

# Patterns and Components: Capturing the Lasting amidst the Changing

*Hans-W. Gellersen<sup>1</sup>, Fernando Lyardet<sup>2</sup>, Martin Gaedke<sup>1</sup>, Daniel Schwabe<sup>3</sup> and Gustavo Rossi<sup>2</sup>*

<sup>1</sup>TecO, University of Karlsruhe, Germany. Email: {hwg|gaedke}@teco.edu

<sup>2</sup>LIFIA, Depto de Informática, UNLP, Argentina. Email: {fer|gustavo}@sol.info.unlp.edu.ar

<sup>3</sup>Depto de Informática, PUC-RIO, Brazil. Email: schwabe@inf.puc-rio.br

**Keywords:** Web applications, Design Patterns, Object-oriented Modeling, OOHDM, XML, WCML

## 1. Introduction

Documents, sites and in general applications in the World-Wide Web are undergoing constant evolution. Yet we would argue that not everything changes but that a large proportion of design decisions are of lasting quality. We propose that such design decision be captured explicitly through complimentary use of design patterns in design methods, and illustrate this with the use of patterns in the Object-Oriented Hypermedia Design Method (OOHDM). Further, we propose the use of an object-oriented web implementation technology, the WebComposition Markup Language (WCML), to facilitate straightforward mapping of object-oriented design patterns to an implementation, and so to facilitate the tracking of higher-level design decisions in a web implementation.

## 2. Patterns and Components

The popularity of the Web and its advantages as client-server platform have lead to an avalanche of dynamic applications being delivered over the web. Building such applications is increasingly complex, facing an ever-growing set of requirements and technologies that must be present in a system. We share the view of Coda et al. [1] that this complexity can be greatly reduced by a better integration of software engineering knowledge at both design and implementation stages.

As has been stated elsewhere [5][7][9], the most outstanding design decisions must be made explicit using some design notation. The results of this process can be greatly enhanced if it were possible to apply existing knowledge of good design at the different stages of design: information modeling, navigational design over the information, and the interface. In this context, design patterns [2] introduced a powerful instrument to complement methods in that they address problems at a higher level of abstraction. Many design decisions that cannot be recorded by the primitives of a method can be described using patterns.

H-W. Gellersen, F. Lyardet, M. Gaedke, D. Schwabe, G. Rossi: Patterns and Components: Capturing the Lasting amidst the Changing. Active Web Conference, British Computer Society Special Interest Group in Human Computer Interaction, U.K., January 20, 1999.

Design patterns are a good means for recording design experience as they systematically name, explain and evaluate important and recurrent designs in software systems. They describe problems that occur repeatedly, and describe the core of the solution to that problem, in such a way that we can use this solution many times in different contexts and applications. Looking at known uses of particular design patterns, we can see how a successful designer solves recurrent problems. Patterns are not just good solutions that we invent to solve particular problems; they are solutions that appear repeatedly in different applications; patterns are not invented, they are discovered.

The implementation stage presents similar problems but with different (and changing) forces to those we encounter during the design phase, which in turn generates the well-known gap between the design and implementation. Available web implementation technologies support ever more specific presentation effects but generally lack in support of abstraction, rendering it particularly difficult to bridge this gap. To address this issue, we propose a web implementation technology enriched with object-oriented concepts, to facilitate a more straightforward mapping of design concepts to an implementation in the web.

## **2.1 Extending Design Methods with Patterns**

As we previously explained, design patterns are useful to record recurrent design themes in software systems. The process of discovering new design patterns is triggered by the study of successful applications and how they solved recurrent problems in their designs. This helps us to increase our understanding on the design process and therefore to create relationships among similar applications, establishing the context in which these solutions are applied, the different forces involved, extract the solution proposed to balance the different design constraints, and search how the solution that has been found is related with other existing patterns.

Being knowledgeable of a catalogue of patterns, a designer can work at a higher level of abstraction, applying existing experience rather than solving every problem from scratch. There are two ways of incorporating these patterns into the process of designing and implementing hypermedia applications. The first is to use patterns as active guidelines in the design enterprise. The other is by enriching existing methods with higher level primitives that allow expressing some patterns with a single notation.

At PUC-RIO and LIFIA, we have developed the Object-Oriented Hypermedia Design Method (OOHDM)[8], have used it to design, and implement different Web applications [9]; because of our design experience, we have incorporated some patterns into our method. OOHDM separates conceptual from navigational and interface design, thus allowing to decouple the different design concerns into proper activities.

During the conceptual Design, a model of the application domain is built using well known object-oriented modeling principles such as classes and associations, with a notation similar to UML. In this way by treating nodes and links as object-oriented views of conceptual objects we provide an interesting architectural style for Web applications that separates navigation from other kind of computations (for example those existing in legacy applications). The underlying

H-W. Gellersen, F. Lyardet, M. Gaedke, D. Schwabe, G. Rossi: Patterns and Components: Capturing the Lasting amidst the Changing. Active Web Conference, British Computer Society Special Interest Group in Human Computer Interaction, U.K., January 20, 1999.

pattern here is a combination of the Observer and the Decorator in [2].

Since this strategy is quite usual when porting existing applications (or databases) to the Web environment, OOHDM has been enriched with Navigational Contexts as a design primitive; in fact the overall structure of a web (or more general hypermedia) application is specified as a set of navigational contexts [9]. The design notation in OOHDM provides a rich set of primitives for showing the way in which navigational objects (nodes and links) are related with conceptual ones (database objects, for example).

Other examples of design patterns that have enriched OOHDM are the InformationOnDemand pattern that addresses the problem of the reduced amount of displayable information and therefore avoid subordinating well established hypermedia concept of the information node as a unit to display limitations[6]; and the Landmark pattern, which addresses the problem of modeling subsystems that should be accessible from the whole website. Landmarks provide such access and should be presented in a uniform way in different subsystems of a web to provide a consistent cue for the navigating user.

## **2.2 Object-oriented Web Implementation Technology**

Object-oriented designs, and in particular design patterns described at some level of abstraction, can not be captured in the standard web implementation model which is based on decomposition of web applications into specific resources of relatively coarse granularity. The lack of abstraction and coarse granularity of this model hampers construction of generalized components and reusable implementation frameworks. To address this problem, we have develop WebComposition, an object-oriented model for web applications [4], and the WebComposition Markup Language (WCML) as XML-based implementation technology [3].

WebComposition defines an object-oriented model based on decomposition of web applications into components modelling web entities at arbitrary granularity and level of abstraction. In contrast to other proposed object-oriented web models such as WOOM [1], WebComposition is not a generative model requiring objects/components to be composed or derived from a given set of primitives, but of course any set of primitives can be modeled as application building blocks. Components can reference other components to model aggregation (has-part) or specialization (inherits-from). By means of a special reference type, components can refer to so-called prototype components from which they inherit state and behavior. The inheritance model is based on the prototype-instance paradigm so that any component can function as such a prototype. While this model does not distinguish between instances and classes, a class-oriented view can easily be emulated so that the model is conceptually well aligned with any object-oriented design model. With adoption of general object-oriented concepts the WebComposition model takes in the well-known properties modularity, abstraction, encapsulation, and extensibility, while also providing for general applicability.

### **WebComposition Markup Language (WCML)**

The WebComposition Markup Language WCML is based on the eXtensible Markup Language

H-W. Gellersen, F. Lyardet, M. Gaedke, D. Schwabe, G. Rossi: Patterns and Components: Capturing the Lasting amidst the Changing. Active Web Conference, British Computer Society Special Interest Group in Human Computer Interaction, U.K., January 20, 1999.

XML, which facilitates definition of a tag-based textual format for semantic mark-up of documents or data [11]. The XML document type definition of WCML describes a markup notation for WebComposition concepts, for description of components, their properties and their relationships. The following code describes the structure of a WCML document.

```
<wcml>

<component id='Page1Content'>
  <property name='Style' value='H1' />
  <property name='Information' value='Hello World....Content of page 1' />
  <property name='content'>
    <<rp name='Style' />>
    <rp name='Information' />
    </<rp name='Style' />>
  </property>
</component>
<component id='Page1'>
  <prototype is='Page1Content' />
  <property name='wcml.filename' value='Page1.html' />
  <property name='display.pagecontent.component' value='Page1Content' />
</component>
...further component declarations...
</wcml>
```

A WCML document contains one or more components. Each component has an identifier for referencing. The structuring of components into documents is independent of whether the components relate to the same document in the target web implementation. WCML documents can be used to organize components into modules, for instance to capture a set of domain-specific building blocks. While HTML documents are a unit for application delivery at runtime, WCML documents are a unit of application development that developers can effectively use for separation of concerns and enforcement of modularity.

In the above example we present two components for separating content and entities of the coarse-grained web implementation model. A component is defined by a set of properties, which are simple name-value pairs. For instance, the component *Page1Content* is modeling parts of the content of a HTML file with properties defining the information and the layout to provide. According to the WebComposition model every component has to specify its own web implementation which in WCML is done with a special kind of property 'content'. In the above code for *Page1Content*, the property 'content' describes the mapping of *Page1Content* to a representation in HTML. The second component, *Page1* described in the example is derived from *Page1Content* which as specified with the prototype-tag. Components inherit properties of their prototypes but can override them. In this case, *Page1* defines specific values for creating the HTML-file and inherits the content property from *Page1Content*. The example shows the use of abstraction. Specific content can be defined while simply inheriting the content property.

Web code generation from WCML is based on the content property and takes advantage of the availability of XML parsers for all major development platforms. XML based on SGML technology is rigorous in terms of well-formed and valid documents and therefore it is straightforward to parse XML-documents in general, and WCML in particular.

## Implementation Patterns in WCML

WCML supports abstraction and inheritance and thus facilitates the definition of reusable implementation patterns. For illustration, we have implemented the above mentioned Landmark pattern in WCML [10]. Landmarks provide easy access to different though unrelated subsystems of a web application, and are added to the content of different pages based on the decorator pattern as described in [2].

```
<wcml>
...
<component id='PageDecorator'>
  <property name='content'>
    <rp name='content' from='Landmarks' />
    <rp name='content' from='*display.pagecontent.component'>
  </property>
</component>

<component id='Page1Content'>
  <property name='Style' value='H1' />
  <property name='Information' value='Hello World....Content of page 1' />
  <property name='content'>
    <<rp name='Style' />>
    <rp name='Information' />
    </<rp name='Style' />>
  </property>
</component>

<component id='Page1'>
  <prototype is='PageDecorator' />
  <property name='wcml.filename' value='Page1.html' />
  <property name='display.pagecontent.component' value='Page1Content' />
</component>
...
</wcml>
```

In this example the Page1 component now inherits the behaviour of the *PageDecorator* component. The PageDecorator adds behavior by first providing the content of a *Landmark*-component and then the main Page-Content referenced by the *display.pagecontent.component*-property. Inheriting from the PageDecorator-component enriches pages with a uniform way to navigate between loosely related parts of a web application.

## 3. Conclusion

In this extended abstract we have motivated the use of patterns and components to model design decisions at higher levels of abstraction, both in the design and the implementation phase of web applications. For illustration, we have discussed the extension of the OOHDMM method with patterns, and the mapping of such patterns to an implementation in the web. To better facilitate the mapping from design to implementation we have proposed an object-oriented markup language based on the WebComposition model and the eXtensible markup language XML.

H-W. Gellersen, F. Lyardet, M. Gaedke, D. Schwabe, G. Rossi: Patterns and Components: Capturing the Lasting amidst the Changing. Active Web Conference, British Computer Society Special Interest Group in Human Computer Interaction, U.K., January 20, 1999.

## References

[1] F. Coda, C. Ghezzi, G. Vigna and F. Garzotto. *Towards a Software Engineering Approach to Web Site Development*. In Proceedings of 9th International Workshop on Software Specification and Design (IWSSD). April 16-18 1998, Ise-shima, Japan

[2] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[3] M. Gaedke, M. Beigl and H.-W. Gellersen. *Web Content Delivery to Heterogeneous Mobile Platforms*. Workshop on Mobile Data Access, Singapore, Nov. 1998. [4] H.-W. Gellersen, R. Wicke and M. Gaedke. *WebComposition: an object-oriented support system for the Web engineering lifecycle*, Computer Networks and ISDN Systems 29 (1997), p. 1429-1437. <http://www.teco.edu/~hwg/www6/PAPER232.html>

[5] T. Isakowitz, E.A. Stohr and P. Balasubramanian. *RMM: A Methodology for Structured Hypermedia Design*, Communications of the ACM, August 1995.

[6] F. Lyardet, G. Rossi and D. Schwabe. *Using Design Patterns in Educational Multimedia applications*. In Proceedings of ED-Media'98, World Conference on Educational Multimedia and Hypermedia, Freiburg, Germany, June 1998.

[7] G. Rossi, A. Garrido and S. Carvalho. *Design Patterns for Object-Oriented Hypermedia Applications*. In: Pattern Languages of Programs 2, Vlissides, Coplien and Kerth (eds.), Addison-Wesley, 1996.

[8] D. Schwabe, G. Rossi and S. Barbosa. *Systematic Hypermedia Design with OOHD*. In Proceedings of the ACM International Conference on Hypertext, Hypertext '96, Washington, March 1996.

[9] D. Schwabe and G. Rossi. *An Object oriented Approach to Web-Based Application Design*. In: TAPOS (Theory and Practice of Object Systems), Wiley and Sons, 1998.

## URLs

[10] Martin Gaedke. *Landmark Pattern in WCML*. <http://www.teco.edu/~gaedke/webe/activeweb/>

[11] World-Wide Web Consortium. *XML: eXtensible Markup Language*. <http://www.w3c.org/XML/>