In Proceedings of the 6th International Conference on Web Engineering; Palo Alto, California, USA; Pages: 117 - 118; ISBN:1-59593-352-2

Stakeholder Collaboration – From Conversation to Contribution

Martin Nussbaumer
University of Karlsruhe
Engesserstr. 4
76128 Karlsruhe, Germany
+49 (721) 608-8073
nussbaumer@tm.uka.de

Patrick Freudenstein
University of Karlsruhe
Engesserstr. 4
76128 Karlsruhe, Germany
+49 (721) 608-8042
freudenstein@tm.uka.de

Martin Gaedke
University of Karlsruhe
Engesserstr. 4
76128 Karlsruhe, Germany
+49 (721) 608-8076
gaedke@tm.uka.de

ABSTRACT
Establishing efficient and intensive stakeholder collaboration is a key factor in Web application development projects. Therefore, the form of collaboration needs to be shifted from simple conversation to valuable contribution, i.e. empowering stakeholders to directly contribute to the development effort. To achieve this, we introduce an approach combining Domain-specific Languages and an underlying technical platform.

Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable Software - *Domain Engineering;* D.2.2 [**Software Engineering**]: Design Tools and Techniques - *Evolutionary prototyping*.

General Terms
Human Factors, Languages, Design

Keywords
Web Engineering, DSL, Workflow, Conceptual Modeling

1. INTRODUCTION
In Web application development projects, intensive stakeholder collaboration is decisive for a project's success, especially when specifying requirements and aspects of the envisioned solution [4]. This requires a shift from stakeholders acting solely as information providers to stakeholders contributing actively and sometimes even autonomously to the development effort by understanding, validating and specifying parts of the solution being built. In order to allow for such an intensive form of stakeholder involvement, a set of common languages being used as a means of specification and communication between the developers and the stakeholders is required.

In our experience, stakeholders are mostly non-programmers with completely diverse educational backgrounds and work areas. Furthermore, most of them are not skilled in Web application development methodologies or modeling techniques. In our projects, we found existing approaches [3] towards a formal and systematic specification of Web applications to be a good means of specification and communication within the developer team, especially as they were designed to capture their associated problem domain as extensive as possible. However, for stakeholders they were too complex and required too much time to learn.

In this regard, *Domain-Specific Languages (DSLs)* are a promising alternative. DSLs are small, simple and highly-focused specification languages tailored to a clear problem domain, i.e. a specific aspect of a Web application. Incorporating well-known abstractions and notations derived from the problem domain, they are easy to learn, understand and use – both for developers *and* stakeholders. Employing DSLs in Web application development enables leveraging stakeholder collaboration from conversation to contribution.

2. DSL BUILDING BLOCKS
Figure 1 gives an overview of a DSL's components, assigned to different layers according to their level of abstraction.

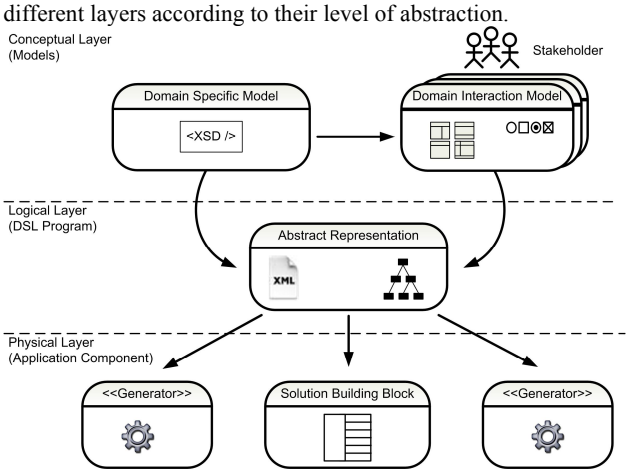

Figure 1: Overview of the components of a DSL

The conceptual layer comprises two models that constitute the core of a Domain-Specific Language: *The Domain-Specific Model (DSM)* and the *Domain Interaction Model (DIM)*. The DSM, usually an XML Schema, embodies the core of a DSL specification and provides a formalized model for describing solutions within the DSL's problem domain. In addition, a DSL includes one or more Domain Interaction Models which are based on the DSM and specify dedicated (graphical) notations for

In Proceedings of the 6th International Conference on Web Engineering; Palo Alto, California, USA; Pages: 117 - 118; ISBN:1-59593-352-2

particular stakeholder groups. By providing audience-tailored DIM notations and complementary editing tools, the simplicity and usability of a DSL can be notably improved. The *Domain Abstract Representation (DAR)*, located on the logical layer, represents a "DSL program". Thus, it is based on the DSM, edited by using the DIM notations and stored in an XML document. The physical layer of a DSL consists of so-called *Solution Building Blocks (SBBs)* which are implemented as components of the WebComposition Service Linking System [1]. In conclusion, Web applications can be built in an evolutionary manner by composing SBBs and configuring them with DARs.

3. WEB-BASED PROCESS GUIDANCE

Problem Domain: Composing software products by assembling existing components is one of the challenging factors within EAI projects. While service composition for describing Web service choreography or orchestration plays a major role, the aspect of user guidance in Web applications is often neglected. We propose a DSL for describing such user guidance through various components of a Web application.

Domain Specific Model: Although the discussions about standardization of business process modeling (BPM) lasts over ten years, the absence of a commonly accepted interchange format still forms the main burden to business process management. Based on our experience during the process analysis in a university-wide integration project, we found a common denominator in using finite state machines (FSM). Our Process Guidance DSM utilizes common XLink attributes to formulate the behavior, traversal rules and its semantics. The ability to define all links in an external linkbase promises to cope with even more advanced concepts like personalization or even business process federations.

In the following, selected parts of the DSM are outlined in more detail. A workflow is described by a set of *workstepTypes* (1) that represent the states of the FSM and are instances of available Solution Building Blocks. They are realized as `locator` which form the connection to our technical platform [1] by specifying the `linkbase` and the `xlink:href` attributes.

```
<xs:complexType name="workstepType">                    (1)
  <xs:attribute ref="xlink:type" fixed="locator"/>
  <xs:attribute ref="xlink:title" />
  <xs:attribute ref="xlink:role" />
  <xs:attribute ref="xlink:href"/>
  <xs:attribute name="linkbase" type="xs:string"/>
  <xs:attribute name="final" type="xs:boolean" />
</xs:complexType>
```

The *transitionType* (2) expressed as `xlink:type="arc"` connects two states specified by `xlink:from` and `xlink:to` and activated by a `token`. In addition to behavioral aspects regarding the activation, `xlink:actuate="onLoad"` can be used to indicate *spontaneous transitions* to perform a continuous chain of worksteps without requesting a user for activation. In contrast, transitions that require a user input, are expressed with an `xlink:actuate="onRequest"`.

```
<xs:complexType name="transitionType">                  (2)
  <xs:attribute name="token" type="xs:NMTOKEN" />
  <xs:attribute ref="xlink:type" fixed="arc" />
  <xs:attribute ref="xlink:from" />
  <xs:attribute ref="xlink:to" />
  <xs:attribute ref="xlink:arcrole" />
  <xs:attribute ref="xlink:actuate" />
</xs:complexType>
```

Domain Interaction Model: In our project, all business processes were gathered by interviewing various stakeholders. The essence of these conversations was modeled with Petri nets using the INCOME [2] tool as editor. The business processes are described comprehensively, which means that Human Workflows, as well as service automation like choreography or orchestration, are covered. Obviously the resulting Petri net models are much more expressive than necessary to satisfy a FSM. Nevertheless we transformed the XML export of INCOME to a DSL program according to our DSM, which has a couple of advantages: on the one hand, reuse of already existing (partial) process information becomes possible, which increases the project team's productivity. On the other hand, the modeling tool is well known by the domain experts who describe the processes, so there is no extra effort for them to learn a new one. The illustrated scenario (cf. Figure 2) shows the support for students when they organize their course schedule by subscribing to courses. The Petri net business process acquired via the XML export of INCOME is transformed to a DSL program using XSL(T). Finally, a dedicated SBB executes this program to realize the workflow.

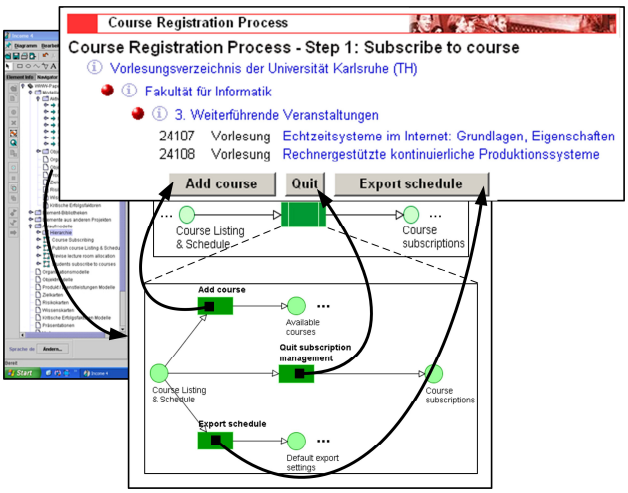

Figure 2: Editing the Web-based User Guidance DAR in INCOME with Petri nets.

4. REFERENCES

[1] Gaedke, M., Nussbaumer, M., and Meinecke, J., WSLS: An Agile System Facilitating the Production of Service-Oriented Web Applications, in Engineering Advanced Web Applications, S.C. M. Maristella, Editor. 2005, Rinton Press. p. 26-37.

[2] Lausen, G., et al. The INCOME approach for conceptual modelling and prototyping of information systems. in CASE - The 1st Nordic Conference on Advanced Systems Engineering. 1987. Stockholm, Sweden.

[3] Matera, M. and Comai, S., Engineering Advanced Web Applications, in Engineering Advanced Web Applications, S.C. M. Maristella, Editor. 2005, Rinton Press: Munich. p. 366.

[4] The Standish Group International, Extreme Chaos - Research Report (2001): http://www.standishgroup.com.