UI design without a task modeling language – using BPMN and Diamodl for task modeling and dialog design

Hallvard Trætteberg

Associate Professor
Dept. of Computer and Information Sciences
Norwegian University of Science and Technology
hal@idi.ntnu.no

Abstract. In the field of model-based user interface design (MB-UID) task modeling is established as a necessary activity. However, in many (industrial) contexts, it is not realistic to introduce yet another modeling notation, particularly when user interface design is considered less important than overall process logic and system architecture. Therefore, it may make more sense to adapt existing process-oriented notations to task modeling, than vice versa (adapting task modeling languages to process modeling). This paper describes our experiences with using BPMN and Diamodl for process and task modeling and dialog design, respectively.

Keywords: User interface design, dialog modeling, business process management notation.

1 Introduction

Within the field of model-based user interface design (MB-UID), the standard design process includes task modeling, dialog modeling and concrete design 2. . Specialized modeling task and dialog modeling languages/notations have been developed for supporting the first two of these, while the latter involves mapping from dialog to either a concrete model or specific toolkit or runtime platform. Specialized languages are important, for at least two reasons: 1) they put focus on the specific information that an activity should result in, and 2) they enable better tool support by formalizing the relevant information. There is however a cost associated with introducing new notations in software development, as it adds to the already high complexity of modern development methods and tools.

An alternative approach is taking established (within the target industry) modeling languages as a starting point and augmenting the methods built around them, so the desired information still is captured, although in a different form. The advantage lies in lowering the cost of adopting the methods (hopefully below the threshold of adoption). In addition, we see a potential for coupling information in different models, i.e. there may be a synergy between the main usage of the notation and the new, augmented usage. In our work we have looked at how the Business Process Modeling Notation (BPMN) may be extended to cover tasks and augmented with extra

information concerning object life-cycle. The basic idea is that business processes and tasks are similar concepts at different levels of abstraction, and that the essential information from task analysis may captured by using BPMN in a different way, augmented with some extra information. As a bonus, the relation between the high-level processes and lower-level task structures becomes clearer and the gap between system logic and architectural and dialog structure and behavior, becomes smaller.

In the following sections we will review related work, describe the overall approach and outline a practical method for modeling and deployment of applications using BPMN 8. , Diamodl 10. and Eclipse-based tools.

2 Related work

In 6. , several task and process modeling languages are compared, to see how they may support model-based design of eServices in eGovernment applications. We have previously discussed the relation between process modeling and task modeling in 3. and more recently in 4. . Our focus on this paper is on a lean method based on a the standard process modeling language BPMN and Diamodl and deployment using standard, open-source tools and modern architecture. 7. also take a business process model (BPM) as a starting point, but uses a less formal UI model with a weaker coupling to the BPM. The goal of 5. is similar to ours, that of supporting server-side workflow with model-based UI client, but they do not use a standard workflow modeling notation.

3 Overall approach

In the prototypical MB-UID process, a task model is the starting point for developing a dialog model and subsequent concrete user interface design. The task model may be seen as capturing human behavior, the dialog model describes software behavior. The deployment of the UI will be a combination of concrete user interface elements and the software and models necessary for implementing the dialog behavior, like state machinery, data binding, etc. and the concrete interface describes what is actually deployed.

This is actually fairly similar to the standard approach of business process modeling using BPMN and execution and deployment using the Business Process Execution Language (BPEL) 9. . First, the behavior of the process, or rather the roles and systems taking part in the process, is described as communicating processes, activities and tasks in a BPMN diagram. This model is transformed to a BPEL model, which describes the software part of the (future) process, i.e. the (automation of) coordination (also called choreography and orchestration) aspects of the process and relies on web services for linking all the participants (people, processes and external services). The BPEL model is then deployed, together with other supporting software like business objects, web services, persistence etc.

As can be seen, the and overall approach and role of the models is similar, although they have the (group) system perspective instead of the (individual) UI

perspective. This more than suggests that the models can be related across the domains of business process management and user interface design, as illustrated in 0. According to this figure, process models (in BPMN) may be related to task models since they both capture the behavior of people, BPEL models may be related to dialog models, since they both model software for supporting people and BPEL and a deployed BPEL model executed by a server-side engine may interact with the client-side UI runtime. We are currently investigating how this may be more than analogy, i.e. we propose method whereby BPMN is used for both business process and task modeling and BPEL and diamodl are used for modeling software support and deployment on a SOA-based platform.

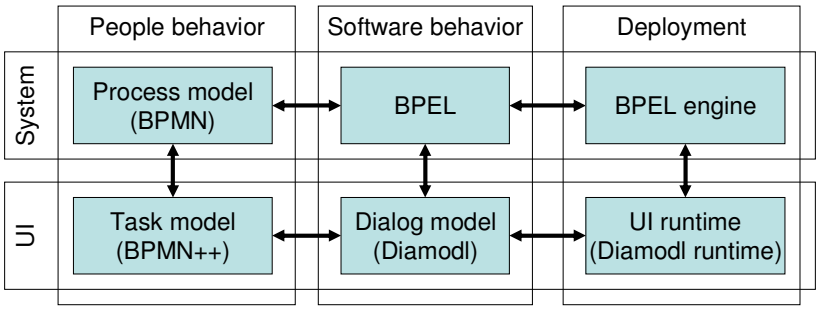

Fig. 1. Relationship between system and user interface domains

4 Using BPMN for task modeling

According to www.bpmn.org "… Business Process Modeling Notation (BPMN) will provide businesses with the capability of understanding their internal business procedures in a graphical notation…". Such a business procedure is a set of coordinated tasks performed by a set of roles and structured in hierarchy (called activity). Tasks in *different* processes communicate and implicitly coordinate by means of message connections. Tasks in the *same* process use flow connections for controlling sequencing and variables for storing XML data as process state. A task may repeat and be conditional. Web services are used for communicating with external systems, including business objects and UI clients.

A task modeling language typically structures tasks in a hierarchy. *Operators* are used for controlling the enablement and sequencing of tasks, e.g. tasks may be performed in sequence, in parallel, one of several tasks may be conditionally selected, a task (structure) may repeat, etc. Events from the environment, including objects representing the domain, may trigger or enable tasks, and operations may be performed on the environment.

The main difference between BPMN and task modeling languages is more a matter of style than expressive power and both essentially model a task hierarchy. Similarly, the control flow connections of BPMN and operators in task modeling languages are

visually different, but have essentially the same expressive power. Finally, messages may take the role of events, to model tasks that are triggered by changes in the environment.

The weakest point of BPMN is domain modeling and data. Due to its focus on process message exchange and integration of web services, XML schemas and XML data has been chosen as the data model. Fortunately, many tools for object-oriented modeling, e.g. EMF 1. , can generate XML schemas, serialize models as XML and in general interoperate with XML, so this is more a practical obstacle than a conceptual problem. E.g. although a variable cannot be declared to reference an object of a particular class, is can be declared to refer to an XML fragment that *represents* an object of a particular class.

What is still missing is a way of declaring pre-conditions and post-conditions in terms of objects and their life-cycle (creation and destruction) and state. E.g. a pre-condition for performing a review of an application is of course the application, and the post-condition is that the review has been created. Hence, we augment the BMPN "task" model with annotations on each task that makes these conditions explicit, not very different from how Use case diagrams are elaborated be means of structured text.

5 Step-by-step modeling method

Fig 1 shows the relationship between system and UI perspectives on the process of going from a process/task model to a deployed system which combines a BPEL engine and the Diamodl runtime. In this section we detail the practical method we propose for this process. The process is illustrated by a simple example, that of reviewing a request (for something) and returning the answer. As shown in fig 2, the Customer sends in an application that is received by our User role. The User performs a shallow review and may decide to either let the Expert role perform a deep review or do it himself. The resulting review is sent back to the Customer.

Creating this BPMN model is the first step in our method, combined with domain modeling, where concepts in the domain are formalized in a class diagram. In practice, the domain model may already exist, either from previous projects or as a reference model for a well-established domain, e.g. order management. Since BPMN is XML-centered, we need to be able to convert the domain model to an XML Schema, before annotating the connections between processes (and possibly internal variables) with XML types. We use Ecore, the Eclipse Modeling Framework's modeling language for domain modeling, and export the XML Schema from the Ecore editor. The Intalio Designer Eclipse application, which we use for BPMN modeling, allows us to open the XML Schema in the Process navigator and drag XML types into the connections in the diagram.

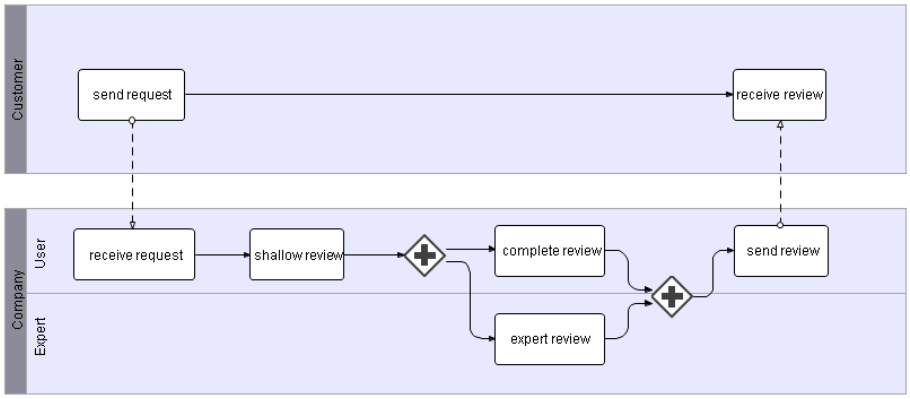

Fig. 2. Business process

This model is system centric, in that it does not focus on any particular user or distinguish between the user and the system. The next step in the method is disentangling the users' task from the system, as a kind of process *refactoring*. The general idea is to model the User role in a process of its own and make the connection (interface) to other roles and processes explicit. The refactored process model is shown in fig 3. As can be seen, this process interacts with both the Executable process, i.e. the system, and the Expert role.

This refactored process model is similar to a task model, in that it makes explicit what each uses does (task structure) and how it interacts with its environment (events and data). It may require further decomposition to be detailed enough, and in addition we annotate it with pre- and post-conditions that make explicit how domain data is operated on (life-cycle and states). E.g. the pre-condition for the User task "shallow review" is that there exist an unhandled request and the post-condition is that a review has been created and is in progress. This step may result in a refined domain model, to better capture the objects' possible states.

The connections flowing into and out of the User process, defines the necessary input and output of the user interface, and hence the dialog model, which is the next step. Our dialog modeling language Diamodl fits well with the dataflow nature of process models and web services. The connections are modeled as computations in Diamodl, the in-flowing connections become computations without input (sources of data), while out-flowing connections become computations with one input and no output (sinks of data).

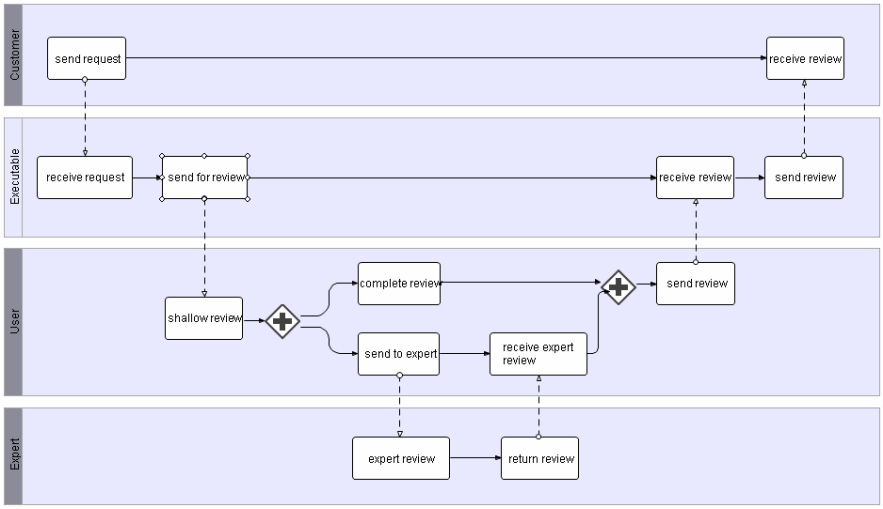

Fig. 3. Refactored process

Although the BPMN diagram is a model of how the user works, it is not a model of how the user works with the to-be-designed UI. In our experience, one of the main decisions to be made is how the user manages multiple and possibly parallel instances of the process. This possibility is implicit in the process model and if not considered in the design process, we may end up with a user interface that forces the user to work with each process instance independently. E.g. in this case, we should consider if the user should be able to see the finished review of one request while performing the shallow review of another, and perhaps support copying the former review.

Part of the dialog model and corresponding GUI prototype is shown in fig 4. The two large, shaded triangles are computations that represent connections from the process model, receiving a request and sending a review to the expert, respectively. This models lets the user see the list of unhandled requests, select and view one and choose to review the selected one. There is also a list of reviews in-progress, from which the user may select one and send to the expert. The GUI prototype has mostly been generated from the model, with only the layout and labels added by hand. The sample data that populates the GUI has been created with standard EMF tools, based on and validated against the domain model.

The last step is deployment, which in general will include the part of the BPMN process marked as executable, the GUI and dialog and supporting services like task and data management. As work in-progress, this is the weak part of the current tool chain. A valid (and executable) BPMN process fragment may be translated to BPEL code and deploying it on one of several open source BPEL engines, and Intalio Designer is able to generate and deploy to a standards-compliant server in a few clicks.

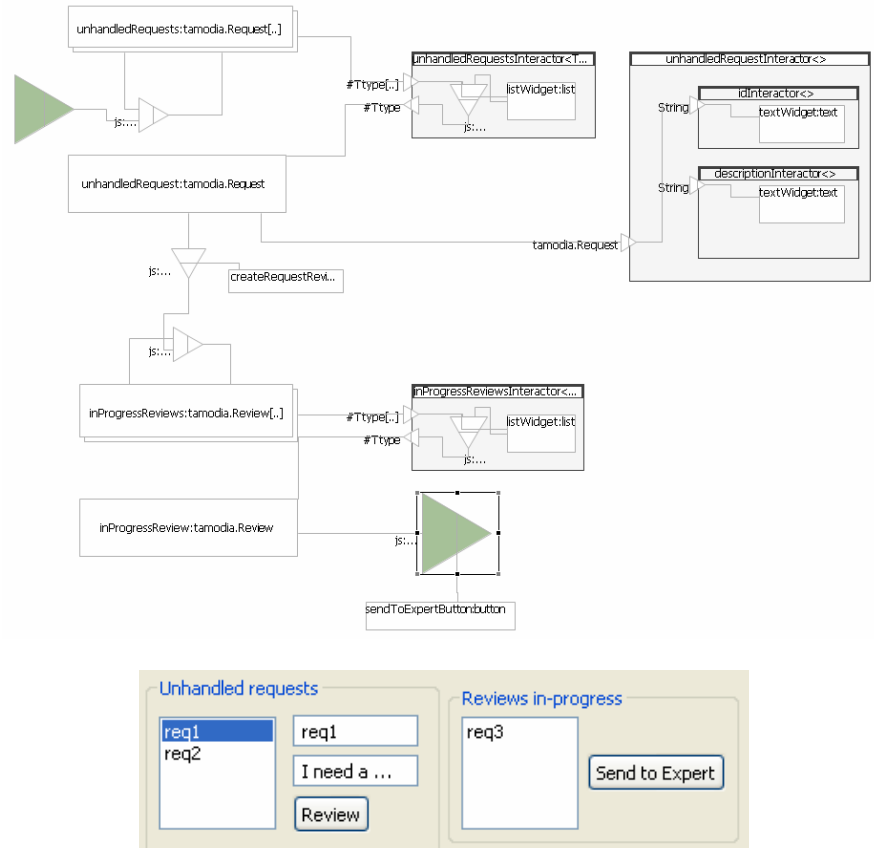

Fig. 4. Dialog model fragment and GUI prototype

The GUI and dialog model is executable, but the Diamodl runtime currently lacks general support for web services, so the final link between GUI and the BPEL engine is missing. We have, however, validated that we can initiate tasks from the Diamodl runtime and receive data from the BPEL engine, using the existing support for Javascript and XML. Similarly, although EMF-based data hasn't been integrated into the BPEL engine, EMF supports serializing and de-serializing Ecore instances as XML, so in principle any BPEL engine can store and communicate EMF-based data to and from the Diamodl runtime and web services.

6 Conclusion and further work

We have presented an approach for modeling business applications using BPMN and Diamodl, where BPMN is used for both process and task modeling and Diamodl for the UI structure and behavior. We have shown how these two modeling methods fit together and outlined a practical method for modeling and deployment, based on

standard components and architecture. Although some technical components have not been implemented, we have validated the feasibility of both the method and technology. Part of the method is currently being taught in an advanced course on model-driven development of IS at our department.

The goal is to complete the missing parts, by improving the connection between the three main elements of our approach, domain, process and dialog modeling using EMF, BPMN and diamodl. More specifically, we need to 1) add support for modeling web services in the domain model using EMF, to enable deployment of domain-specific web services, 2) add two-way support for invoking web services in the diamodl runtime and 3) improve handling of EMF-based data in a BPEL engine.

References

1. The Eclipse Modeling Framework home page: http://www.eclipse.org/modeling/emf/
2. Paternò, F. Model-based Design and Evaluation of Interactive Applications. Series of Applied Computing, Springer-Verlag London (2000).
3. Trætteberg H. Workflow and task modelling. Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces CADUI'99. Louvain-la-Neuve, Belgium, 21-23 October (1999).
4. Kristiansen, R., Trætteberg, H. Model-based user interface design in the context of workflow models. Proceedings of Tamodia'07, Toulouse, France, Nov. 2007. LNCS Springer (2007).
5. Bruno, A., Paternò, F., Santoro, C. Supporting interactive workflow systems through graphical web interfaces and interactive simulators. Proceedings of Tamodia'05. Gdansk, Poland. ACM International Conference Proceeding Series; Vol. 127 (2005)
6. Pontico, F.; Farenc, C.; Winckler, M. Model-based support for specifying eService eGovernment Applications. 5th International Workshop on TAsk MOdels and DIAgrams. Hasselt, Belgium. October 23-24 (2006).
7. Sukaviriya, N., Sinha, V., Ramachandra, T., Mani, S., Stolze, M. User-Centered Design and Business Process Modeling: Cross Road in Rapid Prototyping Tools. Human-Computer Interaction – INTERACT 2007, LNCS Volume 4662/2007, Springer (2007).
8. Object Management Group. Business process modeling notation specification, final adopted specification dtc/06-02-01 (2006).
9. Havey, M. Essential Business Process modeling. O'Reilly, Aug. (2005).
10. Trætteberg H. Dialog modelling with interactors and UML Statecharts - a hybrid approach. Design, Specification and Verification of Interactive Systems. Funcall, Madeira, June (2003).