# Towards DSL-based Web Engineering

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
Strong user involvement and clear business objectives, both relying on efficient communication between the developers and the business, are key factors for a project's success. Domain-Specific Languages (DSLs) being simple, highly-focused and tailored to a clear problem domain are a promising alternative to heavy-weight modeling approaches in the field of Web Engineering. Thus, they enable stakeholders to validate, modify and even develop parts of a distributed Web-based solution.

## 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, Web Services, Conceptual Modeling

## 1. INTRODUCTION
Communication problems between the developers and the great diversity of stakeholders, faced especially in EAI projects, form a major roadblock to the efficient and reliable specification of a distributed Web-based solution. On the other hand, experience reports and research studies reveal the importance of clear business objectives and strong user involvement throughout the whole project lifecycle.

Within the last years, a variety of complex and extensive modeling approaches aiming at providing a basis for the formal and systematic specification of aspects of Web applications has emerged [1, 4] . Based on our experiences gained in several large-scale projects, this kind of modeling methodologies turned out to be a good means of specification and communication within the developer team. Regarding the strong collaboration with stakeholders, who are usually non-programmers with completely diverse educational backgrounds and work areas, they were too complex and too hard to learn.

In contrast to this, Domain-Specific Languages (DSLs) are small, simple and highly-focused specification languages for a clear and small problem domain [2], i.e. a specific aspect of a distributed Web-based solution. They employ well-known concepts, abstractions and notations derived from the problem domain and thus are easy to learn, understand and use, both by developers *and* stakeholders. Our vision of DSL-based Web Engineering is to empower stakeholders and domain experts to directly contribute to the development effort by validating, modifying and even autonomously developing DSL programs.


*WWW 2006,* May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

## 2. EVOLUTIONARY DSL FRAMEWORK
Figure 1 depicts the three slices of our evolutionary DSL framework approach. During a continuous evolution, the components of a DSL for a particular aspect of a distributed Web-based solution are being built or adapted (Conceptual Slice), they are applied to specify a part of the envisioned solution in terms of a DSL program (Logical Slice) and finally they are used to execute a DSL program (Physical Slice). Based on the experiences and requests for changes and improvements gained in an iteration, a new iteration is triggered. In order to be able to efficiently handle the emerging multitude of DSLs, we propose a Reuse Repository as the central storage for all DSL elements and associated metadata as well as a "DSL Librarian" team role.

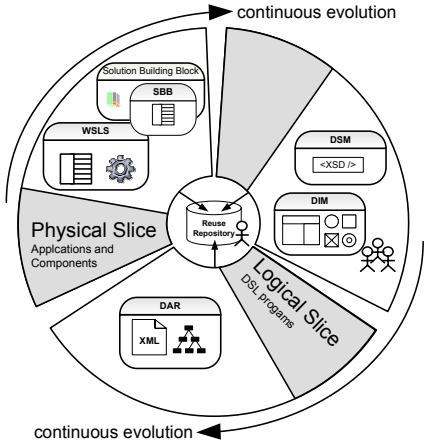


**Figure 1:** Overview of our evolutionary DSL framework

The core elements of a DSL are the Domain-Specific Model (DSM) and one or more Domain Interaction Models (DIM). The former represents the formalized schema (usually an XML Schema) of all solutions that can be specified within the DSL's associated problem domain. The latter, a DIM, is based on the DSM and provides dedicated (graphical) notations derived from and tailored to the DSL's problem domain. A DIM specifies the "editing notation" that is used to develop a DSL program – the Domain Abstract Representations (DAR). As it must strictly adhere to the DSM, a DAR is usually serialized into an XML document. In order to execute a DSL program, we use a so-called Solution Building Block (SBB). A SBB is a software component

whose behavior can be configured by a DAR. Web applications can thus be built by composing SBBs and configuring them with DSL programs. The WebComposition Service Linking System (WSLS) [3] serves as technical platform facilitating the systematic composition and configuration of SBBs.

# 3. DSL CATALOGUE

In the following, a selection from our DSL catalogue concerning the three important dimensions *navigation*, *data interaction* and *web-based process guidance* of an EAI project is presented.

## 3.1 Data Interaction

**Problem Domain:** In distributed web-based solutions, the integration of and interaction on various data sources, usually in form of web services, is a key requirement.

**Domain-Specific Model:** The DSM describing Data Interaction is based on the CRUDS primitives, i.e. accessing and modifying data via create, read, update, delete and search. Based on XPointer expressions referring to the WSDL data type specifications of the associated web service, the interaction modes on each data element can be defined.

**Domain Interaction Model:** In order to allow for a quick and intuitive configuration of the data interaction modes according to a web service's WSDL specification, we integrated the DIM in the WSLS framework [3] by customizing the Property Editor. Hence, this editor allows for the easy specification and selection of the desired parameters for the data interaction like the web service URL or the desired data type.

## 3.2 Web-based Process Guidance

**Problem Domain:** In advanced web applications, user guidance processes, i.e. the traversal of a set of application domains according to events triggered by the user, play an important role.

**Domain-Specific Model:** The concepts known from Finite State Machines represent the foundation of our DSM. The realization of our DSM is based on XLink and includes elements for specifying states (i.e. application domains), transitions and events.

**Domain Interaction Model:** As a first step, due to the characteristics and requirements found in one of our projects, we employed Petri nets as a DIM, just focusing on very simple and intuitive constructs.

## 3.3 Linklist

**Problem Domain**: Interconnecting application domains to build linking structures like menus or an index, is inherently required for web applications. In EAI projects, parts of an integrated system landscape are composed (i.e. linked) to new application domains through appropriate linking structures.

**Domain-Specific Model**: The Linklist DSM is based on XLink and provides concepts for structuring links and specifying the traversal behavior from the source to the target (embed or replace). The extract from the DSM schema shows the declaration of link types that connect application domains.

```
<xs:complexType name="domainType">                    (1)
 <xs:attribute name="linkbase" type="xs:string" />
 <xs:attribute ref="xlink:title"/>
 <xs:attribute fixed="simple" ref="xlink:type" />
 <xs:attribute ref="xlink:href" />
 <xs:attribute ref="xlink:arcrole" />
 <xs:attribute ref="xlink:role" />
 <xs:attribute ref="xlink:show" />
 <xs:attribute fixed="onRequest"
  ref="xlink:actuate" />
</xs:complexType>
```

An extract of a DSL program (i.e. a DAR) according to the DSM is depicted in the following code fragment.

```
<linklist>                                            (2)
  <level label="Organizational Services">
    <domain xlink:type="simple"
      xlink:actuate="onRequest"
      xlink:href="personalScheduleDomain"
      xlink:title="My Schedule"
      linkbase="self" xlink:show="embed" />
      […links to further application domains…]
  </level>
</linklist>
```

**Domain Interaction Model**: So far, we have developed a DIM which defines a graphical notation for a "Level", an "Internal Link", an "External Link" and a "Connector". While "Level" is capable of grouping a set of links and allows for defining hierarchies, the link types specify the link target and the traversal behavior. The "Connector" realizes the aggregation of links and nesting of levels. Figure 2 depicts the application of the Linklist DSL using the defined DIM symbols.

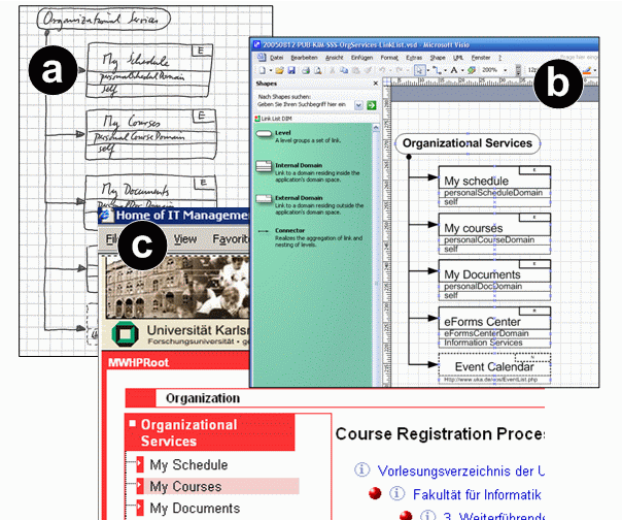


**Figure 2:** Application of the Linklist DSL: (a) pen and paper drafted DIM, (b) MS Visio DIM, (c) applied and executed DSL program within a portal based on the WSLS framework.

# 4. REFERENCES


[1] Ceri, S., Fraternali, P., and Bongio, A. Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. in 9th International WWW Conference. 2000.

[2] Deursen, A.V., Klint, P., and Visser, J., Domain-Specific Languages: An Annotated Bibliography. ACM SIGPLAN Notices, 2000. 35(6): p. 26-36.

[3] Gaedke, M., Nussbaumer, M., and Meinecke, J., WSLS: An Agile System Facilitating the Production of Service-Oriented Web Applications, in Engineering Advanced Web Applications, M. Matera and S. Comai, Editors. 2004, Rinton Press.

[4] Schwabe, D., Rossi, G., and Barbosa, S. Systematic Hypermedia Design with OOHDM. in ACM International Conference on Hypertext' 96. 1996. Washington, USA.