

A Repository to facilitate Reuse in Component-Based Web Engineering

Martin Gaedke, Joern Rehse, Guntram Graef

Telecooperation Office (TecO), University of Karlsruhe, Germany

{gaedke, rehse, graef}@teco.edu

Abstract: The World Wide Web has become a platform for distributed applications of all kinds. Its originally anticipated scope has changed dramatically but without major changes to the primary mechanisms. Web application development suffers increasingly from the coarse-grained implementation model of the Web, as established software design concepts are hardly applicable to it. The object-oriented WebComposition Markup Language (WCML) addresses these problems with a fine-grained component-based model and thus supports the application of software engineering practice to applications in the Web. In this paper, we introduce the WebComposition Repository as a key tool for a systematic approach to code reuse. The repository is used for storing, managing, and retrieving large numbers of WCML components. Consequently, it facilitates reuse in component-based Web Engineering.

Keywords: Repository, Reuse, CBWE, Pattern, WCML

1. Introduction

The applications put to use in the Web now have moved a long way from the basic hypermedia systems of the early nineties. The WWW has gone far beyond presenting information. It has become a standard platform for distributed applications. Although the applications become increasingly complex, the development process remains ad-hoc in real-life Web development. A large gap between design models and the implementation model of the Web has been recognized to be one of the main reasons for the low acceptance of disciplined development in the community of Web developers. The answer to the Web's particular blend of software-crisis is widely considered to be Web Engineering: applying software engineering practice to the Web [CGV98, GWG97, Kris98]. Software engineering methods have been brought to the Web through design models especially suited for Web and other hypermedia applications, for example OOHDM[SRB96], RMM[ISB98], and JESSICA[BaS98]. Nevertheless, the lack of representation of higher level concepts in the implementation remains. Another essential aspect of engineering sciences is reusing previously developed artifacts. Once more the WWW suffers from its heritage. The intentionally simple and coarse-grained implementation model fit well the original purpose of document publication but makes reuse of complex code fragments extremely difficult. In the last years, a very important topic in research on code reuse has been component-based software engineering. It is

seen as an enabling technology for successfully developing more complex software by more code reuse, leading to an increase in development productivity [Ber97], [FrN87]. Assembling systems from a choice of software components is an obvious example of code reuse.

We have introduced the object-oriented WebComposition model [GWG97] and the WebComposition Markup Language (WCML) [GBG98] to allow a complete mapping of design concepts to implementation. WCML facilitates Web application development by means of composing Web applications from components. Examining the requirements for successful code reuse it became clear that adequate libraries or repositories as well as tools for locating and retrieving code from them were required [MBK91], [FrN87]. To fully exploit the advantages of code reuse when using the WebComposition system a repository for WCML components is being built.

In this paper we will give an overview of the WebComposition approach and introduce our research and development concerning the repository for WCML components. Finally, conclusions and a perspective of our future work are presented.

2. Components for the Web

Common Reuse Practices and the Web

Component-Based Software Engineering (CBSE) refers to the construction of (large) software systems from existing parts [HCF97]. This approach to code reuse reduces the production costs and enhances the maintainability of the software system. Consequently, we will use the term Component-Based Web Engineering to describe the construction of (large) Web Sites from existing parts.

Another major new technique of reusing code and previously invented solutions is the utilization of design-patterns [GHJ94]. Design-Patterns have found their way to the Web in the form of Hypermedia Design-Patterns [RGC96]. The integration of knowledge about design-patterns into a component-based system for Web Engineering seems to be desirable.

Granularity of Entities in Design and the Web Implementation model

The Web has a resource-based (file) implementation model. During the design of Web applications the entities handled by the designers are often defined at a much higher resolution than possible in the actual code produced during the development process. Objects seen as different system parts by the designers will have to be integrated into a single resource while a single design entity may very well appear in different documents of the implementation [GGS99]. Mapping changes in the design to changes in the implementation becomes a tedious and error-prone task. Increasing complexity of requirements has made the coarse-grained implementation model of the Web a burden to developing and maintaining parties. Abstract design decisions based on higher level concepts are difficult to represent and maintain in the semantically deprived Web implementations as they get lost on their way to code. Consequently, a reverse mapping

from Web implementations to higher-level (fine-grained) concepts is impossible. This effect is undesirable especially as the artifacts cannot be used for evolution and application management during the rest of the life cycle due to the Web's legacy.

Component-Based Web Engineering: The WebComposition Approach

The WebComposition approach is based on modeling with components. A component in the WebComposition model is a code abstraction of any target language, e.g. HTML, script code, VRML, or Latex. The granularity of WebComposition components is solely dependent on the nature of the design artifact that is captured in the component. A component may represent e.g. an atomic feature like the font size attribute, a complex navigation structure, implementations of hypermedia design-patterns, or simply compositions of other components. In this way complete target language resources can be constructed by compiling WCML components.

The WebComposition model is object-oriented, which allows code-reuse when creating new components as a component may be constructed using existing components. This is achieved using fundamental principles of object-orientation: specialization (inheritance) and aggregation (has-part relation). Inheritance works according to a prototype-instance model as in SELF [UnS87] and not according to the class-based inheritance models well known from languages such as C++ and Java. Prototyping is a technique of code sharing. Any component can be used as a prototype or referenced to facilitate code-reuse. The WebComposition Markup Language (WCML) is an application of the eXtensible Markup Language (XML) [W3C98] and allows a (tag-based) definition of components, properties and relationships between components on top of the prototype-instance model mentioned above. Being a child of XML the WCML comes with all the good traits we expect from its family: WCML is platform independent, easy to parse, and it is rigorous in terms of well-formed or valid documents. Components reside in a persistent storage called Component Store cf. [GWG97], which may be represented by an XML-file, a Web-Server, or a database.

3. The WebComposition Repository

As soon as a lot of components are available finding appropriate components becomes one of the main problems of CBSE [FrN87]. To make WebComposition a useful environment for CBWE an instrument for finding components in the Component Store is required. Given a large number of possibly undocumented WCML components the non-hierarchical structure of the Component Store is insufficient for managing and locating components. To address these problems a Repository for WCML components is being constructed. The Repository works on top of the WebComposition Component Store. The Component Store provides persistent storage to components. The Repository's basic mode of operation is the cooperation of three system entities: the Component Store, a Metadata Store, and a search or browsing tool. These tools find appropriate Metadata Stores using the Repository Broker. The architecture of the Repository is given in the following sections:

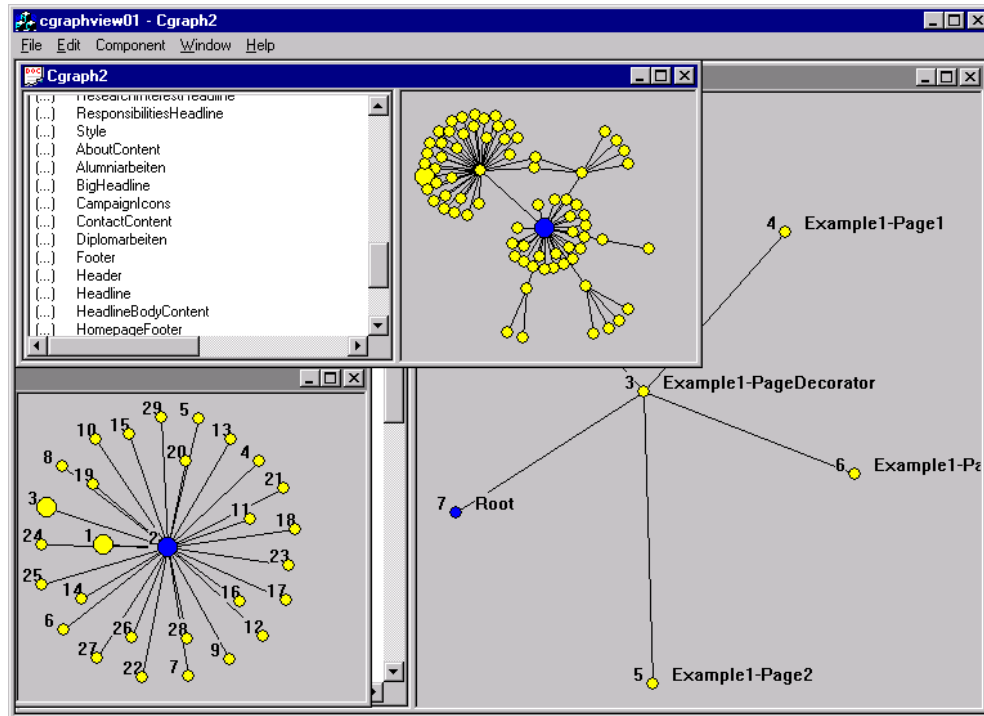


Figure 1 Screenshot of a Browsing Tool

The Repository Broker

The Repository Broker is responsible for the coordination of currently participating and future system entities, thus ensuring an open model. It is used for selecting Metadata Stores by interfaces. A new Metadata Store registers with the broker. The broker maintains a database of interfaces of all available Metadata Stores. A tool can request a list of Metadata Stores supporting a required interface from the broker. The broker can then provide means for the tool to access the interface of a selected Metadata Store directly.

Metadata Store

The added value of the Repository approach to storing WCML components lies in the Metadata Stores that allow access to metadata on the components in the Component Store. The Component Store is seen as a "flat" storage containing no more information than the components. The Metadata Store can be used to store additional indexing-, browsing-, hierarchy-, or any other Meta-information. A typical Metadata Store defines a graph having components as vertices and edges that represent some relation between the components.

Query Tools

The third part of the Repository are the viewers and query tools for retrieving components, analyzing components, or offering any other service using the Repository. Sorting and displaying sets of components in a way defined by structures with varying semantics is possible using Metadata Stores. Additionally it is possible for the Metadata

Store to manipulate metadata as a reaction to query tool user actions. Techniques like adaptive indexing could be used to rearrange components in order to group them according to previously made requests. In such a way the repository could organize its own structure to fit the users needs [Hen97].

Figure 1 shows a Browsing Tool that allows browsing through graphs defined on a set of components. The structure of a graph is metadata. Semantics of the graph may differ depending on the Metadata Store that is in use. In the example edges indicate a prototype-instance relation.

4. Conclusions and Further Work

In this document, we have pointed out how the gap between implementation and design model hinders the use of modern software engineering in the development of Web applications. The WebComposition approach with its implementation technology WCML bridges this gap and allows designing for reuse. In component-based software engineering the problem of storing, retrieving, and managing components becomes a center-point of considerations. We have introduced the WebComposition Repository for storing WCML components to address this problem.

The WebComposition Repository is still under development. We are currently focusing on more sophisticated search tools, adaptive Metadata Stores and tools for processing retrieved WCML Components, to facilitate Web Engineering. Two criteria for the adequacy of a repository were used in [MBK91]. At first a sufficient number of components must be provided, secondly the appropriate code should be easy to locate and retrieve. The existing WebComposition Repository addresses the second issue. The first requirement may be met by automated component extraction.

Research is underway at TecO to develop tools for acquiring massive amounts of “real life” HTML code as base material for further processing as well as the semi-automatic extraction of components. Large amounts of easily accessible HTML code can be found on the Web. Evidently, the WWW is the ideal source for gathering HTML code structures, i.e. downloading bulks of HTML pages from the WWW. Giving a detailed description of these projects is far beyond the scope of this document. Semi-automatic tools for finding recurring patterns in high-level languages exist such as PEEL for Lisp [Hen97]. As HTML is not a high-level programming language and possible structures are far less complicated than in Lisp (for example) it is believed that automating component extraction and finding hints for archiving hypermedia design-patterns will be easier by magnitudes.

Availability

The WCML-compiler is available at: <http://www.teco.edu/~gaedke/webe/>

References

- [BaS98] R.A. Barta, M.W. Schranz (1998). *JESSICA: an object-oriented hypermedia publishing processor*. In Computer Networks and ISDN Systems 30(1998), Special Issue on the 7th Intl. World-Wide Web Conference, Brisbane, Australia, April 1998, 239-249.

- [Ber97] K. Berg (1997). *Component-Based Software Development: No Silver Bullet*, Object Magazine, 3-97.
- [CGV98] F. Coda, C. Ghezzi, G. Vigna, F. Garzotto (1998). *Towards a Software Engineering Approach to Web Site Development*. In Proceedings of 9th International Workshop on Software Specification and Design (IWSSD), Ise-shima, Japan.
- [FrN87] W.B. Frakes, B.A. Nejme (1987). *Software Reuse through Information Retrieval*, In: Proceedings of the 20th Annual Hawaii International Conference On System Sciences, 1987.
- [GHJ94] E. Gamma, R. Helm, R. Johnson, J. Vlissides (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading.
- [GBG98] M. Gaedke, M. Beigl, H. W. Gellersen (1998): *Mobile Information Access: Catering for Heterogeneous Browser Platforms*. In: Proceeding of the International Workshop on Mobile Data Access in conjunction with 17th International Conference on Conceptual Modeling (ER98), Singapore, p. 201-212.
- [GGS99] M. Gaedke, H.-W. Gellersen, A. Schmidt, U. Stegemüller, W. Kurr (1999). *Object-oriented Web Engineering for Large-scale Web Service Management*. In: R. H. Sprague (Ed.) Proceedings of the 32nd Annual Hawaii International Conference On System Sciences, Maui, Hawaii, (CD-ROM).
- [GWG97] H.-W. Gellersen, R. Wicke, M. Gaedke (1997). *WebComposition: an object-oriented support system for the Web engineering lifecycle*. In: Computer Networks and ISDN Systems 29, Special Issue on the 6th Intl. World-Wide Web Conference, Santa Clara, CA, USA, p. 1429-1437.
- [HCF97] Capt. G. Haines, D. Carney, John Foreman (1997). *Component-Based Software Engineering / COTS Integration*. Software Technology Review, Software Engineering Institute, Carnegie Mellon University. <http://www.sei.cmu.edu/str/descriptions/cbsd.html> 01/02/1999.
- [Hen97] S. Henninger (1997). *An Evolutionary Approach to Constructing Effective Software Reuse Repositories*, ACM Transactions on Software Engineering Methodology, 1997.
- [ISB98] T. Isakowitz, E.A. Stohr, P. Balasubramanian (1998). *RMM: A Methodology for Structured Hypermedia Design*, Communications of the ACM, Vol. 38, No. 8, August 1995, pp. 34-44.
- [Kris98] A. Kristensen (1998). *Template resolution in XML/HTML*. In Computer Networks and ISDN Systems 30(1998), Special Issue on the 7th Intl. World-Wide Web Conference, Brisbane, Australia, April 1998, 239-249.
- [RGC96] G. Rossi, A. Garrido, S. Carvalho (1996). *Design Patterns for Object-Oriented Hypermedia Applications*. In: Pattern Languages of Programs 2, Vlissides, Coplien and Kerth (eds.), Addison-Wesley.
- [SRB96] D. Schwabe, G. Rossi, S. Barbosa (1996). *Systematic Hypermedia Design with OOHDM*. In Proceedings of the ACM International Conference on Hypertext, Hypertext '96, Washington, March 1996.
- [UnS87] D. Ungar, R. B. Smith (1987). *Self: The power of Simplicity*, In: OOPSLA'87 Proceedings, p. 227-242.
- [W3C98] World-Wide Web Consortium (1998). XML: eXtensible Markup Language. <http://www.w3c.org/XML/>
- [MBK91] Y.S. Maarek, D.M. Berry, G.E. Kaiser (1991), *An Information Retrieval Approach for Automatically Constructing Software Libraries*, IEEE Transactions of Software Engineering, Vol. 17, No. 8, pp. 800.

About the Authors

All authors share the following address:

Telecooperation Office (TecO), University of Karlsruhe

Vincenz-Prießnitz-Str. 1, 76131 Karlsruhe, GERMANY

<http://www.teco.edu/>

Ph.: +49 (721) 6902-79, Fax: +49 (721) 6902-16

Email: { gaedke, rehse, graef }@tec.uni-karlsruhe.de

Martin Gaedke is research assistant at the Telecooperation Office (TecO) of the University of Karlsruhe and technical lead in collaborative Web engineering projects. His research interests is in application of software engineering practice to applications in the WWW, and specifically in component-based software engineering, reuse, frameworks, reflection, software architectures, patterns and software evolution for Web applications. He obtained a Master degree in computer science from the University of Karlsruhe in 1997.

Joern Rehse is currently preparing a thesis on Web engineering towards a research degree in Computer Science at University of Karlsruhe. Formerly he was a software-developer in several international projects. He has been working at TecO on CBSE in Web technology and database technology. His research interest is in CBSE and software-tools for assisting CBSE as well as applications of interactive computer graphics.

Guntram Graef obtained a Master's degree in computer science from the University of Karlsruhe in 1998. In 1999 he joined TecO as a research assistant and works in collaborative Web engineering projects. Former to that he was an independent consultant designing distributed applications for major companies in Europe and Asia. His current research interest is in methodologies and tools for large scale Web development, user profiling and applications of XML.