Integrating Business Process and User Interface Models using a Model-Driven Approach

Renata Dividino*, Veli Bicer†, Konrad Voigt‡and Jorge Cardoso‡
* ISWeb Research Group University of Koblenz-Landau
Email: dividino@uni-koblenz.de
† Forschungszentrum Informatik an der Universität Karlsruhe (TH)
Email: bicer@fzi.de
‡SAP Research CEC Dresden
Email: {konrad.voigt,jorge.cardoso}@sap.com

***Abstract*—Business services are complex entities that encompass descriptions about different aspects including business processes and user interfaces. Typically, the modeling of business services results on several correlated models. On the one hand, there is the need to keep these models apart in order to attain the levels of abstractions needed to model different aspects of a service. On the other hand, it is important to maintain the consistency and integrity of models. In this paper, we show how to maintain the consistency and integrity of models and present a use case for the integration of user interface design and business process models.**

***Index Terms*—Service Engineering, User Interface Design, Model-driven Engineering, Business Process Integration.**

I. Introduction

The Internet of Services brings a set of new challenges for the research community. One of the challenges that needs to be handled is the complexity associated with the modeling of real-world or business services. Services are complex entities that encompass descriptions about business processes, user interfaces, pricing models, legal constraints, knowledge assets, technological infrastructures, business rules, etc. One approach to manage the lifecycle of a service multifaceted structure is to distill it core characteristics into formal models and afterwards align and integrate the models. This paper describes how a model-driven approach can be used to model and maintain the consistency and integrity of business processes models (which describe services' internal behavior) with service user interface models. While a service is typically composed by several models (up to 20 in our research [3]), we demonstrate the feasibility of our approach using only two models, which takes parts on the traditional design of the user interface (UI) and business processes, since it provides a concrete list of the major steps to take.

As a use case, we deploy a business service and model its underlying business process model. The business service is called EcoCalculator Service (ECS) and its main objective is to calculate the ecological value (eco value or EV) of products (e.g. car parts such as car sits) according to European directives and worldwide environmental regulations. We show the business process modeled with BPMN [1] showing the set of activities that needs to be carried out when the ECS is invoked by a customer, and the UI dialogue modeled with DIAMODL [2] which illustrates the user interaction. In addition, to provide support for business services modeling integrated with user interface design, we extend the modeling languages BPMN and DIAMODL, and implement integration rules in order to provide support for business process modeling integrated with UI design.

II. ISE Methodology for Service Engineering

In this section we cover our approach for service engineering, called ISE (an acronym for Inter-enterprise Service Engineering). First we introduce the service engineering discipline, describing its main aspects. This is followed by an overview of ISE, describing how the methodology applies Model Driven Engineering (MDE) in order to provide an integrated framework and implementing workbench.

A. *Service Engineering*

Service engineering (SE) [3] is a new approach to the design and implementation of services. Typically services evolve in a common ecosystem in which organizations and IT provide value in form of services. SE provides methodologies to cope with the complexity of several business actors and their interaction. Furthermore, SE specifies tools for implementing and deploying services, covering both, IT and business perspectives. Consequently, SE is a structured approach for creating a new (e-)service [3]. Approaches should translate an initial abstract description (e.g. natural language), through a sequence of representations, to a technical standard-based representation. The ISE methodology, described in the following section, addresses exactly the creation of new services using a guiding methodology.

B. *ISE Methodology*

In the context of the TEXO project[1], the ISE methodology [3] and accompanying workbench for service engineering was implemented. Compared to other approaches, the ISE methodology considers a technical perspective as well as a business and operational perspectives. Since the notions

[1]TEXO Business Webs in the Internet of Services (http://theseusprogramm.de/scenarios/en/texo)

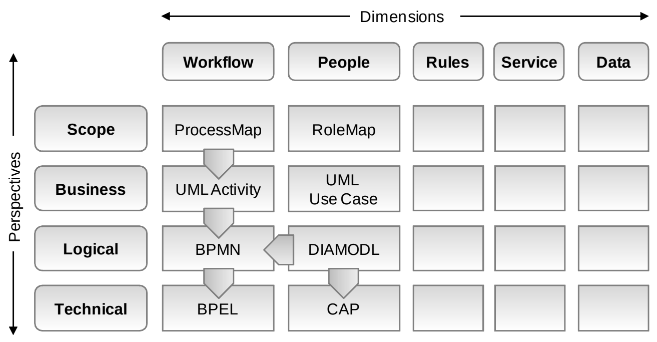

Fig. 1. ISE Matrix with separation into dimensions and perspectives, corresponding models, and exemplary transformations for UI and business processes models

of abstraction (perspective) and dimensions (entities) were important for our approach, we have followed a solution based on the Zachman framework [4]. ISE combines this business approach with the concepts of Model Driven Engineering (MDE) [5].

Figure 1 shows the ISE framework which is part of ISE methodology. Inspired by Zachman framework and following the separation of concerns paradigm raised by [6], it divides and structures services into four main perspectives and five dimensions. The dimensions are: service description, workflow, data, people and rules. Figure 1 presents a reduced view, showing only the models relevant for this paper. Furthermore, each of these dimensions is divided into four perspectives (layers) of abstraction. Each of the perspectives of the ISE methodology can be regarded as a phase in the development (model refinement) of services. Thus, the models which are assigned to each layer support the development from different viewpoints (i.e., scope, business, logical, and technical). Additionally, models at different dimension but belonging to the same layer are integrated/binded to others in order to form the complete business service model at the respective level of abstraction. For all cells of the matrix, we have defined formal models which should be considered in the service development. Examples of models include UML, mind maps, BPMN, BPEL, OWL, DIAMODL, etc.

Therefore, ISE act as an integration platform for several models stored in cells of the framework. Throughout one dimension, models are created with respect to different views and refined until they conform to a technical specification. This leads to multiple representations of information on different layers of abstraction in the corresponding dimensions. Changes in one model have to be propagated into related models holding overlapping information (depicted by arrows in Figure 1). We divide the dependencies between models into two classes: (1) Vertical dependencies cover the synchronization of dependencies between models on different layers of abstraction in one dimension. (e.g. BPMN and BPEL) and (2) Horizontal dependencies define the synchronization of models on the same layer of abstraction. (e.g. BPMN and DIAMODL).

These dependencies requires the integration of models. We have adopted a model transformation from MDE to allow for an automatic support of integration. Based on a common formal representation (e.g. MOF), a domain specific language for model transformation can be used to define rules and apply them to the models, in order to generate elements from a source model to a target model automatically. ISE uses for model integration the the Query, View and Transformation (QVT) technique. We have chosen to rely on QVT because of matured concepts, well established infrastructure for model management and transformation, and available OMG standards. The integration is again depicted in Figure 1 using arrows. In the following sections we will show how such integration can be done.

III. Business Process Modeling with UIs

In this section, we describe the theories for UI design and business process modeling used in this work, and introduce the DIAMODL and BPMN modeling languages.

A. User Interface Design and DIAMODL

A widely accepted principle of methodologies on UI design [7], [8], [9] is the separation (so called modeling in layers) of concerns with respect to the modeling of the structural, behavior and presentational aspects of UIs. In terms of ISE, structural aspects are included at the scope, business dimensions; behavior aspects at the logical dimension; and presentation aspects at the technical dimension. The main idea of the traditional delineation between structure, behavior and presentation (with additional layers of granularity) is to describe the full set of issues and considerations involved in more complex forms of interactive UIs.

Behavioral aspects mainly describes interactive qualities of the UI, i.e. how users and systems cooperate to perform the desired operations. Interactions can be modeled as a dialogue where operations that allow users to exchange message with the interface are defined. The UI is used as an communication channel for the user to exchange message with the system. Note that this layer acts as a glue bringing together system and interface (users).

Within the ISE methodology, behavior aspects of an UI are modeled using the DIAMODL notation. The DIAMODL notation proposed by [2] offers *interactors-with-gates* (i.e. states-with-transitions) interface component abstraction and the hierarchical state formalism statecharts for describing the UI's behavior. DIAMODL's five core elements are:

- *Interactors.* Interactors represent a distinct context for behaviors of the UI, such as an "input" state and an "output" state.
- *Gates.* Gates determine how to interpret states i.e. they model the actions associated with a state, and each state can be associated with anther state through *connections*.
- *Connections.* Connections are directed associations between any two states that dictates under what conditions ("triggers") the UI shifts from the first context (state) to the second.
- *Variables.* Variables are used for holding a resource (data value) and its contained structure.
- *Computations.* Computations enable message exchanges among UI components, and between a UI component and an interactive system.

Figure 2 presents a simplified DIAMODL model for the ECS example. The main input of the ECS is the identification of the part (product) for which the EV will be calculated. The user inputs the PID-value (*textInteractPID*) which is internally stored in a variable (*PDI*). When the user clicks on the OK button (*buttonInteractor*), the PID-value is sent to the system. Since the calculation of EV is highly dependent on directives and environmental regulations holding for the country in where the product is built and/or exported to, the country name has to be be specified by the user (*comboInteractorCountry*) – as default value the European directives are considered. This information is forwarded to the system which calculates the respective EV and returns it to the user (*textInteractorEV*).

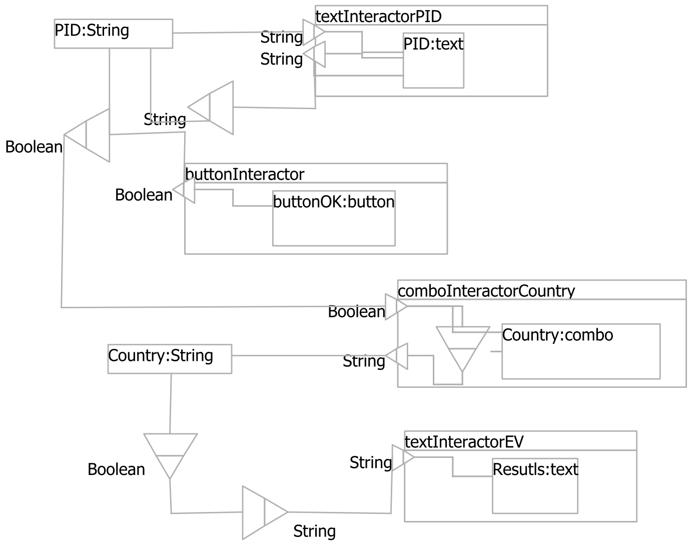

Fig. 2. DIAMODL model for the ECS

B. Business Service Choreographies and BPMN

Modeling business processes, at a high level, requires the specification of the involved parties and process flow according to business objectives of the service. Within ISE methodology, the Business Process Modeling Notation (BPMN) is adopted to address this need. It is one of the well-known languages that aims to specify business processes at a high-level, rather than as executable workflows. It provides a modeling mechanism with a standard visual representation to create business process models and abstract the complexity of the process-level service design. According to BPMN specification [1], there are four types of core elements that helps to define processes:

- *Flow Objects.* These are the main graphical elements to define the behavior of a process model. They mainly includes events, activities, and gateways. Events indicate the triggers that occur during the course of the process. BPMN distinguishes start events and intermediate events as two kinds of event consumption whereas end events are only used for event production. Start events lead to process instantiation and intermediate events are consumed by a running process instance. Furthermore, activities are generic constructs to represent particular actions (e.g. tasks and sub-processes) to be performed in process. Gateways, on the other hand, are used to control splits and joins in the process flow.
- *Connecting Objects.* Connecting objects are used to link flow objects and other elements. There are three ways to connect elements which are: sequence flow, message flow and association.
- *Swimlanes.* Swimlanes are elements to represent the boundaries that exist in a process model due to the inter-organizational nature of business processes. The top-most swimlane element is the pool which categorizes and encapsulates the activities performed by an organization entity or a system. A pool can then be partitioned into lanes.
- *Artifacts.* Artifacts are used to provide additional information about the process such as annotations or data objects.

Figure 3 shows core BPMN elements used to model the business process of the ECS. The service starts with the PID-value input. The first two activities to be executed in parallel are the *Loader* (Data Service) and *Process Logging*. The *Loader* retrieves the *BOM* (bill of material) based on the *PID* supplied by the consumer. In parallel, the *Process Logging* activity writes log information indicating that a process instance has started. Once those two activities are completed, the process model checks if the *BOM* has information describing all the subparts. Additionally, it checks if the name of a country is specified in which regulations will be used for the computation of the *EV*. If no information is missing, the *EV* is computed and *Billing*, *Payment*, *Performance Analysis* and *Logging* activities are executed. For reader's convenience, UI part is also shown as illustrations of the components. Intuitively, this graphical notation of BPMN essentially provides a special kind of flowchart incorporating constructs tailored to business process modeling. Although it is quite useful in this sense, there is a lack of elements to model constructs specific to UI models.

C. DIAMODL and BPMN Synchronization Requirements

In view of service engineering requirements, both service business process and user interaction contribute to the execution of a service to perform its functionality for a desired outcome. They contribute to the design from different angles, but need to be in synchronized in order to result in a coherent execution after service deployment. The common action units for both models are the activities. In DIAMODL, they correspond to the *interactors* and *computations* considering their assignment to a user or system, respectively. In BPMN, activities are either *tasks*, or *subprocesses*. However, BPMN does not explicitly state by whom an activity is performed, and accordingly no distinction of activity types (i.e. user or system). As expected, each model assess the other's activities in a black-box manner – i.e. DIAMODL *computation* represents the whole business process activities and BPMN *tasks* standing for user activities and interactors.

Intuitively, the execution of activities from start to end in both models are *event-driven*. Although the activities lead to accomplishing overall service goal separately in business process and UI, their flows need to be aligned to fit each

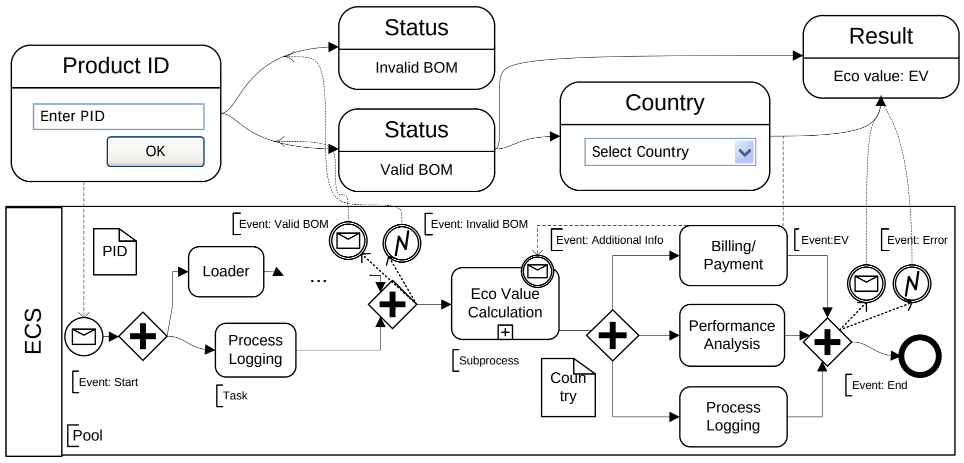

Fig. 3. BPMN model showing core elements and interactions with UI components

other. In Figure 3, this is illustrated using dashed lines between the UI components and business process elements. First, the UI needs to be aware of the state of the business process in order to set its behavior. This include switching from one interactor to another depending on the state. More importantly, the business process is bound exactly to the events triggered by UI with the intention of avoiding non-deterministic behavior. In other words, the business process must ensure that an event occurring during its execution is properly handled by the corresponding UI elements in the case of a human involvement is needed. Therefore, we consider an event-based coordination to achieve the synchronization between two models.

However, utilizing such a synchronization (by means of model transformations as detailed in the next section) requires both models to be semantically equivalent in terms of modeling events. Considering the current state of DIAMODL and BPMN specifications, we distinguish three important points that need to be addressed:

- *Events in DIAMODL.* DIAMODL threats events implicitly as parts of other components such as computation objects or interactors. Computations support communication and synchronization with internal and external entities. However, even though computations explicitly state who are the participants of the dialogue, they do not define explicitly which are the events and event-types to be catched/thrown and the event-conditions-rules holding in the dialogue. In order to utilize their consumption and production with business process events, events triggered within computations need to be explicitly defined as a part of a model.
- *Events in BPMN.* BPMN comes with a rich set of event types. However, it does not have the capability to express different levels of events, i.e. specifying events only to be consumed by UI. This is required since a business process can interact not only with UI, but also with other business processes in a BPMN model.
- *Two-way synchronization.* According to ISE methodology, both models are designed separately as a part of different service aspects. This confronts the requirement of capturing the common information in both models and adapt them for synchronization. A declarative technique (e.g. QVT) is needed to be implemented for model transformations.

IV. Integration of UI oriented methodology

We extend two well-accepted existing languages, the BPMN and DIAMODL, in order to have equivalent well-defined semantic in terms of modeling events. Furthermore, we show the synchronization between both models by means of models transformation algorithms.

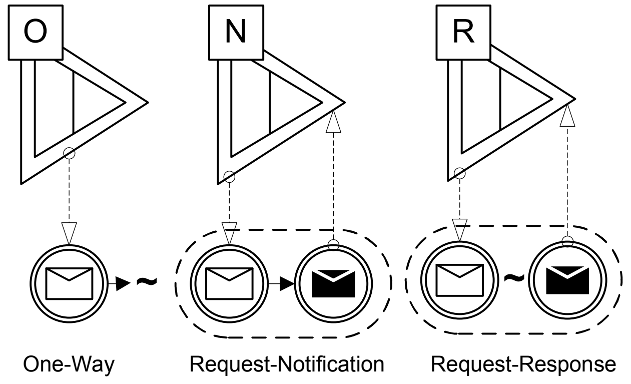

Fig. 4. Matching interactions between DIAMODL and BPMN

A. Extension for the DIAMODL Notation

Following the requirements described in Section III, we propose a three refinements of the computation component. First, we differ between *internal computation* and *external computation* components. Internal computations are value-based mapping functions holding among UI components. External computation, however, are value-based mapping functions holding between UI components and an interactive system (as network services). For this purpose, external computations must include the description of endpoints and their messages regardless of what message formats or network protocols are used to communicate. In addition, when more than one external computation hold a dialgue with a service in which inputs and outputs are dependent from each other, external computation should define the message sequence flow, e.g. define which message should be processed first (*start-message*), which are the intermediate ones (*intermediate-message*), and the last one (*end-message*) before closing the communication channel. The symbols for internal and external computation are shown in Figure 4.

Second, computation components include event-triggers. We define three types of event messages: response, notification and error. Based on these message types, we specify a categorization of event triggers into *catching* and *throwing* events. Additionally, event trigger are sub-divided in interaction types. Throwing events' interaction types are: (1) One-way - an operation that only sends an input (2) Request-response - An operation sends a request, then waits for a response or an error (3) Request-notification - An operation sends a request, then waits for the message notification or an error. Catching events are responsible for the reception of messages. Analogous to the throwing events, we define one-way, request-response, and request-notification interaction types for catching events.

At last, computations are extended with *Event-Condition-Action* (ECA) rules which automatically perform actions in response to events provided that stated conditions hold. ECA rules introduces reactive functionality to computations, and thus allowing computations to determine the direction of the dialogue flow based on a single rule. Example 4.1 shows a ECA rule for a *notify-response*–like message exchange. One direction is specified for the dialogue flow when the event-message is of type notification, and a different one for error messages.

Example 4.1: ECA rule for *notify-response*–like message exchanges

```
ON input-event
IF input-event.type == NOTIFICATION
DO goAssociation.sourceNode = compEx
```

B. Extension for Business Process Modeling Notation (BPMN)

The aforementioned BPMN events (e.g. start, intermediate and end) provide basic functionalities that allows to interact with the UI during the execution of a process. However, a BPMN event only provides one specific functionality such as receiving or sending a message (differentiated by black-white colors in Figure 4). Therefore, only one-way interactions can be represented with the current semantics of events stating that an event is either receiving or sending a message.

Since we want to match different event-type computations with BPMN events for a seamless interaction, there is a need to associate more than one event activity to the other types of interaction. For this, we use the idea of event patterns and introduce an *event conjunction* pattern as a container to include more than one event for a new abstract event (more patterns for complex events can be found in [10]). In Figure 4, this is shown using a dashed-line around two sending and receiving message events. Strictly speaking, an *event conjunction* pattern is just a container intended to be represented in formal syntax of the model, because the events can be interleaved. For example, the start and end events of the process can be in the same container, matching the whole process to a computation (request-response) in DIAMODL.

```
transformation Diamodl2Bpmn(dia:diamodl, bpmn:bpmn)
{
  ...
  top relation Comp2Activity {
   domain dia c:Computation{
     name=nm, type = 'external' }
   domain bpmn a:Activity{ name=nm }
   where{
     if (c.interactionTyp='O') transOneway(c,a)
     elif (c.interactionTyp='N') transNotification(c,a)
     elif (c.interactionTyp='R') transResponse(c,a)
     endif; }
  }
  ...
  relation transResponse(c,a){
      domain dia c:Computation{}
      domain bpmn a:Activity{
         activityType='EventConj',
         subEvent sa1:Activity{},
         subEvent sa2:Activity{}
      }
      where
      {
         setSubEvents(c, a);
       }
  }
  ...
  helper setSubEvents(c:Computation,a:Activity)
  {...}
  ...
}
```

Fig. 5. A fragment of transformation between DIAMOLD and BPMN in QVT syntax

C. Synchronization between Models

Finally, a QVT-based model transformation is used to synchronize the models during service design. Figure 5 shows a fragment of the transformation using the QVT syntax. It should be noted that the transformation is defined based on the corresponding meta-models defined according to Eclipse Modeling Framework[2] (EMF) specification. The actual mapping between the computations and events are handled by top-level *Comp2Activity* relation. It is specified on two *domains* of types Computation (c) and Activity (a). A pattern is applied on Computation to check whether it is external or internal – i.e. only external ones are mapped. Additionally, we correlate the matching elements by their names with an assumption that elements are uniquely labeled in both models.

In BPMN meta-model, all elements belong to a generic activity class, that is further specialized by their *activityType* attributes. This is achieved using three additional sub-relations that are called in the *where* clause based on the *interactionType* of the Computation. Due to space limitations, we only show the *transResponse* relation that maps a request-response type

[2]http://www.eclipse.org/modeling/emf/

Computation to an event conjunction pattern. It is introduced as a new activity type (i.e. EventConj) in BPMN meta-model which involves one or more sub-events.

V. Related Work

The need of modeling human user interaction within process modeling reflect the increase of recent work on this area. In [11] and [12], the authors use the standard process modeling language BPMN for both modeling processes and task modeling, and DIAMODL for modeling the UI structure and behaviour. The authors propose a refactoring process of the original BPMN model as an intermediary step in order to be synchronized with the DIAMODL model. This refactoring process aims to make explicit what each user does and how it interacts with its environments. However, they not specify how the exchange of message-event and data is conducted. In [13] and [14], the authors use a less formal UI model and couple to the BPMN when the business process is used as a starting point for the UI design. UI models and business process models are alignments with business requirements. On the one hand, it is not clear how the business requirements are collected, neither how the alignment of the models is carried out. In our approach we propose the alignment of the models following business requirements which, in turn, are defined in perspectives/dimensions following a formal methodology. We still keep the independence among models, so that we assure, for instance, that one business process model can be coupled with different UI behaviour models (which focus on the interaction regardless the environment considered). In [15], the authors apply a model-driven approach that derives UIs from business processes in order to keep the traceability of the business process and the UI. The authors make use of task models to bridge business process and UI design. In [16], the authors describes the design of user interface for a Web service based on the WSDL (Web Service Description Language) description. For the UI design, they have developed a new UI specification language and the associated authoring environment, which can be used for the abstract and concrete UI definition. Operations and data types defined in WSDL are bound to the UI model.

VI. Conclusion

Service engineering converges into a complex task of modeling business services involving various aspects to specify both execution and usage of services. Typically, the modeling process of business services results on several correlated models. On the one hand, there is the need to keep these models apart in order to attain the levels of abstraction to model different aspects of a service. This is mostly due to the fact that incorporating different models for different aspects of the service enables the separation of concerns during the design while attaining a staged development to evolve the service idea into an executable artifact. On the other hand, it is important to maintain the consistency and integrity of business processes models.

The ISE methodology for engineering services supports the modeling of business services based on perspectives and dimensions. This methodology directly meets the requirements of the independence of models, which however should be aligned and integrated. In this paper, we demonstrate the feasibility of our approach using only two models which takes parts on the tradition design of the user interface and business process of a service. To make the relationship between business process and UI models explicitly, we extend two well-accepted existing languages: the business process modeling language BPMN, and the user interface dialogue language DIAMODL with new event-based components, and implement model-driven transformation rules in order to provide support for business process integrated with user interface design. The extension of both models was necessary to focus on aspects of communication and synchronization.

Our future work aims to complete transformation rules of the ISE matrix and, thus, study in more detail the relationship between all models. Additionally, we plan the evaluation of ISE workbench for guiding independently improvements on the modeling languages at each perspective and dimension.

Acknowledgment

This research was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012. The authors take the responsibility for the contents.

References

[1] J. Cardoso, K. Voigt, and M. Winkler, "Service engineering for the internet of services," in *Enterprise Information Systems X*, 2008, pp. 17–25.

[2] S. White, "Introduction to BPMN," *IBM Cooperation*, pp. 2008–029, 2004.

[3] H. Traetteberg, "A hybrid tool for user interface modelling and prototyping," in *CADUI'06*. Bucharest, Romania: Springer, 2007.

[4] J. A. Zachman, "A framework for information systems architecture," vol. 26, no. 3. NJ, USA: IBM Corp., 1987, pp. 276–292.

[5] T. Stahl and M. Völter, *Model-Driven Software Development: Technology, Engineering, Management*, 1st ed. Wiley&Sons, 2006.

[6] D. L. Parnas, "On the criteria to be used in decomposing systems into modules." NY, USA: Springer, 2002, pp. 411–427.

[7] B. Baxley, "Universal model of a user interface," in *DUX '03*. NY, USA: ACM, 2003, pp. 1–14.

[8] J. J. Garrett, *The Elements of User Experience: User-Centered Design for the Web*. CA, USA: New Riders Publishing, 2002.

[9] L. Constantine and L. Lockwood, *Software for use: a practical guide to the models and methods of usage-centered design*. NY, USA: ACM, 1999.

[10] A. P. Barros, G. Decker, and A. Grosskopf, "Complex events in business processes," in *BIS*, ser. LNCS, vol. 4439. Springer, 2007, pp. 29–40.

[11] H. Traetteberg, "UI Design without a Task Modeling Language – Using BPMN and Diamodl for Task Modeling and Dialog Design," in *HCSE-TAMODIA '08*. Berlin: Springer, 2008, pp. 110–117.

[12] H. Trætteberg and J. Krogstie, "Enhancing the usability of bpm-solutions by combining process and user-interface modelling," in *PoEM*, 2008, pp. 86–97.

[13] N. Sukaviriya, V. Sinha, T. Ramachandra, and S. Mani, "Model-driven approach for managing human interface design life cycle," in *MoDELS*, 2007, pp. 226–240.

[14] N. Sukaviriya, V. Sinha, T. Ramachandra, S. Mani, and M. Stolze, "User-centered design and business process modeling: Cross road in rapid prototyping tools," in *INTERACT (1)*, 2007, pp. 165–178.

[15] K. S. Sousa, H. M. Filho, and J. Vanderdonckt, "Addressing the impact of business process changes on software user interfaces," in *BDIM*, 2008, pp. 11–20.

[16] F. Paternò, C. Santoro, and L. D. Spano, "Designing usable applications based on web services," in *I-USED*, ser. CEUR Workshop Proceedings. CEUR-WS.org, 2008.