Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

WSLS: AN AGILE SYSTEM FACILITATING THE PRODUCTION OF SERVICE-ORIENTED WEB APPLICATIONS

MARTIN GAEDKE, MARTIN NUSSBAUMER, JOHANNES MEINECKE

Universität Karlsruhe (TH)

{gaedke, nussbaumer, meinecke}@tm.uni-karlsruhe.de

The process of application evolution in the Web poses a tremendous challenge for Web Engineering. Changing requirements demand solutions providing flexibility beyond mere adaptation on the surface. The shift towards agile systems results in advantages noticeable during the whole application lifecycle. By enabling a reconfigurable composition of the overall system, a reduction of development and maintenance cost can be achieved. In this paper we introduce the WebComposition Service Linking System (WSLS) that provides an agile approach to the Web application life-cycle. WSLS is a component-based system, which applies the service-oriented paradigm. It supports issues of composing, discovering and reusing services. Services are maintainable, manageable and configurable building blocks – as such they are a unit for reuse and for composition of Web applications. Evolution of the composed services is supported and guided in a systematic way by the underlying WSLS framework.

1 Introduction

Since the DotCom crisis and the consequent break-down of the stock exchange courses at the latest, the times when each new internet-based idea or technology led to “HYPE” and various expensive projects are over. More than ever, enterprises and organizations concentrate on their actual mission and emphasis the added value of their IT environments. This includes primarily optimization of associated value-added chains regarding supply and demand as well as cost-performance oriented criteria.

Due to the recent focusing on value-added chains, the long-proven concept of service-oriented architecture (SOA) received a significant upturn. SOA describes the collaboration of different tasks like *offering* a service by the service provider, *finding* the service by a service registry (respectively service directories or a service repository) as well as *consuming* the service by a consumer of the service. One reason for this boost is based on focusing the architecture and the integration of services rather than the technology used to realize a dedicated service. Although the SOA paradigm provided successfully the basic principle for different approaches like the component technologies CORBA and Microsoft’s (D)COM, the technological gap imposed by heterogeneous IT landscapes could not be bridged. Novel standards and recommendations adopted by the World Wide Web Consortium (W3C) like SOAP and XML-based Web Services [12] allow the realization of business processes in a technology-spanning and platform independent manner.

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

In order to manage the challenges imposed by this innovative service structure, development and operation must change from a monolithic, application-centered perception to a more flexible and integrative one, which considers management of information space and possible involved partners as its primary concern. In such a service structure, the development of dynamic, personalized and adaptable applications in a cost-effective and high-quality manner demands for a dedicated environment that we call Agile System. An *Agile System* supports the mass-customized production and evolution of applications by applying the service-oriented paradigm and management principles on flexible, integrative, and reusable application building blocks.

There exist several different approaches regarding the design and development of Web applications. Most of them are realized as frameworks, like for example WSRP (www.oasis-open.org), Cocoon (cocoon.apache.org), and SharePoint Services (www.microsoft.com). An important goal for these systems is to increase the degree of reuse within the Web application life-cycle. This can be achieved by component based or service oriented principles. The major difference between these two approaches lies in the way distributed functionality is provided. In the first case, this occurs by accessing a Web service, usually hosted by a third party. Components, however, are deployed once and subsequently supply their functionality on the owner's system.

The WebComposition Service Linking System (WSLS) was built on both principles and provides an evolution-oriented approach to Web application development with special emphasis on management, configuration, and composition of business functionality represented by services. Within the context of the project "Notebook University" (NUKATH) funded by the German government, we used this system as agile platform for supporting e-learning scenarios. These scenarios are treated as fundamental business processes (so-called *services*) that are made available in a Web portal. Consequently, the support of teaching and learning scenarios evolves to a management and configuration task.

In the following, we will discuss the WebComposition Service Linking System as foundation service-oriented Web application development. We will show how the system elements and core concepts of WSLS accomplish the aspects of an Agile System. Subsequently, we will outline the core elements of the system architecture of WSLS, presenting an overview of the correlation of management and configuration with services and security. In section 4, the application of the system in the context of the NUKATH project will be discussed as an example. Finally, we summarize and discuss our results and present directions for future work.

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

2 WebComposition Service Linking System

In this chapter we introduce the WebComposition Service Linking System, which is based on the WebComposition approach [6]. Beginning with an overview of the underlying model and concepts, we will explain how the aspects of an Agile System are mapped to single concerns – becoming primary units for reuse.

2.1 EvolutionBus Model and Services in WSLS

The WebComposition Approach identifies the development of an application domain as the initial stage of its evolution. An application domain is an abstract concept of the underlying application model and represents a container for business functionality (called services) and further domains. The EvolutionBus (cf. Figure 1, left hand) hereby represents this initial state of the application's evolution as an empty domain. It acts as a framework with the ability to modify itself by adding further domains, deleting domains, configuring a domain's behaviour, etc. This procedure conducts a Web application during its whole life-cycle [6]. Dedicated application domains integrate services in order to provide a desired functionality.

A *service* describes a generic service with dedicated business functionality. It forms a reusable application building block, which is applicable within different scenarios by adapting to prevailing requirements. Such scenarios could for instance include a local or temporal context, user defined configurations, security settings, privacy, etc. On the right hand of Figure 1 the fundamental classification scheme for services as supported in WSLS is depicted. The classification is based on the WebComposition Approach.

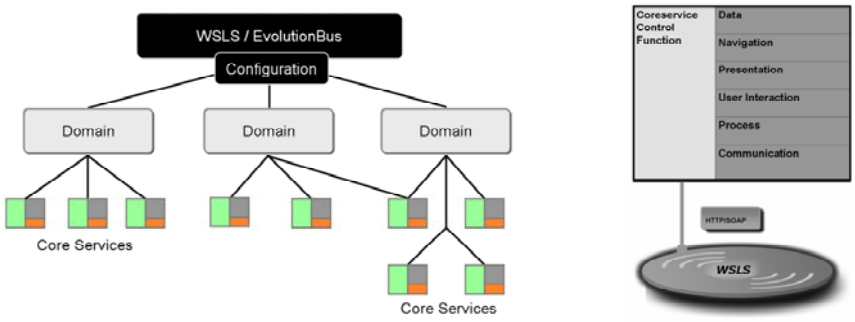

Figure 1: The EvolutionBus (left hand) aggregates and configures domains. Domains reuse services (right hand) to accomplish dedicated functionality.

A service is understood as a component that consists of a set of *service elements*. These elements are again components in the meaning of the component model introduced in [6]. Service elements perform dedicated tasks corresponding to their

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

classification. This classification is based on separation of concerns and thus a foundation for reuse of service elements. By separating concerns, a fine-grained consideration of the service element's tasks is made possible. In the following, the individual classified service elements and their collaboration coordinated by a control function is introduced in more detail.

The WebComposition Model supports object-oriented development [9] and therefore facilitates code-reuse by inheriting new services from existing ones. Additionally each EvolutionBus possesses a configuration that contains the properties defined in each domain, service and service elements within this EvolutionBus.

Control Function

Control functions form the backbone of a service, as they connect several service elements. They are a flexible means to delegate and organize the dataflow, integration and data processing of participating service elements and are responsible for the execution of the service functionality. A control function corresponds to the design pattern Mediator [7] and aims at achieving a preferably loose coupling. Thus, the reusability of a control function and particularly of service elements is dramatically increased.

Additionally this loose coupling facilitates a distribution of tasks within a development team and allows for applying different roles to develop specialized service elements. Hence new service elements for services can continuously be added or improved. Service elements are integrated by using standardized interfaces. These include for instance recommendations of the W3C concerning Web Services like the Web Service Description Language (WSDL) and the Simple Object Access Protocol (SOAP). In addition, efforts made by the Web Services Interoperability Organization to define rules for supporting interoperability between Web services have also been taken into consideration.

Data

The data of a service forms its fundamental unit of processing. All data considered conforms to a base object, which in turn corresponds to an XML Schema Definition (XSD). This object (`DataObject`) serves as a base class in an object-oriented fashion; its structure is kept generic. Metadata is considered to be attached at each data object. Therefore a data object posses a metadata object, which itself contains a complete set of Dublin Core elements as stated in [2]. Furthermore system specific Metadata is also provided.

Figure 2 depicts the general structure of `DataObject` and `Metadata`. Each `DataObject` contains a member `ServiceId`, which marks a unique identifier for

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

the surrounding service. As shown in Figure 2, specialized data objects can be created by inheriting from the base object class to fulfill its specific task. In order to guarantee a most flexible usage of metadata, extending the `Metadata` is also possible. Thus, each data object can provide specialized metadata if necessary.

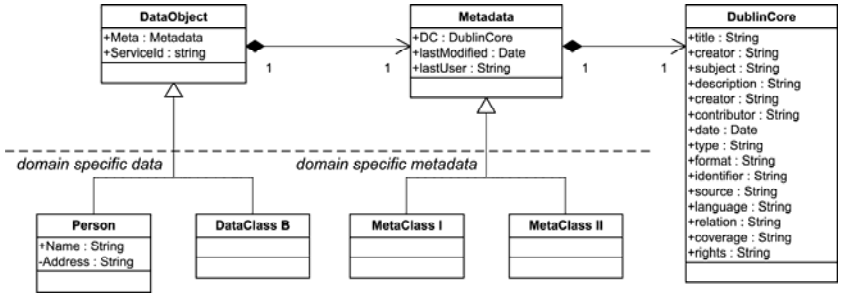

Figure 2: The general structure of data and metadata in a WSLS service.

In the project NUKATH we provided a specialized metadata object based on the Learning Object Model (LOM) [10]. The schema definition illustrated in Figure 3 describes the extended data object `Person` defined in Figure 2.

```
<schema targetNamespace="urn:wsls:Person">
<import namespace="urn:wsls:DataObject" />
<complexType name="Person">
  <complexContent mixed="false">
    <extension base="DataObject">
    <sequence>
      <element name="Name" type="xs:string" />
      <element name="Address" type="xs:string" />
    </sequence>
    </extension>
    </complexContent>
  </complexType>
</schema>
```

Figure 3: Schema definition of sub classed data object within a service.

Due to the widespread availability of web-based technology, in particular XML, arbitrary platforms can be integrated and used as data providers.

Presentation

This service element is dedicated to present the data of a service to users in an appropriate or personalized way. Due to the modular nature of a service, presentation elements can easily be adapted to the special needs of different user demands. Thus, it is possible to provide special presentations for blind or visually impaired people. Additionally, presentation rules and guidelines enforced by organizations can be

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

realized merely by focusing dedicated presentation aspects. Furthermore, the presentation of federated services among enterprises is supported. Each enterprise can easily adjust a service to its enforced presentation rules by reusing the service and exchange the presentation element. Hence the reused service functionality can be adapted in a fast and efficient fashion to local prevailing presentation guidelines.

Within the WSLS Framework, presentation elements are realized according to the design pattern Decorator [7]. As shown in Figure 4 presentation elements can easily be exchanged and "plugged in" to realize a dedicated presentation task. Due to the ability of *stacking* decorators, more general presentation elements like for instance the presentation of a data object's Dublin Core elements as illustrated in Figure 2, can be reused in several services.

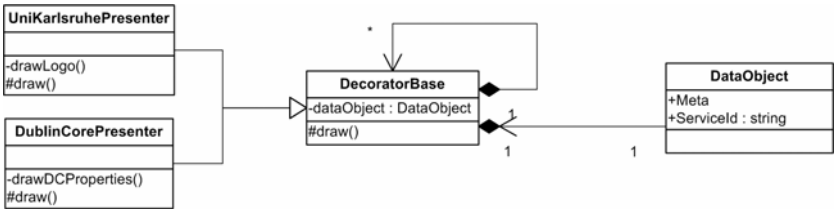

Figure 4: Presentation elements in the specification of the design pattern Decorator.

Navigation

All data within a service is treated as a set of data objects. A set-based view offers the advantage that a differentiated dealing with single nodes can be omitted; they are treated as sets as well. Navigation elements provide the structural linking and partitioning of a service's data set.

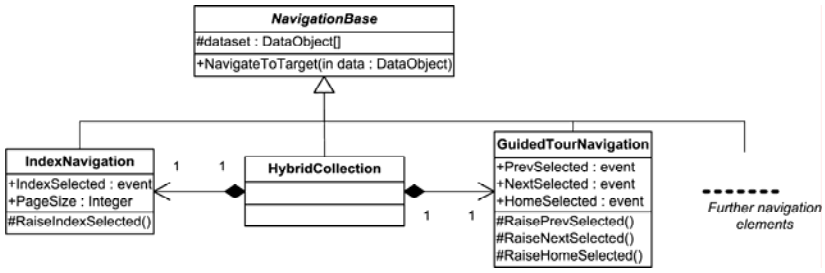

Figure 5: Different navigation elements within WSLS.

The realization of navigation elements is closely related to research conducted in the field of hypermedia and design patterns [8]. Figure 5 shows the three selected navigational design pattern *Index Navigation*, *Guided Tour Navigation* and *Hybrid Collection*. While Index Navigation provides a fast access to a group of data objects

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

via an index, a Guided Tour provides an easy-to-use access within the dataset of a navigation element by providing forward and backward links as well as a link to a home node. As stated in [11], a mixture of an Index Navigation and Guided Tour represents a solution in many cases. Isakowitz et al. call these elements a *hybrid*, which probably coined the name for the corresponding hypermedia design pattern *Hybrid Collection*. For a detailed explanation of these and other patterns confer [1]. Figure 5 illustrates how the hybrid collection is realized by reuse within WSLS. The traversing rules of navigation classes are realized by an event-driven model. If a link is activated, a corresponding event is raised and can be captured by interested parties. This is equivalent to the publish/subscribe communication paradigm [5]. To generate an event, a publishing navigation element calls its corresponding `Raise()`-operation (like `RaiseIndexSelected` for an Index Navigation). The event is propagated by the control function to all relevant subscribers; it can thus be viewed as a proxy for possible subscribers. Subscribers are elements that participate in the navigation process, e.g. the presentation element that renders a navigated data object. We denote this with the term *event-based navigation*. Navigation elements can also define adaptable properties. Thus, the property `PageSize` in the `IndexNavigation` class can be adjusted to a user's individual needs in order to specify the number of accessible objects provided through an index.

User Interaction

The service element *user interaction* comprises all actions that can be performed by a user of a dedicated service. It specifies the behaviour of a service towards the user, in other words it is responsible for the access of the user to the elements of the information space of the Web application.

Each representative within an interactive proceeding is called an *interaction carrier*. In Web applications, interaction carriers are for instance hyperlinks, buttons, etc. The separation of such an interaction carrier from a presentation element is a difficult task and would complicate the design phase of presentation elements if they were divided completely. In WSLS, this issue is solved with the introduction of interfaces that specify the behavioural rules of an interaction carrier (cf. Figure 6). Primarily, these carriers have no intended behaviour; they act just as normal presented entities. A presentation element that satisfies a corresponding user interaction interface assigns the behaviour and allows for wiring the action invoked and the data object. This procedure conforms to the design pattern Command [7].

Figure 6 shows two families of user interaction interfaces supported in WSLS. Common interaction interfaces address general issues like creating, deleting and editing data objects. These interfaces are predefined because of their general character. The second class can be used to create arbitrary behaviour and thence interaction in the purpose of the design pattern Command. After a command is raised

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

the corresponding interaction can be activated, e.g. by calling action methods or showing a dedicated form to manipulate data.

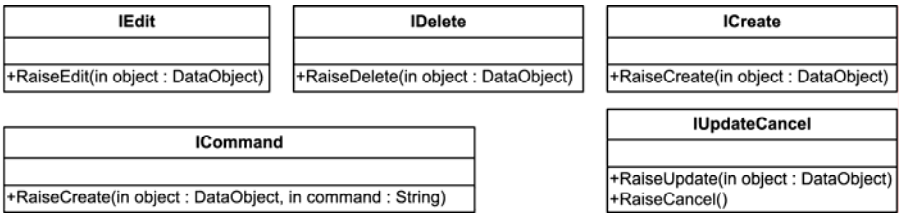

Figure 6: User interaction in WSLS: interfaces assign the corresponding behavior.

Process

The service element *process* specifies the data processing steps necessary for a service. This covers the linking of external workflows to integrate further business functionality. Web services represent a secure and platform independent technology for connecting such external functionality. Legacy Systems can be equipped with additional Web service interfaces to enable their inclusion within WSLS services. Hence, the protection of investment in existing systems can be guaranteed. Focusing on open standard interfaces simplifies future integration of external functionality into services and provides an additional potential for cost reduction.

Communication

The service element *communication* provides a service with communication related aspects addressing protocols, security, caching etc. In order to enable a service's seamless deployment into different environments, the communication with heterogeneous systems requires special consideration. Among proprietary protocols like the local file system, standardized protocols like HTTP, SOAP or WebDAV (Web Distributed Authoring and Versioning) [12] are supported.

3 System Architecture

Service oriented architectures have great potential to meet the challenges imposed by the fast evolution of business rules and corresponding components. A system for developing reusable components is also challenged to support mechanisms for discovering, publishing and exchanging such components and/or services. WSLS is designed for meeting these challenges by introducing an appropriate architecture that is open and agile in the sense of the mentioned definition for Agile Systems. Figure 7 depicts the overall architecture of WSLS. It shows the main sections of this Agile System: (Evolution-)Management and Configuration of domains, publishing and discovering services through a registry, as well as securing the system.

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

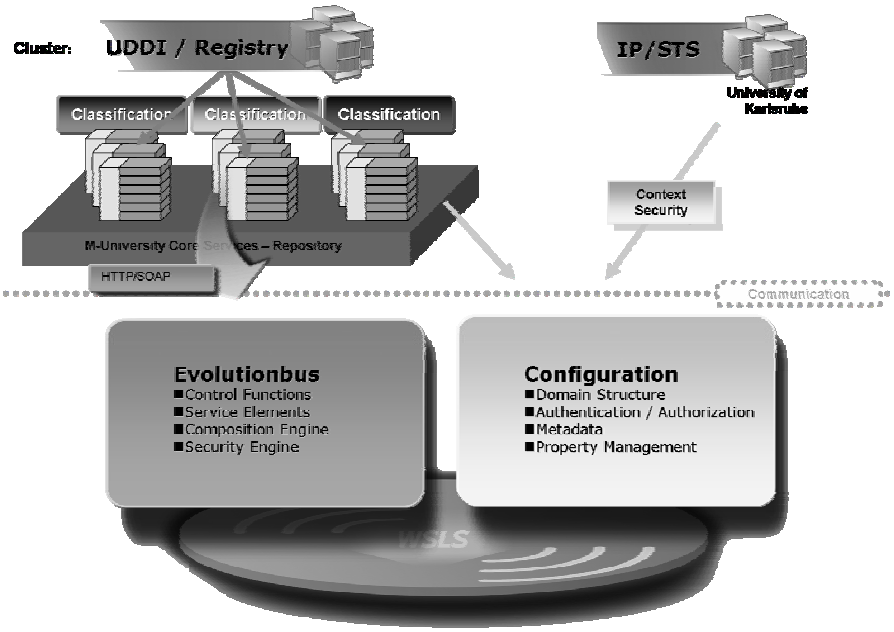

Figure 7: The overall architecture of WSLS in the context of NUKATH.

In a distributed environment where a loose coupling between services is preferred, there is an essential need for finding such services. WSLS supports the classification of services by means of categorization schemas as defined by UDDI [4], which provides a registry that facilitates and standardizes service publishing and discovering. The physical locations or endpoints of the different services and service elements form a distributed reuse repository. The combination with UDDI now allows finding services and service elements in the repository by a variety of classifications.

For the purpose of federating business processes, WSLS provides support for federated security and identity management. The concept of single-sign on (SSO) is supported using the Passive Requestor Profile, which is part of the WS-Federation specification [3]. This means that the task of identifying a user of the system is delegated to an external Identity Provider (IP). The identities of different groups of users can be managed by different IP's, eliminating the need for a single authority knowing all users. In the case of a required authentication, a Security Token Service (STS) is consulted. This service allocates the right IP for the requesting user, checks the encrypted security token received from the IP and finally issues the security tokens necessary for accessing any protected resources.

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

4 Application of the Configuration Principle in NUKATH

WSLS was designed as an Agile System allowing Web applications build on top of it to be prepared for changes and for continuous evolution. In this section we will discuss an example, taken from the NUKATH e-University learning environment we developed for the German government.

Initially an application domain must be created. Subsequently the domain is added to an existing EvolutionBus (which might be empty in its beginning) to provide the desired functionality. The creation of an application domain involves the selection of a corresponding control function and the selection and configuration of appropriate service elements. In WSLS each domain has at least a unique identifier and an associated control function. For the subsequent example the selection of a control function that provides functionality for aggregating RSS Feeds (*R*eal *S*imple *S*yndication) is assumed.

Adding a new domain with an associated control function causes the system to check if there are required properties or service elements that must be configured. In the case of the RSS domain there are two required properties:

1. A collection of Urls (Uniform Resource Locator) that specify the location of the data objects - here the RSS feeds *and*
2. a presentation element, that is capable of presenting the RSS data.

After these steps the business process configuration is complete. In our case of the NUKATH environment, we applied this business functionality to many different scenarios for providing students fast-access to new and federated learning material. Figure 8 depicts three different configurations for an RSS service with respect to the presentation concern. On the right side a grid presentation is used in combination with a window (Boxed Decorator) and an element that is capable of rendering RSS. In the middle of the illustration the RSS Feeds are presented in a style conforming to the prescribed styling rules from the University of Karlsruhe.

In order to provide access to visually impaired and blind students the RSS feeds might be “displayed” using a Braille keyboard or as in this example, the “RSS Braille Presenter” can be assigned to render the RSS data in Braille Code on the Web page. Obviously it has no direct impact as “visual” component - but used in combination with appropriate devices, like e.g. a Tiger Printer, the information can even be printed on special paper for visually impaired and blind persons.

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

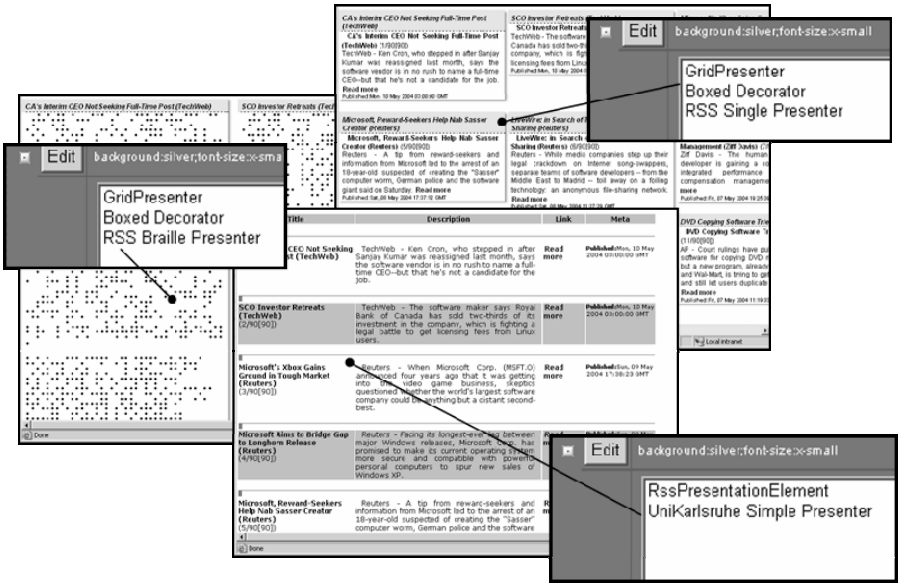

Figure 8: RSS Feeds rendered with different configurations of presentation elements.

5 Status and future work

In this paper, we introduced the WebComposition Service Linking System, which applies the service-oriented paradigm to the reuse-driven development of Web applications. As an Agile System, it supports issues of composing, discovering and reusing services. Composition of the application building blocks is supported by the concept of application domains and the EvolutionBus model. Discovery, Integration and management of services are realized by UDDI-Registries, categorization schemas in combination with Reuse Repositories. Reuse is facilitated by the idea of configuring instead of programming services. The architecture of WSLS allows the integration of legacy systems based on well-defined interfaces. Furthermore, the incorporation of modern standards, like WS-Federation, enables the federation of services transcending organizational boundaries.

The vision of the WSLS project is to support the whole life-cycle of possibly federated Web applications from the initial phase until and during operation. Therefore current work relates to the definition of a process model that takes identity management and semantic relationships between enterprises and services into account. The new method will therefore extend the WebComposition Process Model and provide model-driven support for development and operation of federated systems. The WSLS Framework is the initial step towards our vision.

Engineering Advanced Web Applications, M. Matera, S. Comai (Eds.), Rinton Press, Princeton, New Jersey, USA, Pages 26-37, ISBN 1-58949-046-0, 2004

Acknowledgement

This material is based on research for the NUKATH project initiated by the German Ministry of Education and Research (BMBF) and continued in the project Mobile University supported by Microsoft Research Ltd and Microsoft Deutschland GmbH.

Further information on WSLS: http://wsls.net

References

1. Acm-Hpr, Hypermedia Patterns Repository (HPR) - WebSite (1999): http://www.designpattern.lu.unisi.ch (23/03/99).
2. Andresen, L., Dublin Core Metadata Element Set, Version 1.1: Reference Description. 2003, Dublin Core Metadata Initiative (DCMI).
3. Bajaj, S.E.A., Web Services Federation Language (WS-Federation) - 2003), MSDN: http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-federation.asp.
4. Bellwood, T., et al., UDDI Version 3.0. 2002, UDDI.org.
5. Eugster, P.T., et al., The Many Faces of Publish/Subscribe. ACM Computing Surveys (CSUR), 2003. 35(2): p. 114 - 131.
6. Gaedke, M., Komponententechnik für Entwicklung und Evolution von Anwendungen im World Wide Web. Dissertation an der Universität Karlsruhe. 2000, Aachen: Shaker Verlag.
7. Gamma, E., et al., Design patterns: elements of reusable object-oriented software. Addison-Wesley professional computing series. 1995, Reading, Mass.: Addison-Wesley. xv, 395.
8. Garzotto, F., Mainetti, L., and Paolini, P., Hypermedia design, analysis, and evaluation issues. Communications of the ACM, 1995. 38(8): p. 74-86.
9. Gellersen, H.-W., Wicke, R., and Gaedke, M. WebCompostion: an object-oriented support system for the Web engineering lifecycle. in 6th Intl. World-Wide Web Conference. 1997. Santa Clara, CA, USA.
10. Hodgins, W. and Duval, E., Draft Standard for Learning Object Metadata - Draft Standard (2002), Institute of Electrical and Electronics Engineers (IEEE): http://ltsc.ieee.org/wg12/.
11. Isakowitz, T., Stohr, E.A., and Balasubramaninan, P., RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM, 1995. 38, No. 8 (Aug. 1995): p. 34-44.
12. World wide web consortium, W3C Technical Reports and Publications - 2004), http://www.w3.org/: http://w3.org/TR/ (17.09.2004).