A Model-Driven Development Approach Focusing Human Interaction

Stefan Link, Tilmann Kopp, Sebastian Abeck
Cooperation & Management, Universität Karlsruhe (TH), Germany
{ link | kopp | abeck } @ cm-tm.uka.de

Abstract

Human interaction like entering some data, making decisions etc. has to be dealt with as an integral part of today's business processes and the supporting IT likewise. Hence, human interaction leads to an increased complexity in software development and software systems. Current business- and model-driven development approaches provide promising means to deal with this complexity. Diverse aspects of the business process and the supporting software system are captured in models and automatically transformed to the source code of a desired platform. In the context of human interaction however, there remains a lack of precise models for specifying human interaction aspects. Thus, an extensive manual development and configuration effort is necessary leading to expensive software, badly configured interfaces and frustrated users. In this article, we therefore demonstrate a model-driven development approach focusing on human interaction as an integral part of business processes and software systems likewise. A case study fortifies the applicability of our approach.

1. Introduction

Enterprises are forced to improve their business processes continuously due to fast changing markets [1]. Supporting business processes with Information Technology (IT) allows for increased execution efficiency. Business processes that can be completely supported by IT are the focus of this article. Following the definition of the Workflow Management Coalition (WfMC), this article in short refers to such business processes as workflows [2]. As workflows may also comprise human tasks like entering some data, their development and that of their supporting software systems gains new complexity [1]. Hence, existing process models in software development have to be improved to be able to address all aspects of human interaction as an integral part of the development process [3]. These aspects are manifold. As workflows are intended to increase the enterprise's profit, they have to be the starting point of a modern, business-driven development process [4]. If the workflow contains human tasks, at least a user interface is needed to enable the human user to interact. Additionally, human tasks have to be controlled to ensure their proper execution within workflow instances. In case the user who is supposed to execute a human task is not available, the task has to be dispatched to another user. Such organizational requirements have to be considered within the development process as well [5].

Currently, the integration of humans into workflows is usually accompanied by a large amount of manual development and configuration work. Thus, the developed software is expensive to maintain and not adaptable to meet the requirements of fast changing workflows [1]. New model-driven approaches to software development arise to increase software quality and reduce development effort. With Model-Driven Architecture (MDA) [6] for instance, the modeling of a software system is considered as a major development activity. The model provides on the one hand a better overview of the whole software system [7]. On the other, it can be used as a platform-independent source for automatic transformations to the point of different source codes for different platforms [8]. Hence, the platform-independent perspective is of great importance especially to human interaction aspects as there are many different terminals like PCs, PDAs and cell phones providing different frameworks as JEE or .NET etc. With the MDA approach, a high flexibility in software development, shortened development cycles and an increased software quality can be achieved [11]. Nevertheless, in particular in the context of human interaction the full power of MDA cannot be applied due to insufficient modeling languages [1, 12]. Many manual and error-prone development and configuration steps remaining as important details concerning human interaction have to be captured in an

informal manner or even worse can not be captured at all [13]. In addition to those model-driven principles, it is crucial to focus on a business-driven development in order to achieve a flexible and adaptable support of workflows [1]. Workflows which are comprised of several organizational units, humans and software systems make heavy demands on the development process. Therefore, we will demonstrate a model-driven approach focusing on human interaction. Based on our latest research results presented in [5, 11], our continuous approach yields two major benefits:

- Workflows are the starting point for a flexible, model- and business-driven software development approach where human interaction aspects like graphical user interfaces (GUI) or organizational requirements are considered as an integral part of the development approach.
- A model-based sketch of the GUI can be presented to the customer very early in the development process to ensure requirements are met.

Accordingly, this paper is organized as follows: section 2 introduces the state of the art of model-driven development in the context of human interaction. In section 3, we present an extension to the Unified Modeling Language (UML) [14] as recommended modeling language by the Object Management Group (OMG) for MDA, which enables the modeling of complex human interaction aspects within a continuous development process. Consistently referring to a small case study, we demonstrate the applicability of our approach. A conclusion and outlook on future work in this area closes the body of this paper.

2. Related work

According to Wohed and Russel, the Unified Modeling Language (UML) and the Business Process Modeling Notation (BPMN) [15] are just two of many modeling languages which can be used to model workflows [13, 16]. However, neither of them is sophisticated enough to allow a detailed specification like for instance the structure or navigational aspects of a GUI [17, 18]. Likewise, no further modeling of resources or escalation chains is possible. To overcome the shortcomings of existing modeling languages and to support a complete modeling of human interaction, especially in the context of Web Engineering, several approaches have been presented. UML-based Web Engineering (UWE) [19] for instance follows a systematic model-driven process to develop Web applications based on UML. Thereby UWE focuses on internal lightweight workflows [20] and does not address specifying relations to other applications or organizational structures.

In [21], Ceri et al. present the Web Modeling Language (WebML) as an approach to model-driven development of Web applications. The former data-centered approach is extended in [22, 23] to support a business-driven approach that conforms to MDA specifications. Especially the concept of using different models to specify a GUI is adopted by our own approach. Yet, UWE and WebML focus only on Web-based applications with simple GUIs and do not provide the means for an early specification of GUI sketches to test if the customer's requirements are met.

Paterno presents in [24] an approach to a task-based generation of user interfaces. He emphasizes that, due to manifold platforms, a user-centered development approach has to be considered. We agree with his general approach but focus on more high-level specifications for workflows.

Sukaviriya et al. [1] follow Paterno's approach, but start with high-level business processes using a special modeling language. They additionally provide an UML extension to achieve a detailed UI Extension Model and sketch a tool to allow different views of human interaction aspects [1]. Additionally, they discuss a way to transform their model to a platform-independent and -specific UI model specified with XML. We agree with most of their findings but investigate a more general approach not starting with a specialized modeling language for workflow modeling. We further aim to integrate the customer as a source of crucial information about human interaction aspects even earlier into the development process and to provide a GUI based on developed models.

The Wisdom approach of Nunes and Cunha distinguishes between an application-oriented and an interaction-oriented view [25]. Using a UML profile, they manage to provide the important differentiation of interaction aspects and application aspects. Nunes and Campos continue research towards a User-Centered Design approach [12]. Their UI Design tool allows a platform-independent specification of abstract and concrete graphical user interfaces, yet without any relation to a business process perspective.

3. Model-driven development approach focusing human interaction

Based on MDA principles, we are going to discuss our model-driven approach focusing on human interaction in this section. Starting from a workflow model

we demonstrate, how most of the needed information about human interaction aspects can be captured with models and therefore can be transformed to the source code of the desired platform automatically. Consequently, we do not present manually sketched low-fidelity mock-ups [1] as a prototype of the GUI to the customer in order to capture his/her requirements. Instead, we use UML to provide the model elements to specify essential details of the GUI. Thus, it is possible to present an automatically generated prototype to the customer in a very early stage of the development process as a basis for discussion and further refinement. Additionally, we consider the organizational requirements of a human task, like who is in charge of a human task, as another essential part of human interaction. Therefore, we present the means to capture these requirements with models reducing the configuration effort. To demonstrate the applicability of our approach in a common software development project, we present a simple workflow that serves as case study next.

3.1 A motivating example

In our university context, a student's registration for an exam is to be supported by IT to speed up the registration process. In a first discussion with our customer during the analysis phase of our software development project, the corresponding workflow "Process Examination Registration" is sketched as follows:

1. The student's registration has to be pre-validated by a system checking if the student has all prerequisites like other exams etc.
2. The teacher conducting the examination is requested to suggest a date for the examination.
3. The examination date has to be confirmed by an proctor who is asked to attend the examination.
4. If teacher and proctor agree to a date, a confirmation e-mail is sent by a system to the student.

Figure 1 depicts the workflow by a UML activity diagram. The second action "Choose Examination Date" for instance has to be performed by a human in the role "teacher". Therefore, this action is a human action [5] and needs a GUI to enable the teacher to perform the task. Additionally, it is necessary to ensure the task's proper execution. If for example a certain teacher is not available for a certain period due to illness or vacation, the student's request has to be escalated automatically to a supervisor. We will specify the GUI and the organizational requirements of "Choose Examination Date" pertaining to this case study.

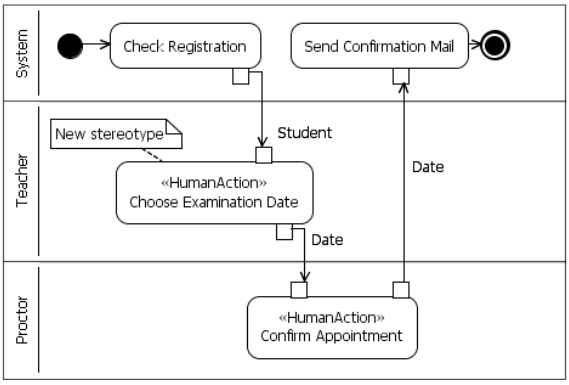

Figure 1. Extended exemplary workflow

3.2 Improving the workflow model

In the context of model-driven development, common modeling languages such as UML do not provide sufficient model elements to specify human interaction aspects [1, 12]. For instance, if a transformation engine has to decide which of the actions in the exemplary workflow (figure 1) needs a GUI, usually it would not be able to identify these actions, as UML does not provide an appropriate model element. Hence, a more detailed differentiation is needed. To cope with such domain specific demands, UML provides a lightweight extension mechanism called UML profiles [14]. UML profiles are based on the existing UML metamodel and extend existing metaclasses like *Action* or *Class*. The benefit of using a UML profile comes with the reusability of the profile and the availability of domain-specific stereotypes. To be able to use UML profiles in development, these must be supported by the used tool. IBM's Rational Software Architect (RSA) [26] is one example of such a tool which was employed to support our model-driven development approach.

Our *Human Interaction Profile* depicted in part in figure 2 is such a UML profile. It provides additional stereotypes for specifying human interaction aspects as will be presented in the following. The first part of the *Human Interaction Profile* introduces three new stereotypes as specialized *Actions*: *System-, Human-* and *ManualAction*. This differentiation determines in a formal manner if an action has to be performed by the system only like pre-validating the student's data in "Check Registration" (*SystemAction*); if user and system interact like in "Choose Examination Date" (*HumanAction*); or if the user has to perform some task without the system like conducting a counselling interview (*ManualAction*). Furthermore, this differentiation allows an automatic decision if a GUI is needed

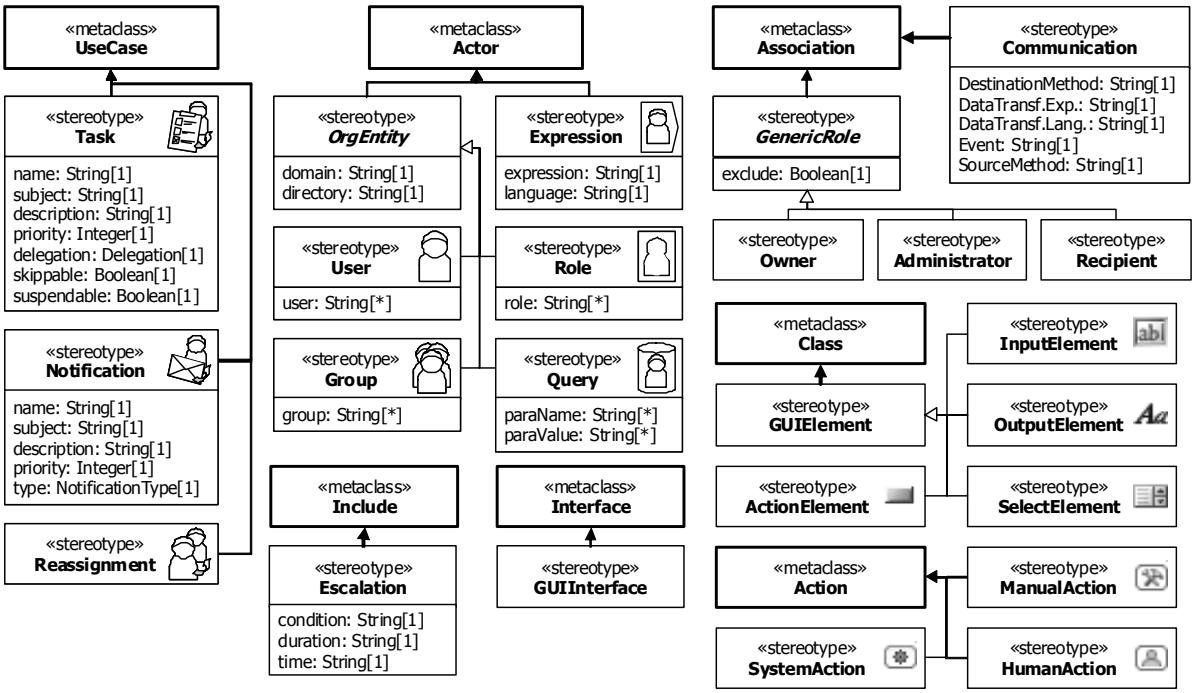

Figure 2. Human Interaction Profile with custom image shapes

(*HumanAction*) or not (*System-* and *ManualAction*). The corresponding stereotypes can be found on the left hand side of figure 2. With the *Human Interaction Profile* used in RSA, the workflow model can be refined and the specialized *Action HumanAction* as depicted in figure 1 is available. Note that the implementation of the *Human Interaction Profile* for RSA is only a setup. After its implementation, all stereotypes can be reused.

Besides the details about the workflow, the customer provides further information. Usually customers have a good understanding about what the GUI should look like and what actions they want to perform using it. Hence, the customer explains that the teacher has to be able to see the student's name, forename and matriculation number. The human with the role "proctor" though is only allowed to see the name. Thus we learn much about the GUI for each human task and discuss the application's business objects to some extent. Next this information could be manifested using some rapid prototyping tool sketching a GUI. Such a sketch would surely please the customer as they usually look good. However, the effort put into those sketches would be lost since they are not built on a model-driven basis. Again we use the *Human Interaction Profile*, capture GUI-related details with models and generate the corresponding GUI automatically. The needed stereotypes are explained in the following.

3.3 Modeling the GUI

As there are different aspects of a GUI to be taken into account, the specification of a GUI needs different types of models [1, 26]. First, a GUI enables a user to read or edit data in some way. Hence, a domain model is needed to specify the business objects that are edited by the system and the user. As mentioned above, the customer provides basic information about the domain model. Unlike other approaches, we split the business object into two parts: a GUI-related and a non GUI-related part. Since the customer is only concerned with the GUI-related part, which will be visible to him/her, we do not specify the complete domain model of our application in detail but focus on the GUI-related part first. To capture the GUI-related part of a business object, the *Human Interaction Profile* provides an extension to the UML interface via a stereotype named *GUIInterface.* A *GUIInterface* comprises all or some attributes of a business object the customer wants to be visible on a particular GUI of a particular human task. A *Class* implementing all associated *GUIInterfaces* represents the business object. Using a *GUIInterface* instead of a *Class* only has one major benefit. According to the case study, the teacher is allowed to see the student's name, forename and matriculation number but the proctor must not see the student's name or forename. Both users have access to different parts of

the same business object. Thus, two *GUIInterfaces* and a *Class* Student inheriting from both *GUIInterfaces* are modeled. Figure 3 illustrates the corresponding part of the domain model.

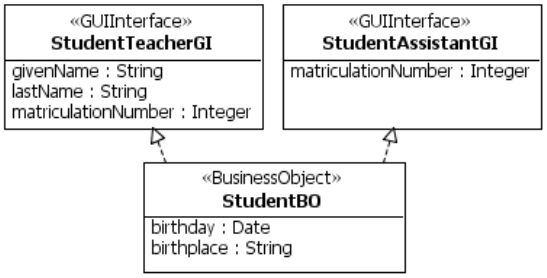

Figure 3. Domain model with GUIInterfaces

Having captured the GUI-related attributes of *HumanAction* "Choose Examination Date" in the domain model, the customer wants to have an impression what the GUI will look like. Using a model-to-model transformation, the structural model of the GUI represented by a UML class diagram and depicted in figure 4 is generated automatically from the workflow model and the domain model. As presented in [11], every GUI has a certain structure and the corresponding structural models of GUIs have similarities. On the uppermost level, there is a container element (e.g. a Web browser window) that holds *GUIElements*, which again may hold *GUIElements* and so on. On the lowest level, a GUI comprises *OutputElements* for displaying, *InputElements* for entering some data, *ActionElements* to perform an action etc. Providing all the *GUIElements* as stereotypes allows for specifying a great variety of GUIs without any platform-related details.

The workflow model depicts that the role teacher has to work with the two business objects "Student" and "Date" in *HumanAction* "Choose Examination Date". Hence, there has to be one top-level *GUIElement* for the *HumanAction* containing two other *GUIElements* for viewing the student's data and another one for choosing a date. Additionally, the GUI-relevant attributes for business object Student have already been captured.

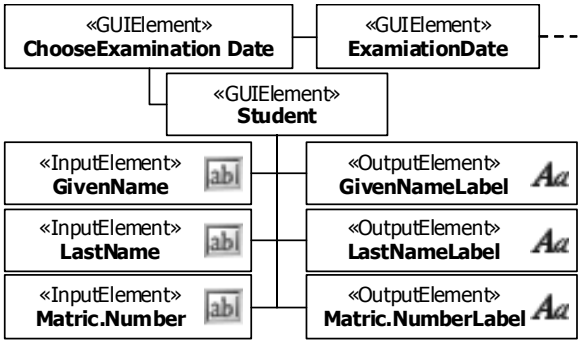

Figure 4. Extract of structural model

So far, all used models are platform-independent models (PIM) in terms of MDA. Therefore, the PIMs can be reused as the source for transformations to a great variety of frameworks and terminals. Assuming the customer wants to have a Web-based application, a next model-to-model transformation provides the corresponding platform-specific model (PSM) based on XForms, XHTML or any other language appropriate. Figure 5 presents the result of that transformation giving the customer a first impression of the GUI.

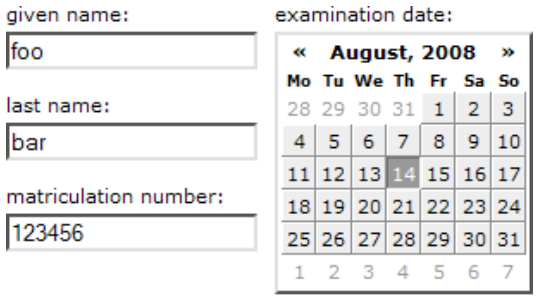

Figure 5. First impression of the GUI

If the customer now recognizes a missing input field like "field of study", an attribute is added in the corresponding *GUIInterfaces*, the transformations are run again and the new GUI is ready. As we focus only on the customer's wishes in the beginning, such a change causes no other effort like manipulating source code and so on. That way, the GUI to a human action can be specified very early in the development process until the customer is satisfied and all needed attributes are present. Having agreed on the basic structure of the workflow and GUIs, the refinement of the models can be started in the design phase.

3.4 Refinement of the GUI

The first impression of the GUI in figure 5 points out another aspect that can be modeled. Usually the calendar object on the right side of figure 5 allows picking a date which initializes a date object with a subject like "Examination of foo bar", a room and so on. Hence, if the teacher chooses August 14 as date of the examination for student "foo bar", it would be beneficial if the subject would be automatically set to "Examination foo bar". The needed information is already present and therefore should not be entered again. Thus, an association between the corresponding *GUIElements* is necessary to express their relation

between the name of the student and the subject of the date object. For modeling such an association, we extend the UML *Association* metaclass and provide a stereotype named *Communication* (cf. figure 6). Several properties allow for a detailed manual specification of a *Communication*. For instance, the *Event* is used to specify which event triggers the communication. *SourceMethod* and *DestinationMethod* are used to specify which operations send and receive data. If the data has to be adapted, attributes *DataTransformationLanguage* and *DataTransformationExpression* can be used to specify the corresponding transformation. Figure 6 provides an example of a communication association.

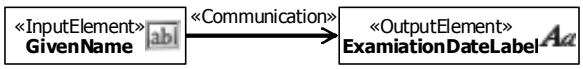

Figure 6. Communication example

The extended workflow model still contains additional information usable in a model-driven development as the next section demonstrates.

3.5 Specifying human tasks

Besides the GUI, monitoring the execution of a human task is another important aspect of human interaction. In our case study, the role teacher is in charge of performing the *HumanAction* "Choose Examination Date". In the analysis phase more details provided by the customer regarding this *HumanAction* should be captured in a formal way. For instance, if the human first assigned to perform the task of *HumanAction* "Choose Examination Date" does not choose an examination date after a certain period (maybe due to maybe illness or vacation), a reassignment of that task

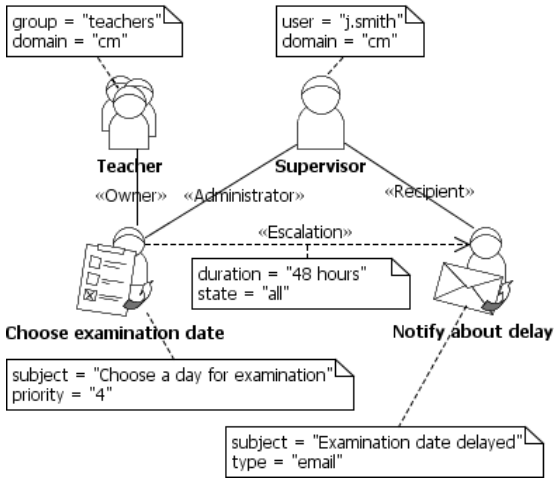

Figure 7. The organizational model

is necessary. To enable a modeling of such details, the *Human Interaction Profile* extends the UML *Use Case* with several stereotypes for usage in an organizational model. The UML stereotype *Use Case* is extended by three additional stereotypes *Task*, *Notification* and *Reassignment*. A *Task* like "Choose Examination Date" provides additional properties like *Priority* to force an order according to which *Tasks* have to be completed. Stereotype *Notification* can be used to model a message that is sent e.g. via e-mail in case of an event like an overdue *Task*. Additionally it is possible to specify an escalation chain with stereotype *Escalation*.

Based on these extensions, another model-to-model transformation uses the workflow model as the source model since it already contains all needed details. *HumanAction* "Choose Examination Date" is automatically transformed to *Task* "Choose Examination Date", *Activity Partition* "teacher" to *Group* teacher and so on. Assuming that a *Task* has to be monitored, this *Task* needs an *Escalation*, *Notification* and a *Supervisor* to be notified. Hence, most parts of the organizational model as depicted in figure 7 can be transformed from our workflow model by a model-to-model transformation. Concrete values, for instance that the *Task* "Choose Examination Date" is being escalated after 48 hours or that John Smith is the *Supervisor* of this *Task*, have to be added manually.

In terms of MDA, the organizational model is also a PIM as it refers to no platform-specific details. Certainly there are similar escalation steps for other human tasks. Having captured all available information, this PIM has to be transformed to a Platform Specific Model (PSM) and enriched with technical details. Web Service Human Task (WS-HumanTask) [9] is one promising candidate to base the PSM on. The XML syntax of WS-HumanTask 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.

Finally, the software, developed in a model-driven way has to be deployed on a software architecture. Due to fast changing workflows as mentioned in the beginning, not only the software development process has to be improved, the employed software architecture has to cope with flexibility and adjustability demands as well. As we discussed in [5], a service-oriented architecture [10] is the promising software architecture to cope with these requirements.

4. Conclusion and Outlook

In this paper, a model-driven development approach focusing on human interaction has been presented. Using the *Human Interaction Profile*, a tool-supported modeling of different human interaction aspects is possible. Starting with a workflow model as the central model for the development process, the provided stereotypes can be used to capture valuable information in a formal way. During the analysis phase, the customer provides valuable information about human interaction aspects that can readily be specified within the workflow models if the *Human Interaction Profile* is used. This allows an early iterative development of GUIs providing a good impression for both customer and developer about the later application.

Additionally, the *Human Interaction Profile* allows for specifying an organizational model. Instead of configuring the involved task management systems manually, the organizational model encompasses all required information. Again, most of the information needed to generate the organizational model is contained in the workflow model. The organizational model is automatically transformed into the corresponding language the task management system is able to interpret, as for instance WS-HumanTask [9].

Besides the need of an early customer integration into the development process especially allowing the capture human interaction requirements, it seems beneficial to monitor the execution of human tasks very closely. If, for instance, users need significantly more time to enter data to one GUI than to another, this might be a hint about a badly configured GUI or a missing help topic etc. Therefore we have already started to focus our research on key performance indicators for human interaction.

5. References

[1] N. Sukaviriya, V. Sinha et al.: "User-Centered Design and Business Process Modeling: Cross Road in Rapid Prototyping Tools", Interact, Brazil, October 2007

[2] Workflow Management Coalition: Terminology & Glossary, WfMC Specification, February 1999. http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf

[3] M. Kloppmann, D. Koenig et al.: "WS-BPEL Extension for People", Joint White Paper by IBM and SAP, July 2005

[4] T. Mitra: "Business-driven development", IBM developer works, 2005, http://www-128.ibm.com/developerworks/webservices/library/ws-bdd/

[5] S. Link, P. Hoyer et al.: "Model-Driven Development of Human Tasks for Workflows", Proc. of 3rd International Conference on Software Engineering Advances, Malta, 2008

[6] J. Mukerji and J. Miller: "MDA Guide Version 1.0.1," OMG, 2003, http://www.omg.org/docs/omg/03-06-01.pdf

[7] B. Hailpern and P. Tarr: "Model-driven development: The good, the bad, and the ugly," IBM Systems Journal, vol. 45, no. 3, 2006.

[8] T. Stahl and M. Völter: Modellgetriebene Softwareentwicklung, 1st edition, dpunkt edition, 2005.

[9] A. Agrawal, M. Amend et al.: Web Services Human Task (WS-HumanTask), version 1.0, 2007.

[10] D. Liebhart: SOA goes real, Hanser, 1st edition, 2007, ISBN 978-3-446-41088-6

[11] S. Link, T. Schuster et al.: "Focusing Graphical User Interfaces in Model-Driven Software Development", Proc. of 1st Conference on Advances in Computer Human Interaction, Martinique, 2008

[12] P.F. Campos and N.J. Nunes: "CanonSketch: a User-Centered Tool for Canonical Abstract Prototyping", Proc. of International Workshop on Design, Specification and Verification of Interactive Systems, 2005

[13] N. Russell, W.M.P. van der Aalst et al.: "On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling", Proc. of 3rd Asia-Pacific Conference on Conceptual Modelling, 2006

[14] Object Management Group: Unified Modeling Language, Version 2.1.1, OMG Standard, 2007, http://omg.org

[15] Object Management Group: Business Process Modeling Notation Specification, OMG Standard, 2006, http://omg.org

[16] P. Wohed, W. van der Aalst et al.: "On the Suitability of BPMN for Business Process Modelling", 4th International Conference on Business Process Management, 2006

[17] R. France and B. Rumpe: "Model-driven Development of Complex Software: A Research Roadmap", Future of Software Engineering, p.37-54, 2007

[18] R. Petrasch and O. Meimberg: "Model Driven Architecture – Eine praxisorientierte Einführung in die MDA", dpunkt edition, ISBN 3-89864-343-3, 2006

[19] N. Koch: "Software Engineering for Adaptive Hypermedia Systems Reference Model, Modeling Techniques and Development Process" Doctoral Thesis, Ludwig-Maximilians-Universität Munich, 2001

[20] N. Koch, A. Kraus et al.: "Modeling Web Business Processes with OO-H and UWE", 3rd Int. Workshop on Web-oriented Software Technology, 2003

[21] S. Ceri, P. Fraternali et al.: "Web Modeling Language (WebML) a modeling language for designing Web sites", Computer Networks Volume 33, 2000

[22] M. Brambilla, S. Ceri et al.: "Process Modeling in Web Applications", ACM Transactions on Software Engineering and Methodology Vol. 15, No. 4, 2006

[23] N. Moreno, P. Fraternalli et al.: "A UML 2.0 Profile for WebML Modeling" ACM Int. Conf. Proc. Vol. 155, 2006

[24] F. Paterno: "Tools for Task Modeling: Where we are, where we are headed", Proc. of International Workshop on Task Models and Diagrams for user interface design, 2002

[25] N.J. Nunes and J.F. Cunha: "Towards a UML Profile for Interaction Design: the Wisdom Approach", in UML 2000, LNCS, vol. 1939, Springer edition, Heidelberg 2000

[26] U. Wahli, L. Ackermann et al.: "Building SOA Solutions Using the Rational SDP", IBM Redbook, April 2007 http://www.redbooks.ibm.com/redbooks/pdfs/sg247356.pdf