Construction by Linking: The Linkbase Method

Johannes Meinecke
University of Karlsruhe
Engesserstr. 4
76128 Karlsruhe, Germany
+49 (721) 608-8072
meinecke@tm.uka.de

Frederic Majer
University of Karlsruhe
Engesserstr. 4
76128 Karlsruhe, Germany
+49 (721) 608-7393
majer@tm.uka.de

Martin Gaedke
Chemnitz University of Technology
Straße der Nationen 62
09111 Chemnitz, Germany
+49 (371) 531-25530
gaedke@cs.tu-chemnitz.de

ABSTRACT
The success of many innovative Web applications is not based on the content they produce – but on how they combine and link existing content. Older Web Engineering methods lack flexibility in a sense that they rely strongly on a-priori knowledge of existing content structures and do not take into account initially unknown content sources. We propose the adoption of principles that are also found in Component-based Software Engineering, to assemble highly extensible solutions from reusable artifacts. The main contribution of our work is a support system, consisting of a central service that manages n:m relationships between arbitrary Web resources, and of Web application components that realize navigation, presentation, and interaction for the linked content.

Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable Software – *reusable libraries.* H.5.4 [**Information Interfaces and Presentation**]: Hypertext / Hypermedia – *Architectures*

General Terms
Management, Design

Keywords
Web Engineering, Content Linking, Web Services, Triple Stores

1. INTRODUCTION
During recent years, the Web has experienced several paradigm shifts. Seen originally as a means for publishing documents worldwide, it has later also been used as a platform for applications. Now, once again, we observe another fundamental change, as sites no longer form isolated applications, but instead combine their functionality with external services and the content contributed by large communities of participating Web users. The achieved added value is especially high, when content is not just integrated as separate sets of resources, but when resources from different sources are linked to each other. Within this work, we speak of *links* as items referencing two resources that somehow belong together, as e.g. a photo and an XML document describing the person who took the photo. The systematic construction of such Web applications is challenged with the need for continuous extensions with new content sources. Unlike in situations where everything is under the control of the application provider, the set of sources that are potentially relevant changes frequently, as old services become obsolete and new services become popular. Consequently, the development process must account for originally unknown sources to be integrated and linked to the existing content later. Related to that, there is a repetitive implementation effort for content source linking that is independent of the application domain. Complexity results e.g. from issues of distribution, caching, support for multiple service interfaces, or the realization of navigation across the linked content. Hence, this raises the question of how we can abstract from specific applications and package generic functionality in reusable components. Additionally, we have to deal with content sources that are unprepared to be linked. Belonging to different organizations, they were most likely developed without knowledge from each other. Therefore, the linking structure has to be imposed retrospectively, without the means to make any changes to the external sources.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

2. THE LINKBASE METHOD
To address the mentioned challenges, the Linkbase method aims at building applications by linking autonomous content sources with the help of a support system. Figure 1 outlines the general architecture and the steps to be performed.

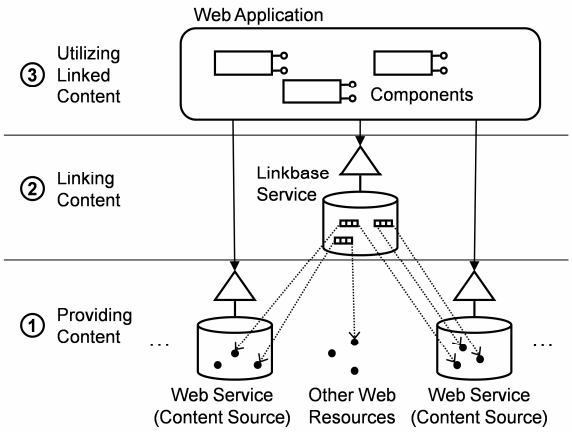

Figure 1: Architectural Overview of the Linkbase Method

Inspired by older Open Hypertext approaches as e.g. [3], links between arbitrary resources are managed separately, with the help of a central Web service (the Linkbase service) in a uniform way. The Web application itself is assembled from generic, domain-independent components that work with the links from the Linkbase and the content from the different sources to provide navigation, presentation, and interaction to the user. In the following, we give a brief overview of the three groups of activities that are necessary to apply the idea of the Linkbase for building applications.

2.1 Providing the Content Sources

Before the content sources can be linked to each other, they need to be accessible in a uniform way. In the case of conventional Web resources, the URI already enable such a uniform access, as e.g. addressing images via URLs over HTTP. If the content originates from Web services, there is a wider choice of access methods and interfaces. To overcome diversity, we propose the introduction of a generic, uniform interface that abstracts from the particular type of content. As one alternative, the CRUDS interface allows the querying and manipulation of arbitrary sets of content objects through the operations *Create*, *Read*, *Update*, *Delete* and *Search*. The interface can either be implemented at directly controlled services, or at wrapping services that delegate the content requests to third-party services. In order to address the resources uniformly, CRUDS uses a naming scheme based on Uniform Resource Nominators (URN) that contain, in addition to the local object identifiers of the resource, also an identifier of the service supplying the resource.

2.2 Linking the Content

Based on the unified information space, the second step of the Linkbase method deals with the actual linking of the resources. The Linkbase approach treats links as triples of URIs, each consisting of a subject, a predicate, and an object. In our case, the subject and object URIs refer to resources provided by the autonomous data sources, and the predicate URI serves as a label for the type of relationship. For storing and managing triples, there already exists a wide variety of triple stores [1]. In correspondence to the loosely coupled architecture of the overall solution to be built, the triple store is connected to the application as a Web service, called *Linkbase service*. The built-in reasoning support provided by many triple stores can be applied to relieve the application from the burden of computing links by itself, as e.g. in the case of transitive relationships. In addition to referencing external content, it may also be necessary to retrieve the information about which resource is linked to which from the outside. To account for this, we propose an extension of the triple store concept to provide triples extracted from external sources at runtime (i.e. the triples do not originate from information stored at the Linkbase, but from queries to external services).

2.3 Using Linked Content in the Application

The aim of the third step is to allow developers to construct the actual Web application without having to program it. Instead, they assemble the application from ready-built, reusable components. To achieve this, the Linkbase method builds on previous work, the WebComposition Service Linking System (WSLS) approach [2]. In WSLS, separation of concerns is realized by developing components as fine-grained implementation artifacts that can be combined with each other by following the Decorator software design pattern. For example, a component providing a list of content objects might be combined with a component for presenting the individual items and a component realizing an index navigation pattern. In this work, we supplemented the WSLS approach with a catalogue of components specialized on dealing with linked content. Since both content sources and links are accessible in a uniform way, we can restrict the number of components to be implemented by focusing on generic functionality. This includes particularly a component that retrieves and caches the content objects from the Web services and that supports the Web service interface chosen for unification (e.g. the CRUDS interface). As a Linkbase-specific navigation component, the **Fisheye** allows users to navigate through the graph formed by the Linkbase, along selected types of links. This is realized by decorating the presentation of a currently active and visible content object with smaller navigatable preview presentations of related objects around it (cf. Figure 2). An example for a component that is concerned with presentation aspects is the **Timeline**, which visualizes objects related to a given context in time. The **Content Connector** supports the interaction between the user and the linked content by allowing the user to insert new links with a single mouse click. The complete component catalogue covers the general functionality of the components, the way they can be configured to suit different applications, and concrete examples.

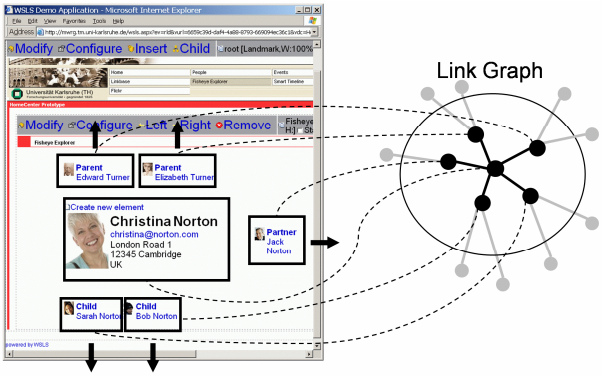

Figure 2: Example of a Fisheye Component

3. THE LINKBASE APPLIED

In order to gain practical experience, we implemented a Linkbase support system, including an extended triple store service as well as several tools to speed up development. This system has been used within the context of the project "Software Engineering for Information Appliances at Home", whose outcome included a Web portal targeted at families at home. The content sources included a number of new services (e.g. for personal profiles of family members) and wrappers for existing non-Web systems (e.g. a calendar service that provides access to a Microsoft Exchange server). Moreover, we developed a part of the components from our catalogue. For example, the Fisheye component supports the user in navigating through a family tree, as well as realizes a presentation slide browser, where users can skim through the sections and subsections of a structured lecture with up to 800 slides. The experiments demonstrated the reusability of the identified components for a wide range of purposes.

4. REFERENCES

[1] Beckett, D., Scalability and Storage: Survey of Free Software/Open Source RDF storage systems - 2002), W3C: http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report/ (12.10.2006).

[2] 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. Matera, Editor. 2005, Rinton Press. p. 26-37.

[3] Pearl, A. Sun's Link Service: A Protocol for Open Linking. in 2nd Annual ACM Conference on Hypertext. 1989. Pittsburgh, USA: ACM Press. p. 137-146.