Enhancing the Usability of BPM-Solutions by Combining Process and User-Interface Modelling

Hallvard Trætteberg and John Krogstie

Norwegian University of Science and Technology (NTNU), Trondheim Norway
{hal,krogstie}@idi.ntnu.no

Abstract. BPMN has over the last years appeared as a major approach for modelling process-oriented solutions. The approach is meant to work well both towards human understanding and execution. Executability is normally based on a mapping of BPMN-models to BPEL and defining a form for each flow where the user is the source or target. As we argue in this paper, this often gives sub-optimal and inflexible user interfaces. This paper describes our experiences with using BPMN for process and task modelling and Diamodl for model-based user interface dialog design. Although the added expressiveness and flexibility comes at the cost of introducing a model-oriented approach for dialog design, the necessary modelling steps follows the same kind of logic that is needed when going from a conceptual BPMN-model to an implementation-oriented model for a naïve generation of BPEL and forms UI, thus is interpreted as not extending the conceptual load of the approach significantly.

Keywords: Process modelling, user interface modelling.

1 Introduction

Models of business and work processes have for a long time been utilized to learn about, guide and support practice in a number of areas. In software process improvement [3], enterprise modelling [5], active knowledge modelling [12], and quality management, process models describe methods and standard working procedures. Simulation and quantitative analyses are also performed to improve process efficiency [11]. In process centric software engineering environments [1] and workflow systems [23] model execution is automated. Thus process modelling is not done for one specific objective only, which partly explains the great diversity of approaches found in literature and practice. Five main categories of usages of process modelling can be distinguished [10]:

1. Human sense-making and communication to make sense of aspects of an enterprise and to support communication among different stakeholders. Sense-making models are used within an activity in order to make sense of something in an ad-hoc manner, and will usually not be maintained afterwards.
2. Computer-assisted analysis to gain knowledge about the enterprise through simulation or deduction based on the contents of the model.

J. Stirna and A. Persson (Eds.): PoEM 2008, LNBIP 15, pp. 86–97, 2008.

3. Quality Management, following up the adherence of the work process to standards and regulations and to support process improvement. Here the model is meant to act as part of a corporate memory meant to exist as a reference point over time.
4. Model deployment and activation to integrate the model in an information system. Deployment can be manual, automatic (in automated workflow systems), or interactive [9].
5. Using the model as a context for a system development project, without being directly implemented (as it is in category 4).

Business Process Management (BPM) is a structured, coherent and consistent way of understanding, documenting, modelling, analyzing, simulating, executing and continuously changing end-to-end business process and all involved resources in light of their contribution to business performance. Thus we see that the potential usage of modelling in BPM is covering all the areas of use for process modelling as outlined above. In BPM and workflow there has traditionally been a wide variety of approaches and notations used. BPMN has over the last years been pushed forward and suggested as a standard and is met with the same kind of diverse needs; i.e. to be understandable for both humans and machines. The main approach for execution of BPMN is a translation to BPEL. The focus of BPEL engines is on process execution and not on the user interface of the application, which in practice can result in internally good process support systems that is hampered by an inappropriate, inflexible user interface, thus meeting unnecessary implementation problems when being introduced in an organization. Along the same line is the issue of the limited possibilities of tailoring such generated systems.

In this article, we will present the state of the art of combining process modelling notations such as BPMN and user interface modelling, to make it possible to guide the design of the process solution and take into account both process execution and user interface at a sufficiently early stage to have impact on the working solution. In addition to having a potential large impact in traditional development, as argued by e.g. Sousa et al [18], there is a need to better support the combined *evolution* of the business process and the interfaces of the systems. A combined model-based approach may improve on the current situation where process and UI must be considered individually, and it is hard to spot the places where user interface changes have impact on the process supported and vice versa.

In the next section we present BPMN in a sufficient detail to follow the argument. Then we present the field of MBUID (model-based user interface design), indicated how BPMN can be used for task modelling and present the Diamodl approach for the dialog part of user interface modelling. The following chapter will present the practical integration of these approaches, before summarizing the work.

2 Business Process Modelling Using BPMN

In this section we briefly introduce BPMN in order to give the reader sufficient background for understanding our subsequent arguments. The presentation is based on [17].

BPMN has over the last years been propelled as the most prominent candidate for an industry standard in process modelling, similar to the position of UML in

object-oriented design. BPMN was originally developed by the Business Process Management Initiative (BPMI.org). Its specification 1.0 was released in May 2004 and adopted by OMG for standardization purposes in February 2006 [2]. The development of BPMN was based on the revision of other notations, including UML, IDEF, ebXML, RosettaNet, LOVeM and EPCs, and stemmed from the demand for a graphical language that complements the BPEL standard for executable business processes. Although this gives BPMN a technical focus, it has been the intention of the BPMN designers to develop a modelling language that can be applied for typical business modelling activities as well. The complete BPMN specification defines thirty-eight distinct language constructs plus attributes, grouped into four basic categories of elements, *viz.*, Flow Objects, Connecting Objects, Swimlanes and Artefacts. *Flow Objects*, such as events, activities and gateways, are the most basic elements used to create Business Process Diagrams (BPDs). *Connecting Objects* are used to inter-connect Flow Objects through different types of arrows. *Swimlanes* are used to group activities into separate categories for different functional capabilities or responsibilities (*e.g.*, different roles or organizational departments). Finally, *Artefacts* may be added to a diagram where deemed appropriate in order to display further related information such as processed data or other comments. Fig.1 gives an example of a BPMN model that shows a simple review process. 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 review is then sent back to the Customer.

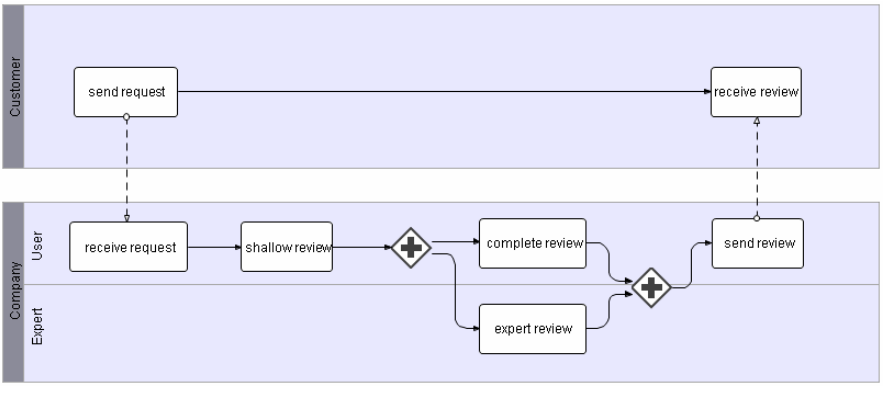

Fig. 1. BPMN model of a simple review process

3 Model-Based User Interface Design

The field of model-based user interface design (MBUID) has a long tradition [15], but it has been noted that diagrammatical UI modelling is not very common [13]. One of the reasons is that the methods, models and activities of MBUID are not aligned with the traditional system development practice including business process modelling. Generally a closer integration of MBUID and standard analysis and design modelling is called for. The attempts to link user interface modelling to design notations such as by creating

particular UML-profiles (e.g. [7,14]) has had limited impact in this regard since it is necessary to think on the user interface on an earlier level than detailed design.

Research in model-based UI design has resulted in many specific modelling languages methods. Although there is no UML of UI design, there seems to be a general agreement on three classes of models [22].

1. Task and domain models focus on the user's and designer's knowledge of the user, their goals and the tasks they need to perform.
2. Dialog models focus on input and output of information, Abstract Interaction Object (AIO) structure and dynamic behaviour. Dialog models abstract away details of interaction style and client platform.
3. Models of concrete design describe the structure and details of Concrete Interaction Objects (CIOs) and is specific to interaction style and client platform.

In the model-based approach, design progresses from task models, through dialog models to concrete interaction design, in a top-down process. Parts may be generated, based on logical dependencies, using formal systems like predicate logic, process algebra, Petri nets, state machines, etc [15].

In [16] several task and process modelling 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 modelling and task modelling in [7]. Our focus on this paper is on a lean method based on BPMN for process and task modelling and the Diamodl language for dialog modelling and deployment using standard, open-source tools and modern service-oriented architecture. [19] 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.

3.1 Using BPMN for Task Modelling

According to www.bpmn.org "… Business Process Modelling 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.

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 modelling 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 a task modelling language is more a matter of perspective than expressive power and both essentially model a task hierarchy. Similarly, the control flow connections of BPMN and operators in task modelling 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 and data modelling. 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 modelling, e.g. EMF [4], 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, it 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 precondition for performing a review of an application is 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 UML Use Case diagrams are elaborated by means of structured text.

3.2 Diamodl

The Diamodl notation [20,21] is a visual language for modelling abstract dialog, covering both information handling and activation logic, which is particularly suited for business applications. The interactor and gate concepts specify the input and output function of an AIO. The variable, computation and connection concepts are used for specifying how data is stored, transformed transported among interactors.

Fig. 2 shows a simple model where these five concepts are used. The upper left box is a variable and is used to store data. This particular one is limited to holding a

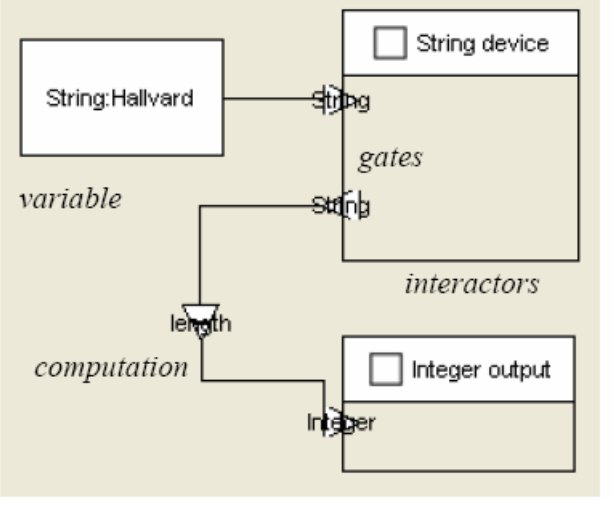

Fig. 2. The variable, interactor, gate, computation and connection concepts

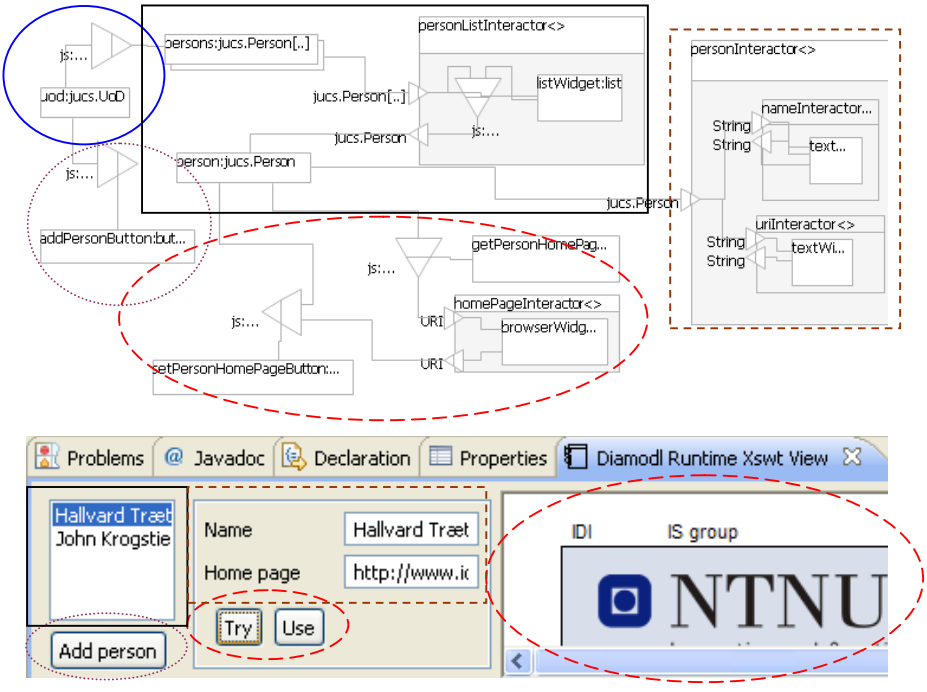

Fig. 3. Diamodl example

String and currently contains “Hallvard”. The two boxes with title bars to the right are interactors and the attached triangles are gates. As indicated by the gates, the upper interactor may output and input a String, while the lower one may output an Integer. The variable is connected to the output gate of the former interactor, while the input gate is connected to a computation, the output of which is further connected to the output gate of the latter interactor. The name and the input/output types suggest that the computation computes the length of the input String. Hence, this particular model specifies a UI through which the user may edit a String and see its length.

In addition to these five concepts, states (an interactor is also a state) and transitions are used for specifying the activation and deactivation of interactors, while UML class and object diagrams are used for data modelling.

In Fig. 3, a more comprehensive Diamodl example is shown, also indicating how the different parts of the model relate to the concrete user interface. As above, interactors are shown as rectangles with gate triangles on their left side, either pointing into for user output or out for user input. Variables are labelled with class names and multiplicity.

Computations are shown as free-floating triangles, and are labelled by attribute, association or operation names from the UML class diagram (not shown). Connections are shown as edges and may optionally be labelled with attribute or association names, to indicate an embedded computation. The links to the interface can be seen through:

- The solid circle contains a variable named `uod` holding the root `UoD` object, in which all other objects are contained (directly or indirectly). All the `Person` objects are extracted by following UoD's `allPersons` association and sent to the `persons` variable.
- The solid rectangle contains the list widget that is used for viewing the Person objects. It includes Javascript for extracting the `Person` instances' names. The `person` variable holds the selected `Person`.
- The `person` variable is connected to `personInteractor` inside the dashed rectangle, which models the fields used for viewing and editing the `name` and `homePage` properties.
- The browser widget and "Try" and "Use" buttons are modelled inside the dashed oval. The "Try" button triggers a computation which sends the selected `Person`'s `homePage` property to the browser, while the "Use" button triggers a computation that sets the same `Person`'s `homePage` property to the `URI` currently shown by the browser.
- The "Add person" button is modeled inside the stippled circle and creates a new `Person` instance and adds it to the `UoD` object's `allPersons` association.

4 Combining BPMN and Diamodl

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 behaviour, the dialog model describes software behaviour. The deployment of the UI will be a combination of concrete user interface elements and the software and models necessary for implementing the dialog behaviour, like state machines, data binding, etc. and the concrete interface describes what is actually deployed. This is actually fairly similar to the standard approach of business process modelling using BPMN and execution and deployment using the Business Process Execution Language (BPEL) [6]. First, the behaviour 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 (automation of) coordination (also called choreography or orchestration) 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, data bases etc.

As can be seen, the overall approach and role of the models is similar, although they have the (group) system perspective instead of the (individual) UI perspective. This suggests that the models can be related across the domains of business process management and user interface design, as illustrated in the architectural figure of Fig. 4. According to this figure, process models (in BPMN) may be related to task models since they both capture the behaviour of people, BPEL models may be related (architecturally, not methodologically) 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 an analogy, i.e. we propose a method whereby BPMN is

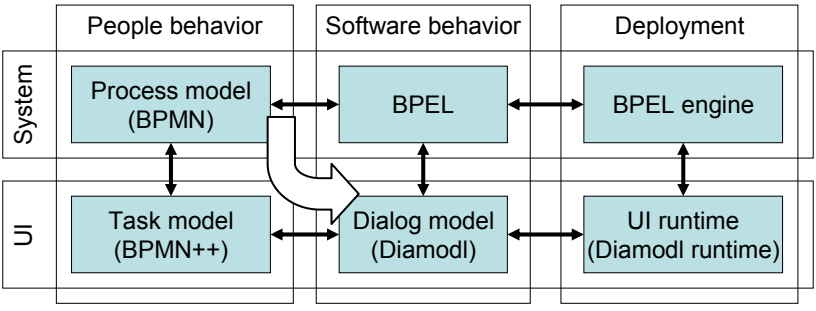

Fig. 4. Relationship between system and user interface domains

used for both business process and task modelling and BPEL and Diamodl are used for modelling software support and deployment on a SOA-based platform. The analysis and design tasks travels from the traditional BPMN model to the task model to the dialog model as indicated with the bold white arrow in Fig.4.

4.1 Step-by-step Modelling Method

Fig. 4 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 process. The process is illustrated by a simple example, that of reviewing a request (for something) and returning the answer as first illustrated in Fig 1. Creating this BPMN model is the first step in our method, combined with domain modelling, 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-centred 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 modelling language for domain modelling, and export the XML Schema from the Ecore editor. The Intalio Designer Eclipse application, which we use for BPMN modelling, allows us to open the XML Schema in the Process navigator and drag XML types into the connections in the diagram.

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. 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 design oriented process model is shown in Fig. 5. As can be seen, this process interacts with both the Executable process, i.e. the system, and the Expert role.

This design oriented 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. E.g. the pre-condition for the User task "shallow review" is that

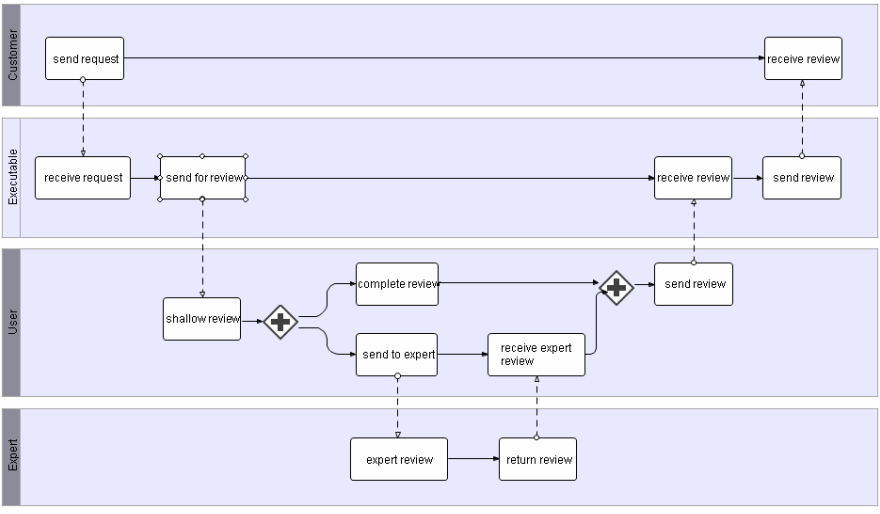

Fig. 5. Design-oriented process model

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 modelling 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).

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. 6. 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 model 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.

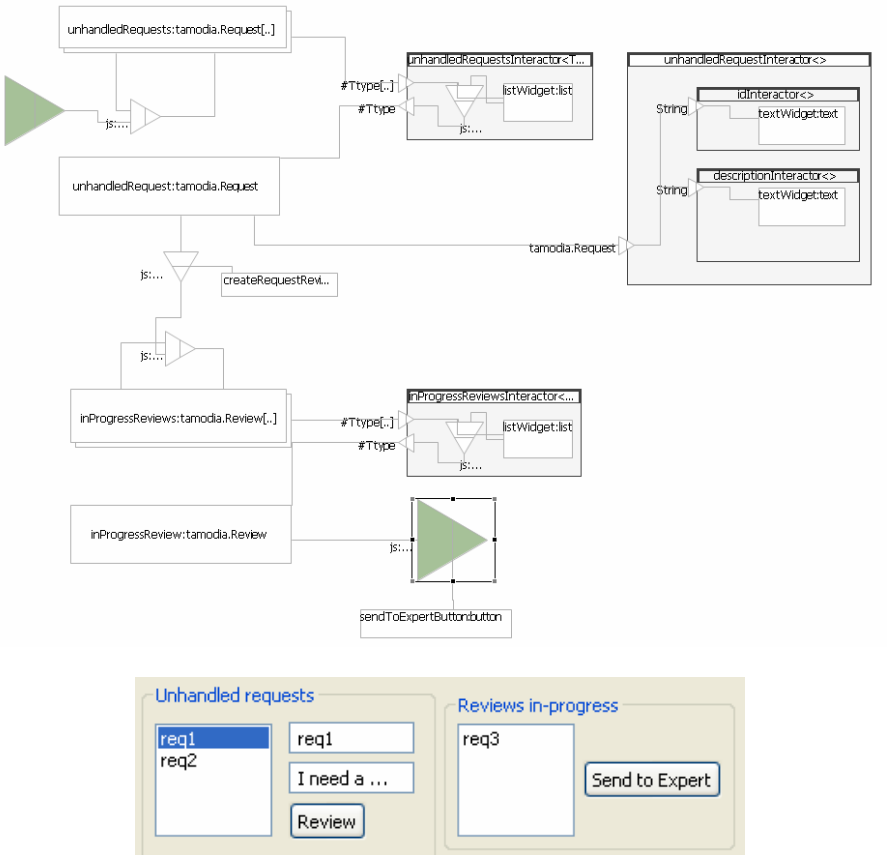

Fig. 6. Dialog model fragment and GUI prototype

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. 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.

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. In addition to doing this ourselves, we had last year 20 students using the approach as part of their regular master BPM-course. After the normal introduction to the different parts of the approach, and the environment, the students were able to develop simple applications using the approach.

5 Conclusions and Further Work

Business process modelling and management is getting increasing attention and use, as are other process-oriented approaches. The focus on process, originating from more radical ideas of BPR, has traditionally emphasized the improvement of organizational solutions. Although useful to avoid sub-optimalization within the different functions, we have seen e.g. in ERP-systems that this often create non-optimal solutions for the *individual* worker. Systems developed or generated to directly support the inter-organizational or organizational process often focus on a different aggregation of tasks than solutions geared towards supporting the individual worker.

This is only one area where traditional process solutions need improvement. The limited success of traditional workflow management systems (WMS) has partially been attributed to the lack of *flexibility*. Thus over the last years flexible workflow has been investigated. Most work within this area looks at how conventional systems can be extended and enhanced in other words how static workflow systems can be made *adaptive and tailorable*.

Traditional ERP systems for instance tend to be quite inflexible, hardly adaptable, and primarily support the organizationally agreed processes in a uniform manner independent on personal needs of the users. Existing workflow management systems have typically been focused on dealing with exceptions and have thus offered some support for adaptive processes. These types of systems, however, have typically overlooked emergent processes, which seem to encompass an increasing part of organized activity.

On the other hand, only supporting the emergent work style and individual needs of the individual knowledge worker is probably at times inefficient, because routine parts of the work can be prescribed and automated, and because sharing of explicit process models facilitates co-ordination, collaboration and communication between multiple parties. Thus there is a need for a balance between prescription and emergent tailorable representations. This is parallel to the main issue in this paper, how to balance the need of the organizational process with the need of user interfaces appropriate for the individual worker. How the integrated process and user-interface modelling approach discussed in this paper can be extended to address also this problem area is material for further work. As a first step, we plan a more thorough validation of the approach described in this paper.

References

1. Ambriola, V., Conradi, R., Fuggetta, A.: Assessing Process-Centered Software Engineering Environments. ACM Transactions on Software Engineering and Methodology 6(3) (1997)
2. BPMI. org and OMG (2006): Business Process Modeling Notation Specification. Final Adopted Specification, Object Management Group, `http://www.bpmn.org` (February 20, 2006)
3. Derniame, J.-C., Kaba, B.A., Wastell, D. (eds.): Promoter-2 1998. LNCS, vol. 1500. Springer, Heidelberg (1999)
4. The Eclipse Modeling Framework home page, `http://www.eclipse.org/modeling/emf/`
5. Fox, M.S., Gruninger, M.: Enterprise modeling. AI Magazine (2000)
6. Havey, M.: Essential Business Process modeling. O'Reilly, Sebastopol (2005)

7. Kovacevic, S.: UML and User Interface Design. In: Bézivin, J., Muller, P.-A. (eds.) UML 1998, LNCS, vol. 1618, pp. 253–266. Springer, Heidelberg (1999)
8. Kristiansen, R., Trætteberg, H.: Model-based user interface design in the context of workflow models. In: Winckler, M., Johnson, H., Palanque, P. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 227–239. Springer, Heidelberg (2007)
9. Krogstie, J., Jørgensen, H.: Interactive models for supporting networked organizations. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 550–563. Springer, Heidelberg (2004)
10. Krogstie, J., Dalberg, V., Jensen, S.M.: Process modeling value framework, Enterprise Information Systems. In: Manolopoulos, Y., Filipe, J., Constantopoulos, P., Cordeiro, J. (eds.) Selected papers from 8th International Conference, ICEIS 2006, Paphos, Cyprus, May 23-27, 2006. Lecture Notes in Business Information Processing, vol. 3. Springer, Heidelberg (2008)
11. Kuntz, J.C., Christiansen, T.R., Cohen, G.P., Jin, Y., Levitt, R.E.: The virtual design team: A computational simulation model of project organizations. Communications of the ACM 41(11) (1998)
12. Lillehagen, F., Krogstie, J.: Active Knowledge Modeling of Enterprises. Springer, Heidelberg (2008)
13. Myers, B., Hudson, S.E., Pausch, R.: Past, Present and Future of User Interface Software Tools. ACM Transactions on Computer-Human Interaction 7(1) (March 2000); Applied Computing. Springer, London
14. Nunes, N.J., Cunha, J.F.: Towards a UML Profile for Interaction Design: The Wisdom Approach. In: Evans, A., Kent, S., Selic, B. (eds.) UML 2000, LNCS, vol. 1939, pp. 101–116. Springer, Heidelberg (2000)
15. Paternò, F.: Model-based Design and Evaluation of Interactive Applications. Series of Applied Computing. Springer, London (2000)
16. Pontico, F., Farenc, C., Winckler, M.: Model-based support for specifying eService eGovernment Applications. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) TAMODIA 2006. LNCS, vol. 4385, pp. 54–67. Springer, Heidelberg (2007)
17. Recker, J., Indulska, M.: An Ontology-Based Evaluation of Process Modeling with Petri Nets. Journal of Interoperability in Business Information Systems 1(2), 45–64 (2007)
18. Sousa, K., Mendonca, H., Vanderdonckt, J.: User Interface Development Lifecycle for Business-Driven Enterprise Applications. In: Proceedings of Seventh International Conference on Computer-Aided Design of User Interfaces CADUI 2008, Albacete, Spain (June 2008)
19. Sukaviriya, N., Sinha, V., Ramachandra, T., Mani, S., Stolze, M.: User-Centered Design and Business Process Modeling: Cross Road in Rapid Prototyping Tools. In: Baranauskas, C., Palanque, P., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 165–178. Springer, Heidelberg (2007)
20. Trætteberg, H.: Dialog modelling with interactors and UML Statecharts - A hybrid approach. In: Markopoulos, P., Johnson, P. (eds.) Proceedings of DSV-IS 1998. Springer, Heidelberg (1998)
21. Trætteberg, H.: Dialog modelling with interactors and UML Statecharts - a hybrid approach. In: Jorge, J.A., Jardim Nunes, N., Falcão e Cunha, J. (eds.) DSV-IS 2003, vol. 2844, pp. 346–361. Springer, Heidelberg (2003)
22. van der Veer, G.C., van Welie, M.: Task Based Groupware Design: putting theory into practice, van der Veer, G. In: van der Veer, G.C., van Welie, M. (eds.) Proceedings of DIS 2000, New York, United States, August 17-19 (2000)
23. Workflow management coalition (WfMC), Terminology & Glossary (1999), `http://www.wfmc.org/standards/docs.htm`