service-like software component monitoring and ensuring the execution of human tasks. As many vendors still use their own process languages and individual software components to execute human tasks, a common approach for SOA has to be established.

In summary there is the need for a means to capture manifold aspects of human tasks on a modeling level to reduce complexity and error-proneness at the same time, while increasing the flexibility and quality of the developed workflows. On the other hand a SOA supporting human tasks is necessary.

In this article, we therefore present an extension of the Unified Modeling Language (UML) [9] enabling a platform-independent and easy to use modeling of human tasks. The developed models are executable and can be transformed to source code of any desired platform. This increases portability significantly. To prove the applicability of our approach, we present a case study transforming the developed models to source code for deployment on an extended SOA, which allows the execution of human tasks.

Accordingly, the remainder of this paper introduces the state of the art in the context of model-driven development focusing human tasks in section 2. In section 3 an extension to UML is presented and put into practice by a case study. Necessary extensions for SOA to handle human tasks are discussed in section 4 followed by a conclusion and outlook on future work in this area closing the body of this paper.

2. Related work

UML and the Business Process Modeling Notation (BPMN) [10] are just two of many well known modeling languages. Both languages can be used to model workflows as Wohed et al. state in [8, 11]. UML activity diagrams and BPMN both support most of van der Aalst's workflow control-flow patterns [12]. To be able to examine common modeling languages regarding their abilities to specify data-flows or resource-related aspects, Russel et al. in [13] present 43 workflow resource patterns. Based on these 43 patterns they point out that both UML [8] and BPMN [14] are not sophisticated enough to allow a detailed modeling of data-flows or resource-related aspects like the interaction of a resource "human" and the data which is manipulated by a human. Although the usage of constructs like pools, partitions or lanes allow for an allocation of resources to actions within UML and BPMN, no further thorough resource-modeling of a role-model or escalation pattern for instance is possible.

To enable a complete modeling of workflows with human tasks in BPMN or UML, several approaches to enrich the languages' metamodels have been presented. Kalnins und Vitolins [15] propose a comprehensive UML profile, which allows for modeling resources and human tasks. Therefore they extend the stereotype *CallBehaviourAction* by a new stereotype *CallHumanTask* and add a partition called *Performer*. The partition's "represents" attribute references a class with stereotype *OrgUnit* or *Position*. Although explicit modeling of resources is possible, no improvement supporting the workflow resource patterns is achieved.

Großkopf presents in [16] an extension of BPMN via its future metamodel BPDM (Business Process Definition Metamodel) aiming for a better support of the resource-perspective in BPMN. He extends activities by three further attributes and an association "assists" between activity and actor (resource). With this extension a better support of workflow resource patterns is given but as BPDM is still a proposal no tool support for this extension is available.

To execute workflows in a service-oriented fashion, within SOA mostly Web service compositions are used [26]. The prominent execution language for these compositions is the Business Process Execution Language (BPEL). BPEL focuses consciously on the execution of workflows without human interaction. Since the aspect of human interaction is still very important for most workflows, several vendors like IBM or Oracle include proprietary BPEL elements in their execution platforms to support human tasks. Using these proprietary elements, workflows specified with BPEL lose their interoperability and portability and cannot be deployed on another vendor's BPEL execution platform any longer. Facing this problem, IBM and SAP released a joint white paper named BPEL4People [1]. Meanwhile two separate specifications BPEL4People [18] and Web Service Human Task (WS-HumanTask) [19] have been released for standardization by the Organization for the Advancement of Structured Information Standards (OASIS). While BPEL4People addresses integration of human tasks to workflows using the new "PeopleActivity", WS-HumanTask is independent from BPEL. It on the one hand provides XML syntax for modeling human tasks and notifications and on the other hand an API for accessing human task instances from a client or the lifecycle of newly created task instances. An evaluation of BPEL4People and WS-HumanTask conducted by Russel and van der Aalst in [20] using their Workflow Resource Patterns shows a fair support.

A different approach to use human interactions in BPEL processes is provided by Thomas, Parci et al. [2]. They use the concepts described in the [1], provide XML syntax to define human tasks and integrate them into BPEL processes. As stated in this article, their approach requires an architectural extension with a component “People Activity Manager”, which coordinates task instances

3. Model-driven development of human tasks for workflows

3.1 Motivating example

In this section we motivate, how a modeling of human tasks can be supported and what benefit for software development yields from this approach. Since our approach makes use of MDA concepts, we use UML as recommended modeling language. A simple example workflow serves as case study: In the context of a university, a student’s registration for an exam should be processed electronically to speed up the registration process. The workflow “Process Examination Registration” is depicted by the UML activity diagram in figure 1. Developed during analysis phase, this diagram, compassing the following activities, is the starting point of the software development process [21]:

1. The student’s registration is pre-validated by a system to check if the student has all necessary prerequisites like other exams etc.
2. A staff member re-validates the registration regarding criteria which the system can not technically validate.
3. The registration is stored for further processing.
4. A denial / confirmation email is send.

The second activity “Check Registration” is a human task [7]. It has to be performed by a human person in the role “Staff Member”. Regarding this human task, there are much more details which should be captured during the analysis phase in a formal way. For instance, who is in charge of this human task, to whom it is to be escalated to if a staff member does not validate a registration after a certain period of time etc. Anyhow, existing UML modeling elements are not adequate to capture these details [8]. Currently, they have to be captured in some informal way, for instance on a sheet of paper or in a text file. Hence, escalation steps, notification paths etc. have to be configured in error-prone and expensive manual steps during the implementation phase. To be adaptable to domain specific needs, UML provides a lightweight extension mechanism called UML profiles [10]. With UML profiles the UML’s metamodel can be extended to specify new stereotypes needed for a certain purpose. Consequently we present an UML profile named *Human Task Profile* which allows modeling of human interaction aspects.

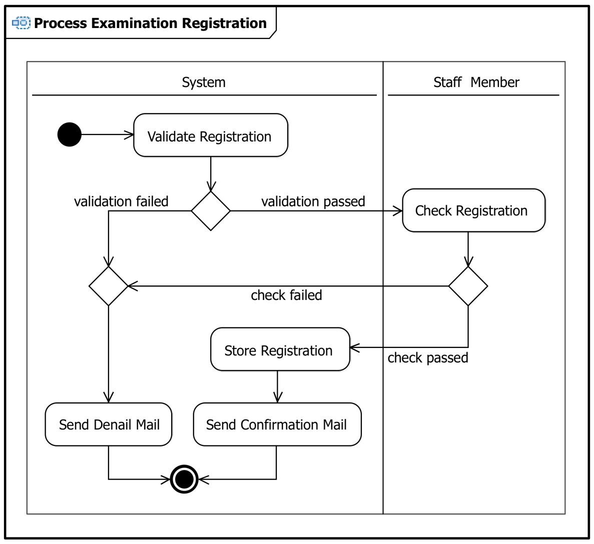


Figure 1. Exemplary workflow

3.2 UML profile for human tasks

We decided to build our profile on the UML use case diagram, as it is designed for modeling human-system interaction. Most of the stereotypes of a UML use case diagram like *Use Case* or *Actor* representing a role have only few attributes which can be specified. To be able to be more precise in modeling human tasks, first of all the stereotype *Use Case* is extended by three additional stereotypes: *Task*, *Notification* and *Reassignment* (cf. figure 2). The human task “Check Registration” can then be specified as *Use Case* of stereotype *Task*. A *Task* has additional attributes like “delegation” to specify to whom the task is delegated, if not processed correctly. Stereotype *Notification* can be used to model a message e.g. via e-mail. With stereotype *Reassignment* it is possible to model an escalation path. For instance, if a human task is not executed within two days by the role “Staff Member”, it may be reassigned to the user “Supervisor”.

Therefore, roles like “Staff Member” representing one or several persons have to be assigned to the human task. The use case diagram knows only one stereotype *Actor* with no further possibility to refine this stereotype. However, enterprises usually follow a more complex organizational structure. Thus the stereotype *Actor* is extended by an abstract stereotype *OrgEntity* and *OrgEntity* by *User, Role* and *Group.* If there is a completely different organizational structure, the stereotype *Query* allowing specifying expressions for any kind of user directory can be used.

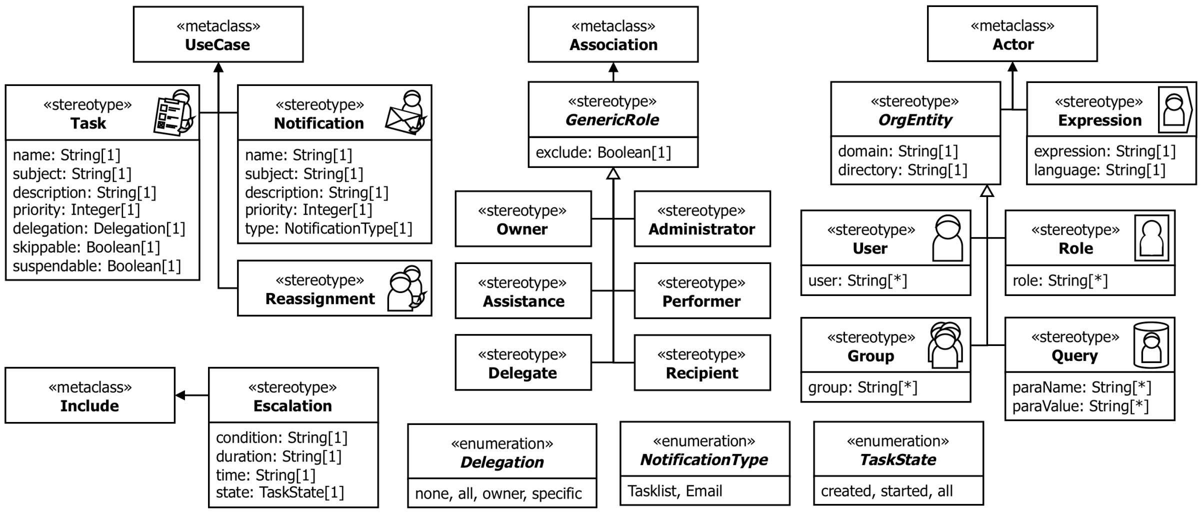


Figure 2. Human task profile with custom shape images

Additionally it is necessary to assign *Task* to *Role,* thus an *Association* is needed. It expresses whether a role is the owner of the human task, the supervisor etc. The stereotype *Association* is therefore extended by an abstract *GenericRole* and several concrete stereotypes, as figure 2 displays. Stereotype *Owner* is used to specify the role in charge, *Recipient* is used to model a user, which has to be informed, if the state of a human task is changed e.g. from "active" to "complete".

To be able to use all new stereotypes for a precise modeling during analysis phase, a one-time setup of the *Human Task Profile* in a development tool is necessary. With modern development tools like IBM's Rational Software Architect (RSA) [22], the implementation of a UML profile is straightforward. Additionally, these tools allow for adding one's own shape images to new stereotypes simplifying the usage and distinction of all new stereotypes. Having implemented the *Human Task Profile*, it can be used in any software development project to specify human tasks in a more precise manner. Figure 3 depicts the new *Human Task Diagram* developed with RSA. It can be easily understood by business analysts and other stakeholders. In particular, it can be modeled during the analysis phase without any technical expertise of the target platform.

An additional benefit of the *Human Task Profile* comes with a better support of the 43 workflow patterns [13] (cf. section 2). Unlike with plain UML, an additional 8 workflow patterns like the "Escalation" pattern as figure 3 shows can now be modeled.

The *Human Task Diagram* is one result of the analysis phase. It does not contain or refer to any technical or platform-related details. Hence it could be implemented in Java, .NET etc. and deployed on different software architectures like a SOA for instance. In terms of MDA, a *Human Task Diagram* is a Platform Independent Model (PIM). As there are similar escalation steps or notification paths for different human tasks, a *Human Task Diagram* can be used as a template and reused many times, saving configuration and development effort. Having captured all available information, this PIM has to be transformed to a Platform Specific Model (PSM) and enriched with more technical details. This is done during the following design phase by a domain expert.

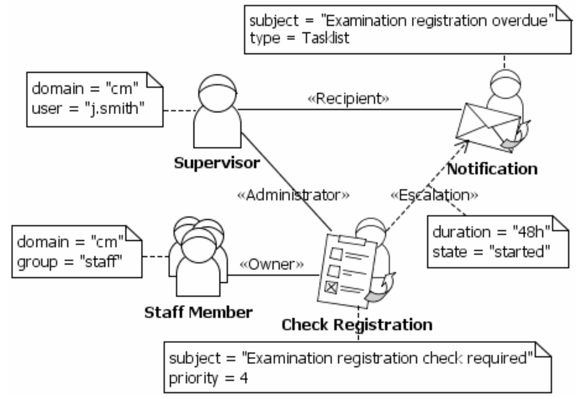


Figure 3: Human Task Diagram

3.3 Transformation to source code

With MDA, transformations need a source model and a platform model in order to create a specific target model. The source and the target model are instances of corresponding metamodels, as a *Human Task Diagram* is an instance of the UML metamodel extended by the *Human Task Profile*. To transform a *Human Task Diagram* to a human task expression language, the metamodel for this language is needed. Sticking to our development tool RSA, we develop the PSM's metamodel with the Eclipse Modeling Framework (EMF) [23]. For XML-based languages, an Ecore model can automatically be generated from an appropriate XML schema. The corresponding EMF model is an instance of an Ecore model, which is compatible with the Essential Meta Object Facility (EMOF) [24]. Therefore, our approach is compliant to MDA, which suggests MOF on the metameta layer.

The presented model-driven approach can be used to execute transformations to any kind of expression language for human tasks. Yet, as pointed out in section 2, there is no standardized human task expression language available. Web Service Human Task (WS-HumanTask) [19] is one promising candidate providing XML syntax based on [1]. It can be used to capture all details of human tasks like the assigned roles, the state of a human task etc. Whether the overall business process is defined in BPEL or any other language is of no concern to WS-HumanTask. Thus, a portable and interoperable specification of human tasks is possible. WS-HumanTask requires, as its name implies, the presence of a Web service based architecture, as presented in section 4. Consequently we use WS-HumanTask as PSM. Figure 4 provides an overview to all used models and transformations.

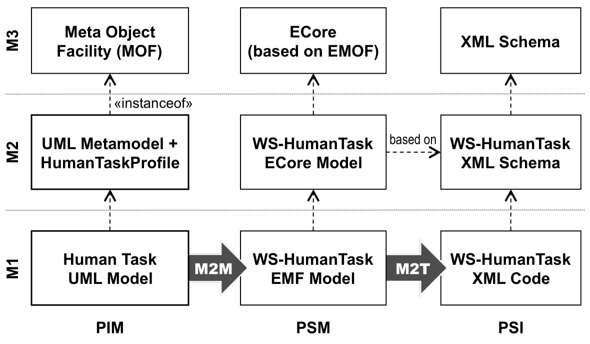


Figure 4: Models and Transformations

Finally, the necessary transformation rules for the PIM-to-PSM transformation have to be implemented. Note that this setup has, as implementing the *Human Task Profile,* only to be done once. Using the Rational Transformation Framework (RTF) provided with RSA, the transformations are written in plain Java and packed together as an Eclipse plug-in. From the MDA viewpoint, the use of a special transformation language like QVT [28] might be more adequate. However, by using RTF, the transformation itself can be used as an extension to the already existing UML-to-SOA transformation provided by IBM [27]. Therefore, we achieve a transformation from UML to e.g. BPEL and with our extension additionally to WS-HumanTask. The transformation rules transfer all model elements from the PIM like the *Group* "Staff Member" or the *Task* "Check Registration" (cf. figure 3) to the PSM. The following exemplary transformation rule transforms the stereotype *Role* to an EMF model using the WS-HumanTask Ecore model as metamodel.

```
Actor src = (Actor) context.getSource();
TGenericHumanRole target = (TGenericHumanRole)
   context.getTargetContainer();

TFrom tFrom = fac.createTFrom();
TGrouplist tGrouplist = fac.createTGrouplist();
TLiteral tLiteral = fac.createTLiteral();

Stereotype st = src.getAppliedStereotype(
   "HumanTaskProfile::Group");
List groups = (List) src.getValue(st, "group");

tGrouplist.getGroup().addAll(groups);
tLiteral.getMixed().add(
   pac.getDocumentRoot_Groups(), tGrouplist);
tFrom.setLiteral(literal);
target.setFrom(from);
```

Having executed the transformation, a domain expert is able to add further platform-related details to the PSM. This enrichment is done on the modeling layer, so there is no need to do any implementation work in the source code. After the domain expert has finished her work, the next phase of the software development process can be started. Since this is the implementation phase [21], usually there would be a lot of implementation work to do, especially concerning human tasks. However, the implementation effort is reduced to serializing the EMF Model to XML. This equals the transformation from PSM to the Platform Specific Implementation (PSI) and results in WS-HumanTask XML source code, representing all details concerning human tasks. Figure 6 depicts our approach at a glance.

Currently no WS-HumanTask reference implementation is available so far. To get our case study up and running, we implemented another set of transformation rules to IBM's task expression language [17]. This language is similar to WS-HumanTask and supported by IBM's Process Server [29].

4. Extended Service-Oriented Architecture

In section 3 a model-driven approach to develop human tasks for workflows has been presented. To be able to deploy the corresponding process specification on an execution environment, the underlying software architecture has to support the execution of human tasks in workflows. SOA for instance needs besides a process engine that executes workflows in terms of

Web service orchestrations, an additional component to manage human tasks during runtime [3, 26]. Further, this "task manager" component has to be used in a service-oriented manner. Consequently, it needs to provide two interfaces: One for the process engine calling the task manager to e.g. create new instances of human tasks. A second one is needed by a presentation component like, in the context of SOA, a Web-based portal [26]. With this second interface, the Web portal is able to request a user's current task list or to control the processing of a human task instance.

Additionally, the task manager has to map human tasks to different organizational structures as discussed in section 3.A. To resolve abstract roles associated to human tasks as modeled with the *Human Task Diagram* (cf. figure 3), a mapping to concrete persons like "Jon Smith" has to be done. Therefore, the task manager invokes a Web service interface to an identity manager Web service as presented in [25]. With both new components in place, the human task specifications can be deployed on SOA and the corresponding workflows can be executed. Based on an abstract model of SOA as presented in [26], two additional components, the task manager and identity manager are presented in figure 5.

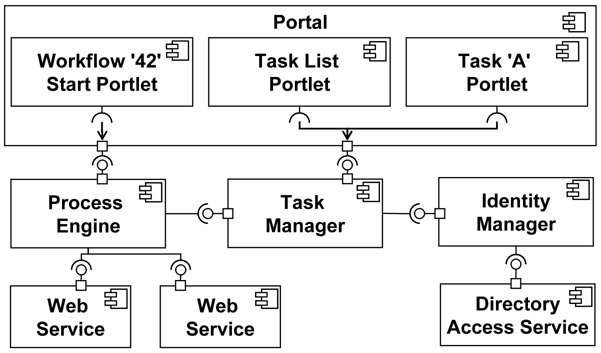
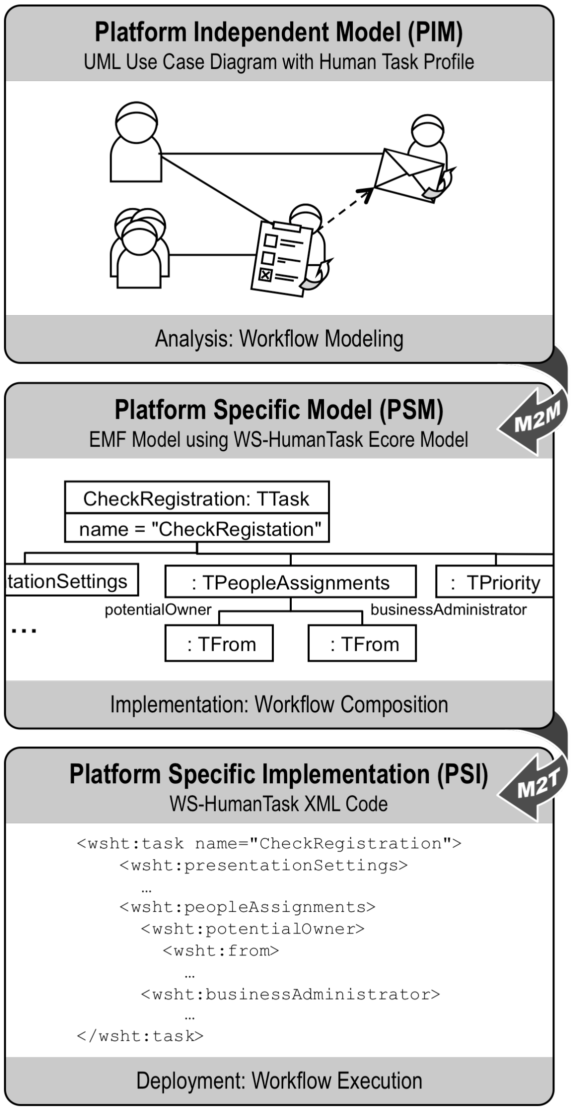


Figure 5. Extended SOA for human tasks

Using WS-HumanTask with an extended SOA has two major advantages. First, WS-HumanTask allows for a service-oriented specification and execution of human tasks according to the SOA approach. It does not concern about any platform or technical details. Therefore, it follows similar goals as SOA, like overcoming the present heterogeneity of IT or achieving platform independence. For instance, as long as the process engine uses WS-HumanTask to invoke the task manager, the internals of the process engine are of no importance. Thus, a flexible IT support is achieved.

Second, using SOA as execution environment for workflows additionally helps reducing the present complexity of IT support as stated at the beginning. With SOA a consistent integration of existing functional components is possible.


Platform Independent Model (PIM)
UML Use Case Diagram with Human Task Profile
Analysis: Workflow Modeling
M2M
Platform Specific Model (PSM)
EMF Model using WS-HumanTask Ecore Model
CheckRegistration: TTask
name = "CheckRegistation"
tationSettings
: TPeopleAssignments
: TPriority
potentialOwner
businessAdministrator
...
: TFrom
: TFrom
Implementation: Workflow Composition
M2T
Platform Specific Implementation (PSI)
WS-HumanTask XML Code
Deployment: Workflow Execution

```
<wsht:task name="CheckRegistration">
    <wsht:presentationSettings>
      …
    <wsht:peopleAssignments>
      <wsht:potentialOwner>
        <wsht:from>
          …
      <wsht:businessAdministrator>
          …
</wsht:task>
```

Figure 6. Case study at a glance

5. Conclusion and Future Work

In this article we were able to demonstrate that a model-driven development of human tasks for workflows is feasible and applicable to common software development processes. Using our UML profile named *Human Task Profile*, a tool-supported modeling of human tasks is possible. Furthermore, the developed models can automatically be transformed to source code of any desired platform. Capturing human tasks throughout a modeling process leads to a reduced development effort and an increased software quality. Additionally, the *Human Task Profile* allows a better support of Russel's common workflow patterns [13] unlike UML without the profile. We followed a software development process and demonstrated the benefit of specifying all details of human tasks with models. In the analysis phase the business analyst is now able to capture many details without any regard to technical or platform-specific details. Thus, he can focus on optimizing the workflows and properly specifying human tasks. In a following design phase, a domain expert may add technical details without regard to the overall business process. The implementation phase is reduced to an execution of two transformation processes from PIM to PSM and PSM to PSI.

Besides modeling human tasks, there are further requirements to support human interaction. If a human has to interact with an IT system, a user interface is needed. The development of a graphical user interface for instance in correlation with the abstract description of a human task, as given in the *Human Task Diagram*, has not been examined by this article. An approach to a model-driven development of graphical user interfaces can be found in [7]. To achieve a complete model-driven development of human interaction in workflows, we will investigate a combination of both approaches as our next step.

6. References

[1] M. Kloppmann, D. Koenig et al.: WS-BPEL Extension for People. Joint White Paper by IBM and SAP, July 2005
[2] J. Thomas, F. Paci et al.: User Tasks and Access Control over Web Services, IEEE International Conference on Web Services, 2007
[3] T. Stahl, M. Völter: Modellgetriebene Software-entwicklung, 1st edition, dpunkt Verlag, 2005.
[4] J. Mukerji and J. Miller: "MDA Guide Version 1.0.1," OMG, 2003.
[5] B. Hailpern and P. Tarr, "Model-driven development: The good, the bad, and the ugly," IBM Systems Journal, vol. 45, no. 3, 2006.
[6] G. Cernosek and E. Naiburg: "The Value of Modeling," IBM Developerworks, June 2004.
[7] S. Link, T. Schuster, et al.: "Focusing Graphical User Interfaces in Mo-del-Driven Software Development". Proc. of 1st IEEE Conference on Advances in Computer Human Interaction, Martinique, February 2008
[8] N. Russell, W.M.P. van der Aalst et al.: On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling, 2006.
[9] Unified Modeling Language (UML), Version 2.1.1: Superstructure, OMG Standard, 2007.
[10] Object Management Group: Business Process Modeling Notation Specification. OMG Final Adopted Specification, February 2006.
[11] P. Wohed, W. van der Aalst et al.: Pattern-based Analysis of the Control-flow Perspective of UML Activity Diagrams, 2004
[12] W. van der Aalst, A. ter Hofstede et al.: Workflow Patterns. Distributed and Parallel Databases, 14(3), S. 5-51, July 2003
[13] N. Russell, A. ter Hofstede et al.: Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, 2004.
[14] P. Wohed, W. van der Aalst et al.: On the Suitability of BPMN for Business Process Modelling. 4th International Conference on Business Process Management (BPM), September 2006
[15] A. Kalnins, V. Vitolins: Use of UML and Model Transformations for Workflow Process Definition. Communications of the 7th International Baltic Conference on Databases and Information Systems, 2006.
[16] A. Großkopf: An Extended Resource Information Layer for BPMN, Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam, 2007.
[17] M. Keen, O. Bahy et al.: Human-Centric Business Process Management with WebSphere Process Server V6, IBM Redbook, October 2006.
[18] A. Agrawal, M. Amend et al.: WS-BPEL Extension for People (BPEL4People), version 1.0, 2007.
[19] A. Agrawal, M. Amend et al.: Web Services Human Task (WS-HumanTask), version 1.0, 2007.
[20] N. Russell, W. van der Aalst: Evaluation of the BPEL4People and WS-HumanTask Extensions to WS-BPEL 2.0 using the Workflow Resource Patterns. BPM Center Report BPM-07-10, 2007.
[21] I. Sommerville: Software Engineering. 7th edition, Pearson Education Limited, 2004.
[22] U. Wahli, L. Ackermann et al.: Building SOA Solutions Using the Rational SDP, IBM Redbook, April 2007.
[23] B. Moore, D. Dean et al.: Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework. IBM Redbook, February 2004.
[24] Object Management Group: Meta Object Facility (MOF) Core Specification. OMG Available Specification, Version 2.0, January 2006.
[25] C. Emig, F. Brandt et al.: Identity as a Service - Towards a Service-Oriented Identity Management Architecture, 13th EUNICE Open European Summer School and IFIP TC6.6, Twente / Netherlands, July 2007.
[26] D. Liebhart, SOA goes real, Hanser Verlag, 1st edition, 2007, ISBN 978-3-446-41088-6
[27] D. Gorelik. Transformation to SOA: Part 3. UML to SOA. IBM Developer Works, January 2008
[28] Meta Object Facility (MOF) 2.0 Query / View / Transformation (QVT) Specification, Final Adopted Specification, OMG Standard, 2005.
[29] IBM WebSphere Process Server, http://www-306.ibm.com/software/integration/wps/