WebComposition Service Linking System: Supporting development, federation and evolution of service-oriented Web applications

Martin Gaedke, Martin Nussbaumer, Emma Tonkin

IT-Management and Web Engineering Research Group,
Institute of Telematics, University of Karlsruhe,
Zirkel 2, D-76128 Karlsruhe, Germany,
E-Mail: {gaedke | nussbaumer | tonkin}@tm.uni-karlsruhe.de

Abstract. There exists a need within many large organizations and their partners to operate cross-organizational Web applications. This paper introduces the WebComposition Service Linking System (WSLS), a component-based and service-oriented system which makes extensive use of Web Services and other standardized Internet technology in order to support development, maintenance and management of reusable and configurable components for cross-organizational Web applications. A real-world application based around the WSLS support system which provides globally accessible core services, NUKATH -- Notebook University Karlsruhe(TH) -- is introduced. Our NUKATH project's vision is that business processes of universities shall be established, supported, and provided by aggregations of high-quality, ubiquitously accessible, federated services. These federation-aware services could thence be the basis for reusable and standardized business building blocks of a university. Furthermore, these services could transcend the "business" borders of a university – thus allowing for unique meta-structures of distributed universities supporting new collaboration scenarios in education and research.

Keywords: Web Service, Reuse, Composition, Federation, Evolution, Component-based Web Engineering

1 Introduction

Effective Web Engineering strategies within any large organizational structure such as an enterprise or university have a well-understood dependency on effective underlying technologies. A frequently embraced example of such technology is the provision of reusable, configurable components [1, 2]. This constitutes a replacement of previously used ad hoc methods and itself stems from the field of software engineering [3] – it is often considered axiomatic to the field of Web Engineering that reusability is a key concern. Recently, much work within the Web Engineering community has focused on Web Services [4], a standards-based and highly interoperable method of invoking remote procedures, an example of which would be SOAP over HTTP [5]. They provide a powerful approach for the development of new kinds of Web applications that are built by composition of distributed components – typically, any given Web Service exists within a known environment; authentication, data storage and so on are therefore dealt with on a local level, within the parent organization. However, one may easily imagine scenarios in which unique services provided by any given organization have value to Web applications of related

organizations and it is therefore desirable to share – or rent – these services. In this case, one could describe the desired situation by imagining a unique repository of shared Web Services from which web applications may be assembled [6]. This scenario presents difficulties; consistent authentication, cross-organizational relationships, or data/business semantics must be compatible across the various organizations, for example.

This argument suggests that Web Services alone in the context of cross-organizational Web applications would be insufficient for many problem sets when deployed on a larger scale. In this case, much of their potential stems more or less directly from the addition of *federation* [7]. Federation has been qualitatively defined by IBM and Microsoft as: *The technology and business arrangements necessary for the interconnecting of users, applications, and system. This includes authentication, distributed processing and storage, data sharing and more.*

This paper is structured as follows; we begin with a brief résumé of our previous work in the field and the concepts that underpin its design and operation. Following this, we continue by describing the concepts, results and practical decisions behind the evolution of our work from the time of its conception to the present day. We introduce our current framework, WSLS, and its application within the university context, NUKATH. We proceed to identify a complex workflow process within the context of university administration, of which we then demonstrate a model and solution. Finally, we conclude with a number of brief comments about the future direction that we envisage for our research.

2 WebComposition

Our approach to Web Engineering is fundamentally based on the choice of supporting web applications throughout the lifecycle at the level of fine-grained objects rather than the course-grained level that file-based resources represent. This approach [8], introduced in 1997, was christened WebComposition; the objects to which it refers are themselves known as web components. A practical implementation of this model was also developed, the WebComposition system, designed as an open platform for Web Engineering tools rather than a self-contained engineering environment.

The initial incarnation of this approach was the WCML system, based on the Web Composition Markup Language [9]. This approach proved to be viable in the sense of being a powerful toolkit and a great deal of support for additional features such as WebDav [10], SOAP were added [11] over time. WCML was first applied in an appropriately-scaled real-world environment during development of a world-wide e-procurement system for Hewlett-Packard [12]. In practice, however, the freedom given to the developer was sometimes too much, since WCML unbound was too powerful and therefore appeared complex to the inexperienced. The learning curve involved simply proved to be too steep in certain circumstances.

2.1 Separation of concerns

WebComposition is designed to force separation of the various elements and towards this end applies the architecture of Service composition as introduced by the EvolutionBus approach [13]. The Service architecture separates clearly and

consistently, as shown in Figure 1 and focuses on simplification of design, conceptualization and operation.

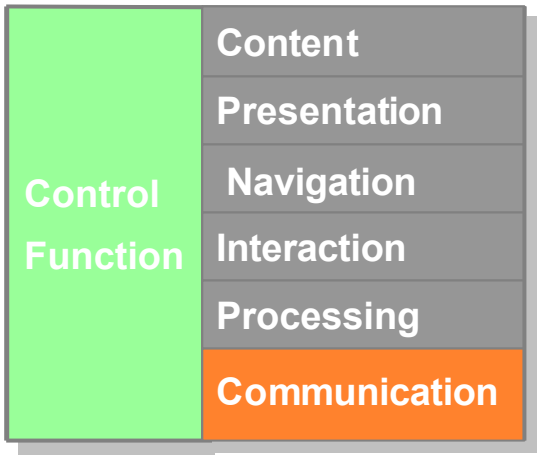

Figure 1 – Separation of concerns within a WebComposition Service

Each of these layers is defined as a *service element*, an abstraction for the purpose of separating concerns. There are six of these contained within the model; content, decoration, navigation, user interaction, process and communication. More conceptually interesting than these points alone is the *control function*, the 'glue' that merges the various layers together.

A simple example of the control function in action follows; the below represents a sample snippet that describes the definition of a services by configuring the required service elements in XML and binding them to the responsible control function (defined by the CFURI).

```
…
<eb:service cf="CFURI" xmlns:eb="urn:wsls:evolutionbus:service">
  <eb:property name="Content">ContentFile.xml</eb:property>
  <eb:property name="Presenter">BookPresenterURI</eb:property>
  <eb:property name="Navigator">IndexNavigatorURI</eb:property>
</eb:service>
…
```

This configuration code and others like it are initially read by the EvolutionBus system, which makes note of the control functions mentioned in the code and runs them using the parameters provided; in this case, the CFURI control function is designed to operate on the source file "ContentFile.xml" and uses the BookPresenterURI service element for presenting the content and providing an index-based navigational context by using the specified navigational service element (as provided by the IndexNavigatorURI).

2.2 EvolutionBus

WebComposition is based on an object-oriented model for web applications. According to this model, a web application is hierarchically composed by *components*. At higher levels in this hierarchy, a component may model not only a page or even a site, but also application-specific concepts such as a document or a dialog consisting of a number of inter-related pages. Further down the hierarchy, components relate to parts of pages, such as for example tables, table entries, anchors and referenced dynamic resources; components at this level may also represent application-defined abstractions for page organization, for example *Overview section*. The leaves in the component hierarchy are called primitives, the other components being composites. This model does not prescribe the grain of primitives. For example, a text consisting of several paragraphs could be modeled as a primitive provided that the text as a whole represents the basic unit of manipulation.

In the WebComposition approach the basic architecture of a web application is described with the term *EvolutionBus*. This initializes functions controlling all application domains of a web application. It enables management and collaboration of domain components, which implement specific application domains such as procurement or reporting.

The evolution for which it is named may take place in two ways:

- Domain specific evolution – defined as the extension of a domain through new services, e.g. adding a new service to domain by adding a service configuration specifying the service elements to use (Cf. XML example shown above).
- Evolution of the domain set – the evolution of an application through the modification of the domain set, e.g. extension of an application's functionality by adding a new application domain. An example of this could be the addition of a procurement domain, with support for a shopping basket and corresponding functionality permitting items to be ordered from a Web-based product catalog (which could exist independently of the procurement domain in the sense of a product catalog domain).

At the present time, we are working towards extension of the EvolutionBus with the concept of federation. The EvolutionBus implementation is already capable of a form of federation, as part of its WebComposition heritage, and this will be extended to form a practical framework.

3 WSLS support system

The next generation framework of the WebComposition approach, known as *WSLS* (WebComposition Service Linking System) and pronounced *Weasels*, was designed to force the developer to operate in a systematic way. Towards this aim, a number of changes were made. A move was made to a restrictive *component-based* concept. WSLS guides the developer more directly in the sense that development with the system is deeply rooted in a good understanding of the abstract (EvolutionBus-based) architecture [13]. Separation of concerns (eg, navigation, content, presentation) is now strictly embedded into the WSLS system whereas WCML merely provided guidelines to this effect; this stricter control should lead to a more consistent development effort. The service-based component model used implies that WebComposition services are

built upon distributed components. The lack of clear control exerted by WCML is not limited to the issues of systematic separation of concerns; other issues, such as "best-practices" for user interaction or presentational issues like accessibility, also fall within this category and will be addressed in future WSLS development.

3.1 Supporting the WebComposition approach

WSLS is effectively an implementation of the WebComposition approach. The WSLS framework provides a shared platform; it may be perceived as a framework but also provides a number of support services and basic functions. Primary amongst these services, lying at the core of the system, is the EvolutionBus. In the same way that the microkernel for an operating system provides basic operations, so does the EvolutionBus for the WSLS system.

This includes support required for developing all elements of the service architecture as described in section 2.1 as well as for basic operation of the services and domains. However, it is not exhaustive. Additional elements are, notably;

- Security and DRM, to support authentication and authorization also in the future context of federated WSLS-based applications [14-16]
- Dedicated support for remote services and connecting EvolutionBuses using Web Services

The UML class-diagram in Figure 2 describes an abstracted form of the WSLS service definition as defined by the WebComposition approach (Cf. Figure 1). The term 'domain' refers to a logical artifact (an area on a web page). Within the domain, the control function involved acts to define the nature/structure of sub-domains or service(s) visible within the domain. A domain including only services and service elements, but which includes no other domains, is referred to as a container.

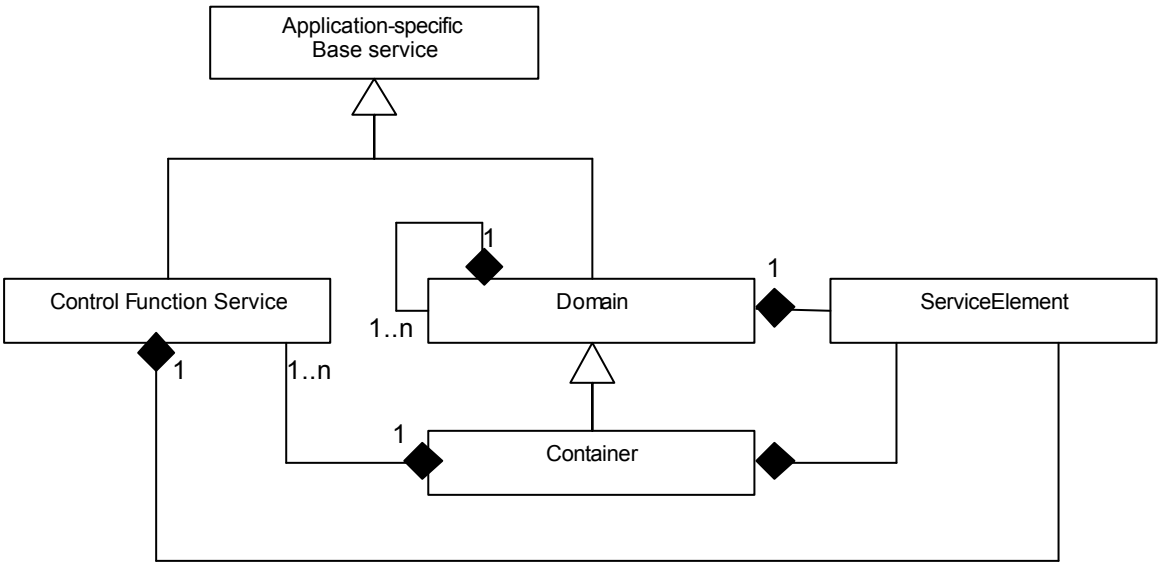

Figure 2 – WSLS service definition/abstraction

A domain consists therefore of sub-domains or containers. A simple example of a container is a *grid container*, which simply shows every service held within it in a grid layout. A more complex container such as one that controls a shopping process

includes logic destined to tacitly control the process (eg. by showing services appropriate to the current context within the process). Domains and container domains both include the various service elements.

The WSLS implementation uses an event-based approach [17], which is to say that an event, once raised, calls a chain of tasks which must be resolved by the control function and delegated to responsible service elements for further operation. An example of this is the event-based navigation; clicking on a link raises an event that invokes the associated navigational service element. Although web user interfaces are quite different from more traditional interfaces, passing events remains an effective strategy – particularly given the WSLS design focus on configurable, reusable, component-based systems. This is analogous to more traditional dialog-based development environments such as Visual Basic and company.

The following Figure 3 shows a simplified example of domain architecture consisting of presentation, content and navigation. The domain (here a container) itself has three child services, Services A to C. As the diagram shows, the decorator element provides a placeholder for the services as well as defining decoration in a literal sense. The navigation element, in this case a simple previous/next navigation, joins pages A to C. The control function responsible for the operation of the domain handles incoming events and delegates necessary tasks to the service elements of the domain.

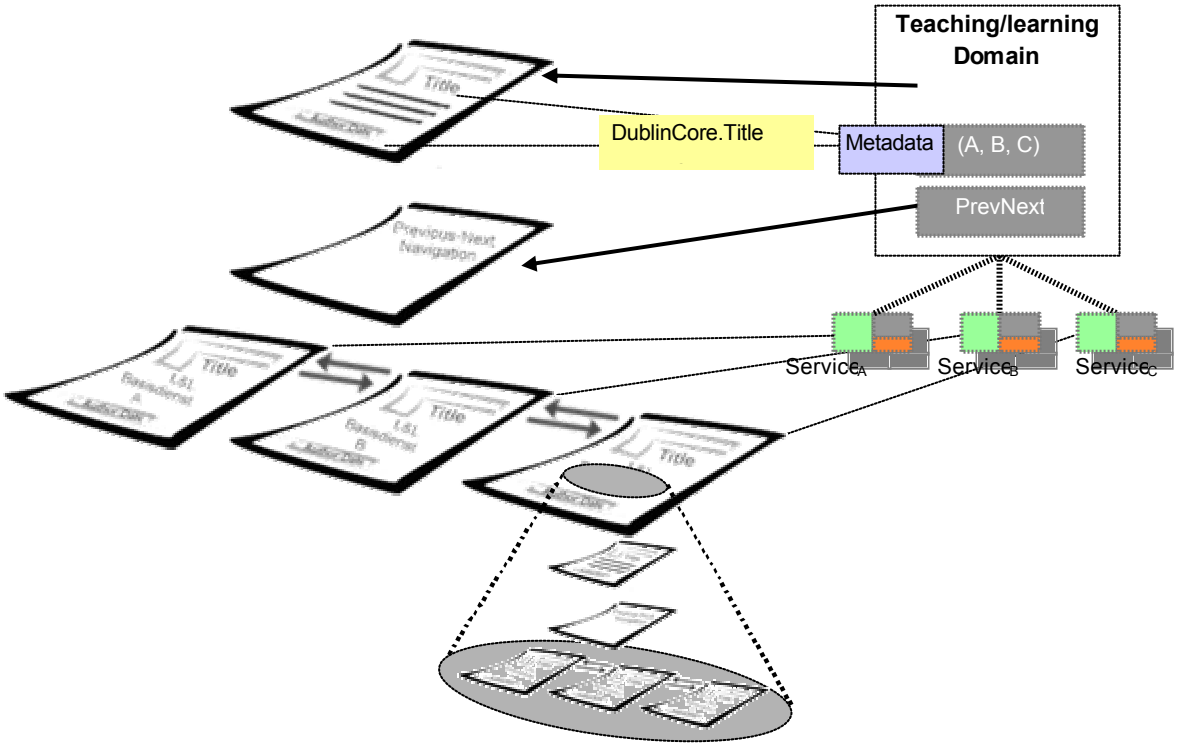

Figure 3 – A simple domain example within WSLS

3.2 Effective, flexible modeling of security and authentication

It has often been remarked upon in industry that role-based authentication is an effective and easy modeling system with which to model business needs and workflows. The security model implements a form of role-based authentication. It

provides the ability to bind tickets to user accounts. This is equally usable above accounting systems such as databasing, LDAP, .Net passport; it is effectively an adaption layer. 'Flexibility' also implies the ability to work in a distributed way, permitting federated services, in which this model succeeds.

A brief synopsis of the security model is as follows; the WSLS implementation includes a user management system, which assigns *roles* to users according to context – for example, a visitor to a web page may be simply a visitor, a student, professor, or an administrator. Within each role, the user will be assigned one *ticket,* the task of which is to provide information of the user's permissions within the context of the role, thus separating the visitors further into ticket-defined "sub-roles". As an example, imagine a faculty bulletin board where lecturers are given read, write and delete access across all boards and students are permitted to add new comments and read existing ones. The professor is provided with a ticket of unlimited duration certifying his/her status, whilst the student is given a time-limited student ticket that expires regularly, for example, at the end of each semester, at which time it has to be renewed by the student office. Given this basis, XML-based configuration then permits the site administrator to grant permissions as defined above:

```
…
<service name="test" cf="eventsURI"
         xmlns='urn:wsls:evolutionbus:service'>
     <grant permission="View" role="GUESTS"/>
     <grant permission="View" role="AUTHENTICATED"/>
     <grant permission="Modify" role="Professors"/>
     <grant permission="Delete" role="CreatorOwner"/>
     <grant permission="Add" role="Students"/>
     <property name="Content.WebService.Adaptor.tmReader">
       http://mw.tm.uni-karlsruhe.de/services/events/events.asmx
     </property>
     <property name="Presenter">EventPresenterURI</property>
</service>
…
```

This service grants permissions according to the status of each entry in the bulletin board events service; the Creator/Owner role refers to the author of a given event. Note that the 'Modify' permission refers to "Add + Delete + (literal) Modify".

It is notable that, firstly, the ticket provided to the user narrows the definition of the role, secondly, that the tickets themselves may legally exist by default independently of the role in which they are provided. One may conceive of a 'separate' subclass of tickets which, by means of containing additional information or appropriate attributes (such as a signed hash value), become suitable for use with Web Services – a security token. The logic contained within the operation of tickets may be of arbitrary complexity, and it is therefore possible to implement sophisticated rules, including DRM-style systems depending on complex networks of multiple servers [18].

3.3 Overview of the WSLS-architecture

As shown below, the WSLS-architecture provides a supporting system and basic framework upon the EvolutionBus foundation, including domains, container, services, service elements, security services and metadata. The web application specific services in the upper layer are built upon the WSLS framework. They may, however,

also include elements from remote services such as RSS news feeds or the Google Web Service provided by Google Inc.

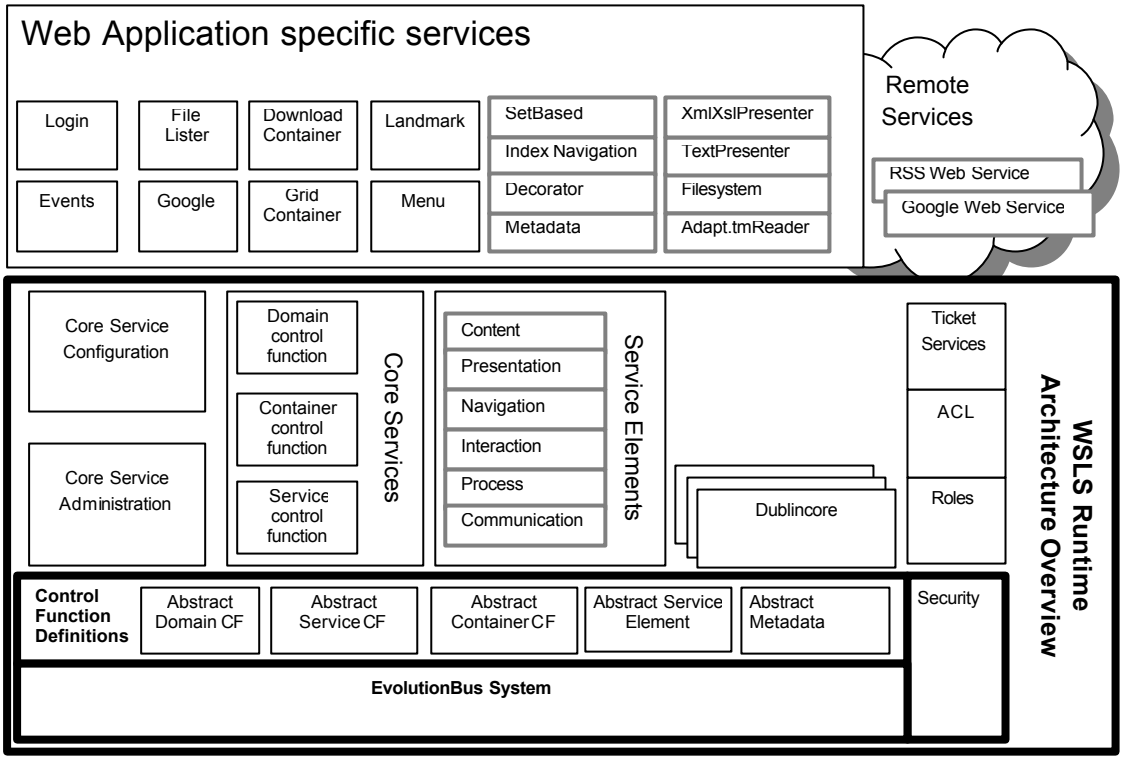

Figure 4 – Architectural Overview of WSLS

4 NUKATH: Notebook University Karlsruhe (TH)

The following discussion of the WSLS support system exists in the context of the NUKATH – Notebook University Karlsruhe (TH) – project, based around WSLS, which provides globally accessible core services for education and knowledge transfer. The NUKATH project vision is that business processes of universities shall be established, supported, and provided by aggregations of high-quality, ubiquitously accessible federated services. These federation-aware services could thence be the basis for reusable and standardized business building blocks of a university or organization and due to their federated nature may even provide the ability to transcend the borders of one organization, permitting unique meta-structures of distributed universities to be constructed. One might perhaps here stress the point that services, basic or otherwise, are by no means restricted to being local in nature and are designed in order to promote rich, open interaction.

WSLS is the provider of the basic functions (WSLS services and domains), whilst NUKATH is the provider of the application-specific services, an implementation designed specifically for university teaching and learning scenarios. Note that the NUKATH 'meta-structure', which unites the various teaching and learning domains, works in this way as an effect of the underpinning EvolutionBus implementation. Returning for a moment to the tree structure as indicated by the UML diagram view previously mentioned (Cf. Figure 2), one can relate the basic services in Figure 5, below, to the leaves in the WebComposition component hierarchy.

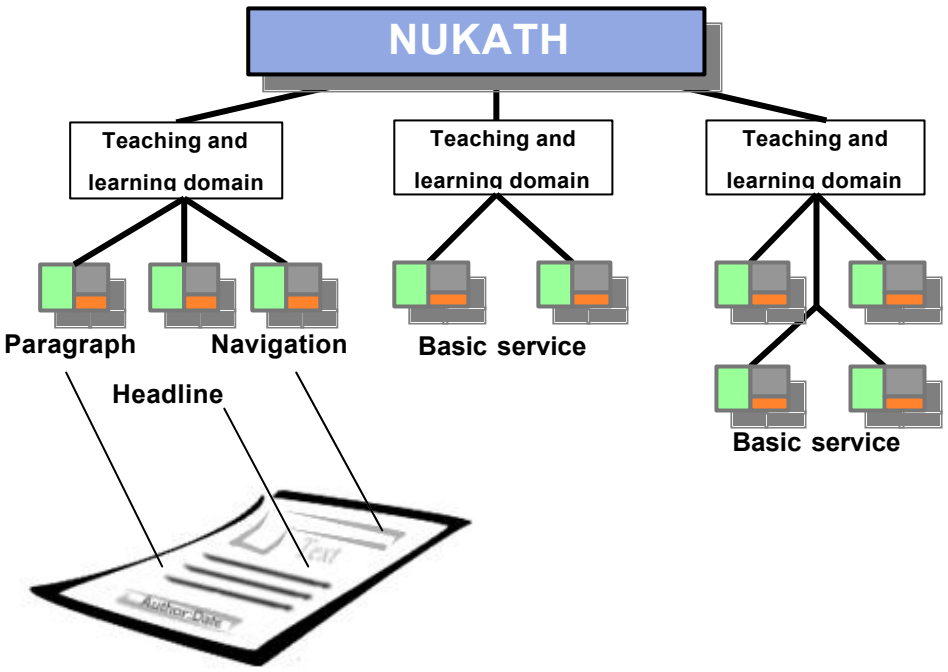

Figure 5 – EvolutionBus architecture applied to the NUKATH environment

WSLS itself includes a number of control function base services, which are characterized by their simplicity; NUKATH adds a number of application-specific services related to e-learning, knowledge transfer, administration and business processes.

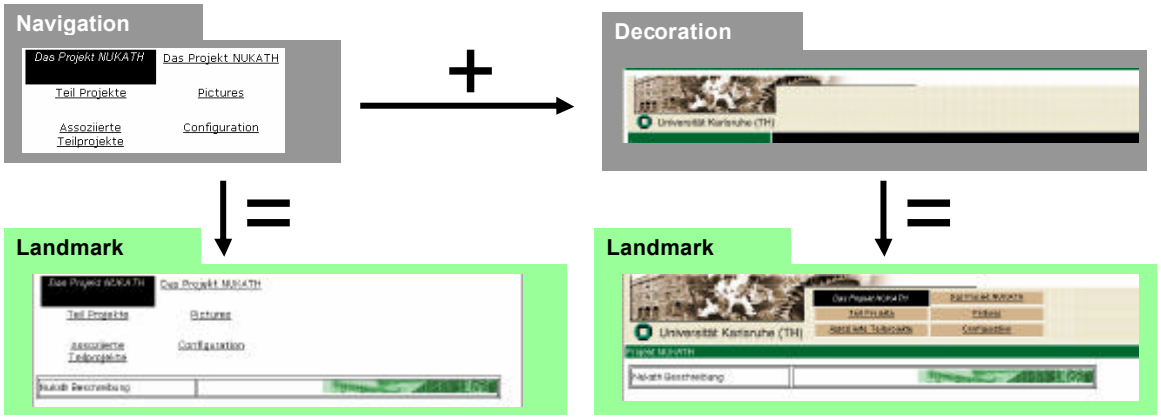

Figure 6 – Merging navigation and decoration to create a landmark

Figure 6 shows a real-world example of the merging between navigation and presentation (provided by a decoration component) elements in an example domain, plus the application logic that controls the landmark (e.g.: to list as the landmark pattern requires, this internally requires the inclusion of index navigation). Recall from 2.1 that there are several layers not explicitly shown in this example, such as interaction, process and so on; these are inherent from the domain except where specified otherwise; e.g. content may be given inherently from the domain

configuration (an XML document for a web application describing domains, container, services and their relationships).

5 Sample NUKATH workflows

We have chosen to concentrate for the purposes of this paper on two processes taken from the NUKATH project: firstly, a subscription system, and secondly, the workflow process that surrounds a student's progress through a semester worth of lectures.

5.1 Subscription system

The subscription system is initially developed for use at the University in the context of enrolment on a *four* month practical course, where students are subscribed for the full *three* months. Following the end of the practical course, the subscription assigned to the student changes to indicate a 'passed' status. This permits them a different form of access to the site which shows, for example, possible thesis topics and suggested further reading. When used elsewhere, a number of different functionalities are required. As an example, more commercial environments may need to offer subscriptions on a monthly basis, or to sell 'ticket books' providing a number of accesses. These requirements are easy to model; consider Figure 7.

Bear in mind, however, that the simplicity of this model implies that a great deal of the innate complexity of the problem has simply been redefined such that the tickets models the required business or organizational rules. The ticket logic may be relatively complex, including the application of cryptography standards and the use of intermediate servers (as in the case of DRM systems) [15, 19].

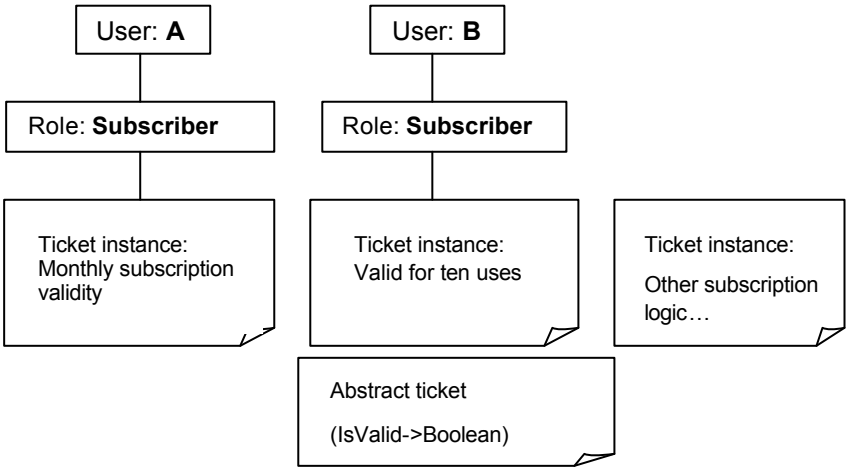

Figure 7 – A simple subscription example

The following lecture/exam process is somewhat more complex, as described in Figure 8, but the process can nonetheless be managed entirely by correctly modeling the roles and tickets. The workflow process is described on the left, whilst on the right is a model of the process showing the management of the workflow using tickets.

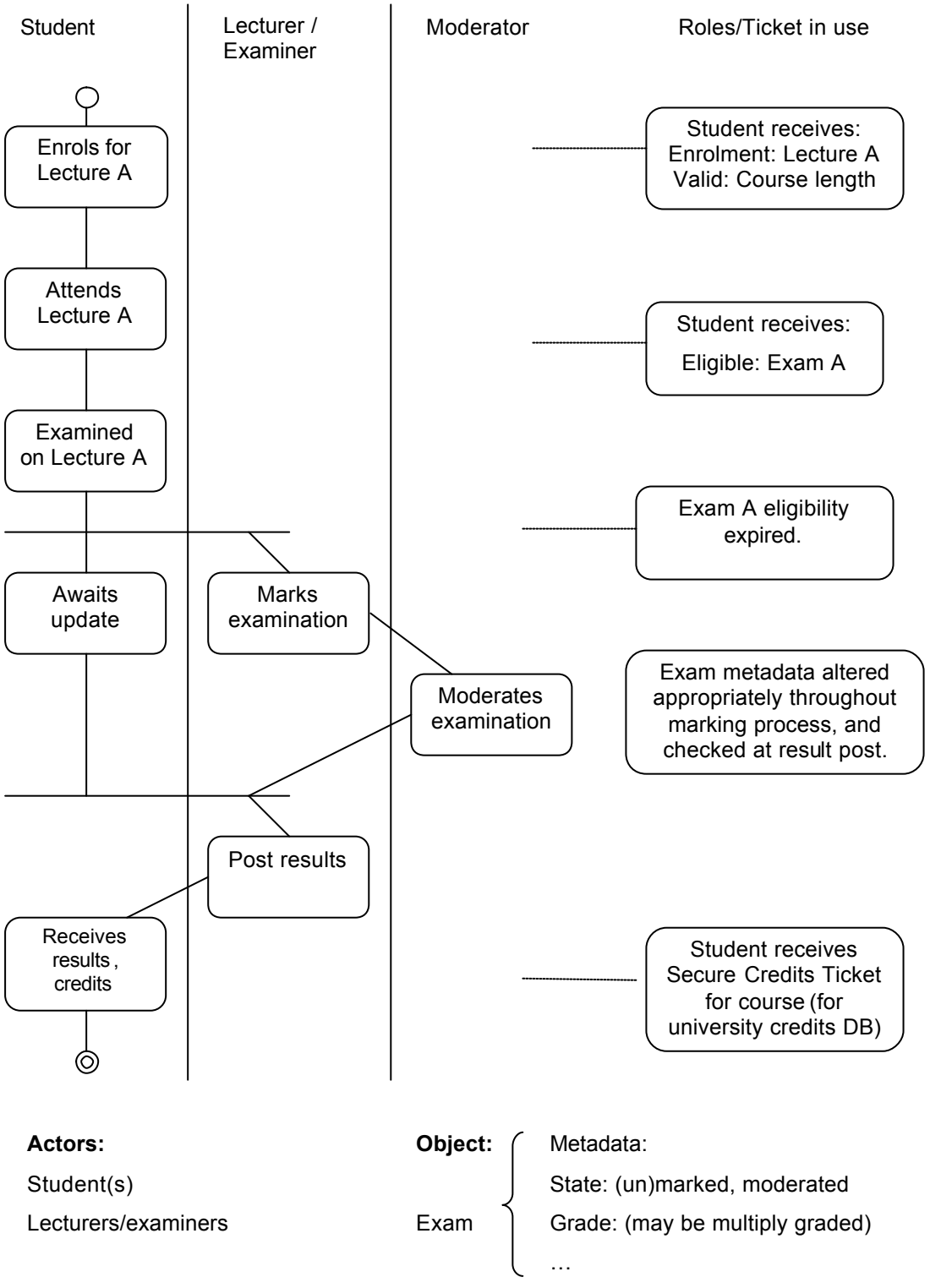

Figure 8 – Lecture/exam process

Notice that the stage concerning enrolment to the course is essentially a reprise of the simple example previously discussed in this section. However, the later stages concern multiple actors as well as asynchronous parallel actions and are as such more complex. The service described here is effectively a ticket-based system for managing a term's worth of student enrolment. It is an application-specific base service within

the NUKATH system and is as such reusable for any NUKATH deployment. Its elements are also reusable, for example the rating service, the service that controls result posting, and so forth, as of course are the object/metadata combination(s).

The utility of this system lies within the opportunity for federative strategies. Whilst it describes an acceptable solution within a simple network, the additional level of abstraction typically overlays an existing authentication method and might even be considered to be redundant in some situations. However, once multiple organizations are involved, this shared abstract layer becomes indispensable. Consider the following scenario; a student at Karlsruhe University may complete modules with any of a number of affiliated universities, such as Augsburg University. During their time at the affiliated university the moment-to-moment administrative details are naturally controlled by the partner university and on successful completion of the course, the student receives credits from the partner university. When the student returns to his or her home university, the credits represented in the student's ticket are presented in order to permit their education to continue. These may be copied into a regular student profile database at the home university.

6 Conclusion and further work

We have identified a number of currently important directions in which to expand our research. The first of these is the previously mentioned concept of federation. We are working towards a robust definition of federation in the context of cross-organizational Web applications, and exploring the limits of the security structures proposed within this paper.

Now that the WSLS/NUKATH system exists, a detailed way of modeling web sites is available based on the tree-view structure defined by the EvolutionBus as described within this paper. This also encompasses modern additions such as Web Service integration as currently defined. Now that this framework and support system exists in a form which provides an appropriate level of restriction to the design process, we can work on the new opportunities offered within the development process. This provides a basis for exploring further issues that may arise during modern web development, like issues of usability or IPR, whilst retaining a clear focus on reusability.

Examples

The WSLS-System and further information related to the WebComposition approach can be found at: http://www.wsls.net

Acknowledgments

The work on the NUKATH project is supported by a grant from the Bundesministerium für Bildung und Forschung (BMBF), Germany.

References

[1] P. Allen and S. Frost, *Component-Based Development for Enterprise Systems: Applying The Select Perspective*. Cambridge: Cambridge University Press, 1998.

[2] C. Szyperski, "Component-Oriented Programming: A Refined Variation on Object-Oriented Programming," *The Oberon Tribune*, vol. 1, 1996.

[3] R. Prieto-Díaz, "Reuse as a New Paradigm for Software Development," presented at Systematic Resue: Issues in Initiating and Improving a Reuse Program: Proccedings of the International Workshop on Systematic Reuse, Liverpool, 1996.

[4] H. Haas, "Web Services Activity Statement," World Wide Web Consortium (W3C), 2002.

[5] N. Mitro, "SOAP Version 1.2 Part 0: Primer - W3C Proposed Recommendation 07 May 2003," vol. 2003: World Wide Web Consortium (W3C), 2002.

[6] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen, "UDDI Version 3.0," UDDI.org, Published Specification uddi-v3.00-published-20020719, 19.07.2002 2002.

[7] G. Della-Libera, B. Dixon, J. Farrell, P. Garg, M. Hondo, C. Kaler, B. Lampson, K. Lawrence, A. Layman, P. Leach, J. Manferdelli, H. Maruyama, A. Nadalin, N. Nagaratnam, R. Rashid, J. Shewchuk, D. Simon, and A. Wesley, "Security in a Web Services World: A Proposed Architecture and Roadmap," IBM Corporation and Microsoft Corporation, Web Page 01.05.2002 2003.

[8] H.-W. Gellersen, R. Wicke, and M. Gaedke, "WebCompostion: an object-oriented support system for the Web engineering lifecycle," *Computer Networks and ISDN Systems*, vol. 29, pp. 1429-1437, 1997.

[9] M. Gaedke, C. Segor, and H.-W. Gellersen, "WCML: Paving the Way for Reuse in Object-Oriented Web Engineering," presented at 2000 ACM Symposium on Applied Computing (SAC 2000), Villa Olmo, Como, Italy, 2000.

[10] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen, "RFC 2518: HTTP Extensions for Distributed Authoring -- WEBDAV," IETF, Network Working Group, RFC 2518, February 1999 1999.

[11] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer, "Simple Object Access Protocol (SOAP) 1.1," vol. 2000: World Wide Web Consortium (W3C), 2000.

[12] M. Gaedke, H.-W. Gellersen, A. Schmidt, U. Stegemüller, and W. Kurr, "Object-oriented Web Engineering for Large-scale Web Service Management," presented at Thirty-Second Annual Hawaii International Conference On System Sciences (HICSS-32), Island of Maui, USA, 1999.

[13] M. Gaedke, *Komponententechnik für Entwicklung und Evolution von Anwendungen im World Wide Web*. Aachen: Shaker Verlag, 2000.

[14] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," IETF, Network Working Group, Cambridge, MA, USA, Request for Comments 1510, 09.1993 1993.

[15] J. S. Erickson, "Fair Use, DRM, And Trusted Computing," *Communications of the ACM (CACM)*, vol. 56, pp. 34-39, 2003.

[16] B. L. Fox and B. A. LaMacchia, "Encouraging Recognition Of Fair uses In DRM Systems," *Communications of the ACM (CACM)*, vol. 56, pp. 47-49, 2003.

[17] D. S. Rosenblum and A. L. Wolf, "A Design Framework for Internet-Scale Event Observation and Notification," in *Proceedings of the Sixth European Software Engineering Conference (ESEC/FSE 97)*, M. Jazayeri and H. Schauer, Eds. Germany: Springer-Verlag, 1997, pp. 344-360.

[18] D. Eastlake, J. M. Reagle, D. Solo, M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "XML-Signature Syntax and Processing - W3C Recommendation 12 February 2002," http://www.w3.org/, W3C Recommendation REC-xmldsig-core-20020212, 12.02.2002 2002.

[19] T. Imamura, B. Dillaway, and E. Simon, "XML Encryption Syntax and Processing - W3C Recommendation 10 December 2002," http://www.w3.org/, W3C Recommendation REC-xmlenc-core-20021210, 10.12.2003 2002.