COMPACT Web Design Approach:
A Methodology and Modelling Technique for communicating the High-level Design of Struts Web Applications.

Neil Axtell1, Liz Bacon2, Gill Windall2
1 Landmark Information Group, The Smith Centre, Fairmile, Henley-on-Thames, RG9 6AB, neil.axtell@landmarkinfo.co.uk
2 Dept of Computing Science, CMS, University of Greenwich, Park Row, London SE10 9LS, e.bacon@gre.ac.uk, g.f.windall@gre.ac.uk

Abstract

This paper describes the COMPACT web design approach. COMPACT (Composite Object Modelling Process for Architectural Communication in Teams) is used to enhance team communications and the quality of the design during the development of Struts applications. Almost every website is made up of a number of web pages and their associated server-side processing. As a result, most web development teams include people with specific technical strengths. In a web context, these are generally split between front-end skills (HTML, JavaScript, JSP, tag libraries) and middle to back-end skills (e.g. Server-side Java, Database Stored Procedures). These skills broadly reflect the division of work required in a web application. One challenge of developing a website is that design activities have to take place at more than one level. Developers with very different skills need a common model to communicate effectively and the Project Manager needs some way of identifying progress. This paper reviews a number of existing approaches to the modelling of web applications in the light of these requirements. It then explains the COMPACT approach itself and how it has been applied on a number of projects in industry. The strengths of the approach are that the one model described is flexible, easy to understand, uses colour to increase comprehension and encapsulates navigational and system functionality.

1. Introduction

The web has been with us now for over 15 years. During this time a variety of modelling techniques and methodologies to specify and design websites have also evolved. In the early years many of these techniques focused on one aspect of the system (e.g. user navigation), more recently approaches have been developed that encompass multiple aspects of the system (e.g. navigation, user interface design, classes) and support a greater proportion of the development lifecycle. Many of these more recent approaches are adaptations of UML.

This paper presents the COMPACT web design approach (Composite Object...
Modelling Process for Architectural Communication in Teams). It is a technique developed by Landmark Information Group for modelling the design of web applications. The company supplies land and property search information. Over the last three years the company has developed this technique and used it during the design and implementation of a number of web applications developed using the Struts framework [1]. This framework implements the Model-View-Controller (MVC) Design Pattern for Java web applications running within a web application server. The Views are implemented as Java Server Pages (JSP), the Controller is a Java Servlet and the Model is implemented as special Java Classes known as “Actions”. A single XML file controls the configuration.

Before developing their own modelling technique Landmark looked at existing approaches and concluded that none sufficiently met their needs. As has been commented on many occasions (e.g. [2]) there is often a gulf between the methods that academics develop and the methods that practitioners require to build large scale commercial applications. Landmark’s primary requirement is to have a single model that can be used as a tool to enhance the quality of the communication between all members of the development team. They found that existing techniques either fragmented essential information across several models or were just too complex to be accepted by the range of staff working on a project.

The remainder of this paper is structured as follows: Section 2 describes the need for this model and describes the problem in more detail. Section 3 reviews existing approaches and summarises the problems with them. Section 4 describes the COMPACT solution and section 5 summarises the work and discusses future developments.

2. The problem

Web development teams are typically made up of individuals with differing skill sets. These are split between front-end skills (HTML, JavaScript, JSP, tag libraries) and middle to back-end skills (e.g. Server-side Java, Database Stored Procedures). There is need for a simple yet comprehensive model that all developers can understand.

Whilst requirements analysis usually results in a set of front-end mock-ups to show the customer, a significant amount of detail still needs to be defined. The mock-ups show the basic flow through the website from the end user’s perspective, but they do not provide the developers with sufficient information about the server-side work required. Before developing detailed front-end and server-side designs an overview that everyone in the team can understand is needed. This overview needs to identify: the navigation of the web pages within the site, the server-side processing, the points within the application that connect to external systems, development responsibilities and any areas within the website that will be particularly complex.

Once the overall design has been agreed, the various developers can concentrate on their own areas of expertise. Having an overview also allows the developers to
negotiate internal “contracts”/interfaces between the front-end web pages and the server-side processing required. Without such an overview it is difficult to identify areas that may be particularly difficult and the project has to be dealt with on a page-by-page basis. A page-by-page approach does not allow individual developers to work at their own rate and often results in much reworking and a poorer quality system as the “contract” between the front-end and the server-side components is not identified and defined in the early stages of the project. Another problem is that it only allows developers to resolve issues as they come across them rather than dealing with them earlier, possibly by changing the overall design.

The situation and needs described above apply to web development in Landmark, however the authors know through experience that the lack of a high-level model to facilitate team communications is common to a number of organisations. They are confident that the solution developed by Landmark may therefore be applicable outside the context that originally gave rise to it.

3. Review of approaches to designing web based applications

Initially Landmark hoped to adopt an existing approach to meet their needs for a high-level design model but found none suitable. This section reviews some of the existing approaches to modelling web-based applications.

Approaches to developing hypermedia systems have been around much longer than the web and have influenced the design of web based systems. Key approaches have focused on the development of models and methodologies. Initially the main emphasis was on navigation and the user interface. Server-side technologies were not well developed and consequently there was little need to model this area. Approaches to the design of early web based systems had a lot in common with developing large information systems and so concentrated on how to organise and provide the user with lots of information.

Some of the approaches are now obsolete (e.g. HDM [3]) but have evolved or influenced later approaches. In recent times more sophisticated approaches have been developed that focus on languages, architectures, methods, methodologies and processes to develop web-based applications. However, these still typically only address part of the development process and further research is required to find a solution or solutions that meet all the requirements of developing a wide variety of web based systems.

There are a number of classifications that have been use to categorise approaches to developing web based applications. For example by lifecycle phase (e.g. Escalona [4]), Methodology type (e.g. object-oriented such as OOHDM – Object Oriented Hypermedia Design Model [5] versus non object-oriented such as RMM – Relationship Management Methodology [6]) etc. Our approach to classification looks at the aspect of the application addressed:
1. Data Modelling based Approaches. This includes those that developed from E-R modelling and database design or modelling of the information stored by the application.

2. User focussed approaches. Typically these focus on navigation, interface design, requirements modelling etc.

3. Process focussed approaches. Here the main concern is about how the information is processed e.g. The OO-Method [7].

4. Life-cycle focussed approaches e.g. UML-Based Web Engineering [8].

5. Struts-specific Modelling approaches e.g. UML Profile for Analysis and Design of Jakarta Struts Framework Based Web Applications [9].

The general trend of more recent methodologies is to broaden the approach to cover more of the lifecycle and address wider issues, for example HFPM [10]. These more recent approaches tend to cover 1-3 above. Although all approaches have some relevance to what we are discussing here, our problem and solution are specific to the modelling of processing of information as it flows through a web-based application. We are therefore focussing on the process aspect of developing applications in this paper.

3.1 Data Modelling based Approaches

Many early approaches to designing web based applications tended to be based on E-R modelling. The first being HDM [3]. Other approaches enhanced this by adding a methodology e.g. RMM (Relationship Management Methodology) [6] and EORM (Enhanced Object Relationship Methodology) [11]. The latter being the first really to make use of an object-oriented paradigm. At this early stage in the evolution of web applications however, the focus was still very much on storage and navigation. As time progressed these approaches were extended further to enhance the use of object orientation and introduce the use of patterns e.g. OOHDM (Object Oriented Hypermedia Design Model) [5] which has become one of the most widely accepted methodologies in multimedia application development.

Later approaches in this genre have tended to focus on automatic generation of pages directly from a database using various notations e.g. WebML (Web Modelling Language) [12] and Model-driven development of Web applications: the AutoWeb system [13]. The main drawback of these approaches is that they do not address the complex server-side processing that is needed for many web applications.

3.2 User focussed Approaches

These approaches concentrate on user modelling end and capturing user requirements accurately. There have been a variety of approaches, some using classic object-oriented techniques to support the development e.g. The MacWeb Hypermedia Development Environment [14]. This was the first approach that really took the user into account in the development. Many approaches ignore the system functionality e.g. WSDM (Web Site Design Method) [15]. The focus here is on
how to design the application based on user groups. SOHDM (Scenario based OO Hypermedia Design Methodology) [16] is based on RMM, OOHDIM and EORM but enhances the methodology to make the development of the system scenario based. Others such as RNA (Relational Navigational Analysis) [17] propose to enhance the approach by emphasising the importance of understanding the environment and the full scope of the problem before tackling it.

3.3 Process focussed Approaches

The main debate in process focussed approaches is about the use of UML and whether it is sufficient to support all the modelling requirements for web based systems. Henniker and Koch [18] argue that UML is powerful enough to model all Web application needs however, their focus is on hypermedia applications which have multimedia elements and hyperlinks and databases in an internet environment. Other approaches, such as those proposed by Mandel et al [19] and Conallen [20,21] argue for extensions to UML models (for example through UML stereotypes) to support the development of web applications.


3.4 Lifecycle Focussed Approaches

These approaches have developed more recently and usually build on past approaches. For example, several of these extend OOHDM. HFPM (Hypermedia Flexible Process Modelling) [10] integrates OOHDM and classic OO tools such as UML and reuses the OOHDM models. SHDM (Semantic Hypermedia Design Method) [24] is also a successor to OOHDM and defines a HyperDe environment which is based on the MNVC framework [25]. This framework extends MVC to allow the designer to enter details of the navigational model and interface aspects to generate complete applications written in Ruby. HyperDe is most suited to the rapid prototyping of web applications. The OO/Pattern Approach [26] is similar to HFPM as both of them propose the use of patterns and OO for navigational and interface design. However, unlike HFPM this approach does not cover the whole lifecycle but was the first proposal to use UML use cases in the analysis phase.

Graph transformations are used for the verification of correctness and adaptability of models. These can generate code automatically and are used by OOWS (Object Oriented Web Solution) [27] and W2000 [28]. Other approaches focus on model driven architectures and development such as MDA (Model-driven Architecture) [29] and UWE (UML-Based Web Engineering) [8]. This latter approach to model-driven development supports the construction of platform independent models and model transformations allowing the software development process to be viewed as a chain of models. Functional models (PIMs) are integrated into a big picture model mainly for the purposes of verification. A merge with architectural models results
in an integrated PIM model covering architectural and functional aspects. The main characteristic of UWE is the semi-automatic, model-driven and transformation based support for the development of a web based system.

3.5 Struts-specific Modelling approaches

Most of these approaches tend to support the model driven design of a Struts application via a tool that typically generates some code. For example Andromda [30] does precisely this. It provides a selection of “cartridges” for different technologies to generate specific code e.g. Spring, EJB 2 / 3, Webservices, Hibernate and Struts. Flux [31] which is an Eclipse plug-in will re-generate (as required) the struts-config.xml files from UML activity diagrams. Exadel Studio [32] generates code from a graphical tree-like representation. However, the graphical representation is just part of an application model and does not contain everything that the developer needs to cover the whole analysis and design phases.

The UML Profile for Analysis and Design of Jakarta Struts Framework Based Web Applications [9] approach proposes five models: a web application model, use case model, flow model, structure model, and presentation model. The flow model shows all possible logical flows using a collaboration diagram. It does not however contain a good overview model.

3.6 Why a new approach is needed

Many of the key current approaches for producing a development overview of a website are based on UML. Whilst UML and the approaches and extensions described above have their merits for documenting and designing the detail of OO systems and their internal relationships, they do not have the sort of “overview” model that can easily be used to meet the needs described in section 2 of this paper. UML was designed for use by highly skilled OO software engineers working with clearly defined Objects. A Web Application calls upon different skills and often uses developers who do not come from a formal software engineering background. These skilled individuals often struggle to understand a UML based diagram. UML encourages a system to be divided up into a number of different diagrams. This makes it difficult to form any sort of high-level overview because the individual has to assimilate and cross-reference multiple models (using different notations) at the same time. In practise, this is very hard to do for anything but trivial systems.

Many UML models such as Communication diagrams tend to use a restricted number of element types. This does not allow the eye to “scan” the model and then home in on the particular area of interest. The authors have found UML to be unsuitable for producing an overview model of a web application because:

- The notation used must be easily associated within the application domain (e.g. web pages, Java Beans and Java Classes) and understood by developers who are not formally qualified Software Engineers.
- Both the user flow through the site and the data that is passed around needs to be captured on the same model.
- The number of individual models that needs to be maintained needs to be kept to a minimum.
- None of the existing UML-based approaches provide a sufficiently good overview “map” to guide the developers and assist discussion.

Solutions that only concentrate on one aspect of a web application have similar issues.

4 Description of COMPACT

The model needs to support the early design discussions between the different developers in a web team each with different skill-sets as well as ensure a definitive record of the decisions is made. Specifically it needs to:
- Show all the pages in the website/web application.
- Show how the individual web pages are linked together.
- Show what transitions cause the user to move from one page to another.
- Identify the major external systems (e.g. Database, external data feed etc.) that the web application interacts with.
- Show what server-side components are used in processing a user request.
- Give some indication as to what data is sent between the web pages and the server-side components.
- Be understandable by developers who do not have a formal software engineering background. It cannot assume an in-depth knowledge of UML.

The solution proposed is a model to help developers navigate their way around the application. It does not record minute detail of how every Java class/server-side component is built but is designed to help the project team:
- Work out where they are and where they need to go.
- Stimulate debate about the best way of getting from “A” to “B”.
- Introduce new developers to the project quickly by giving them access to a context within which to understand the detail they will rapidly encounter.
- Allow the development work to be easily split between developers.

While designing the shapes that form the elements of the COMPACT model, the following principles were followed:
- The final model should allow the individuals in a team to scan the information visually to build a mental overview of the application without the need to read each item of text on the model.
- Colour as well as shape should be used to impart meaning, as proposed by Coad et al [33], but each shape should also be unique for printing in black and white.
- The model should support multiple pages easily because only the very simplest web applications can be described on one sheet of paper.

The following section describes the main graphical elements that make up the COMPACT web design approach and provides an example of its use in Figure 1, which is taken from a real project. In the case of the Landmark, the model shapes
were implemented as a custom Microsoft Visio stencil. Additional graphical elements are also used but have been left out of this discussion to aid clarity.
<table>
<thead>
<tr>
<th>Element Shape</th>
<th>Element Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="JSP page name" /></td>
<td>A standard “vanilla” JSP or HTML page. The name of the page is its full filename with its extension, but without the pathname.</td>
</tr>
<tr>
<td><img src="image" alt="JSP page name" /></td>
<td>A JSP/Struts Form page. The name of the page is its filename with its extension, but without the pathname. The grey coloured “banner” at the top of the shape simply serves to indicate that this page is a Form page that will be submitting HTTP Form data.</td>
</tr>
<tr>
<td><img src="image" alt="Form Bean Name" /></td>
<td>A Struts Form Bean. The name of the Form Bean is its Class name without an extension. The Form Bean is connected to the transition between the Form page that populates it and the Struts Action that processes it.</td>
</tr>
<tr>
<td><img src="image" alt="Transition" /></td>
<td>A transition between items within the model. This shows the routes that can be taken between a JSP/HTML page and the Struts Action(s) that processes the user’s request. Each transition is labelled with either the web event that triggered the transition and/or the Struts Forward Name (e.g. “Success”, “Failure” as stated in the Struts config XML) within square brackets.</td>
</tr>
<tr>
<td><img src="image" alt="ActionClassName" /></td>
<td>A Struts Action Class, drawn with a dark yellow background. The top part gives the name of the Struts Action as it appears in the Struts Config XML file. The lower part is the Action Class name without an extension.</td>
</tr>
<tr>
<td><img src="image" alt="HTTP Data" /></td>
<td>Depicts a single element of HTTP data. Usually these will be simple HTTP parameters.</td>
</tr>
<tr>
<td><img src="image" alt="Java Bean" /></td>
<td>Represents a single Java Bean that has been populated by a Struts Action.</td>
</tr>
<tr>
<td><img src="image" alt="Data Collection" /></td>
<td>Represents a collection of identical Java Beans/Data Elements. This representation can be used for both HTTP Data and Java Beans.</td>
</tr>
<tr>
<td><img src="image" alt="External System" /></td>
<td>Represents some system outside the scope of the web application e.g. databases, Web Services, other websites etc. This is drawn with a green background.</td>
</tr>
<tr>
<td><img src="image" alt="Ext. Sys. Data" /></td>
<td>An item of data that is exchanged with an external system. This is also drawn with a green background. Using these elements makes it much easier to identify inter-system dependencies.</td>
</tr>
</tbody>
</table>
4.1 Methodology

This section describes a typical methodology for using the COMPACT model to support the quality of the design and the communication between the team members.

1. Once a project starts, a high level Requirements document is produced which
lists the “What” about the application. More detail is added and a set of screen mock-ups are produced concentrating on the user’s “journey” through the proposed application.

2. After the screen mock-ups and user flow has been agreed the development team produces an outline COMPACT design. This shows the individual web pages linked to place-holder Struts Actions. This iteration identifies all the transitions from every page and those pages submitting form data.

3. The next iteration of the COMPACT design adds any external systems that will need to be accessed e.g. databases or web services.

4. The third iteration adds the data that will be passed between the various components on the page. This enables the front-end and server-side developers to negotiate a “contract” regarding the data that will be passed between them.

5. After a variable number of further iterations the degree of “churn” in the COMPACT design (i.e. number of changes made) reaches a sufficiently low level that the developers can begin the detailed design of the implementation. At this stage there is sufficient information on the COMPACT model for a fairly complete Struts configuration file to be produced and the supporting skeleton Java Classes and Ant build file to be generated.

6. The project now enters the implementation phase where the COMPACT design is turned into real web pages and their supporting Struts Action classes.

7. It is at this point that the COMPACT design model has a further use which is tracking the work that has been completed. By simply crossing out (on a hardcopy of the design) each web page and server-side component as it is completed it becomes very obvious if something has been overlooked.

8. When all the web pages and Struts Actions on the COMPACT model have been implemented the web application is ready to test and deploy.

5 Conclusions and Further Work

This paper has described the COMPACT web design model and explains how it has been used to enhance the quality of the software development process by supporting team communications at the early stages of the design process. The model provides an overview of both the key website navigation and server-side functions that need to be modelled during the development of a web application and also serves as a focus to stimulate debate and discussion about the design between the different types of developers.

Future work involves the development of a non-Struts based version for use in the design of general web applications. This will be trialled with students work at the University of Greenwich before being used in a live project.

The first COMPACT design template was produced in late 2004 and has been used on a number of large web application development projects demonstrating that it supports the development of a quality product and is what practitioners need. Two of these have been platforms for ordering the Environmental and Planning reports that Landmark produce one of which takes over 2000 complex orders per day. The
design template is in now in its sixth revision and has shown itself to be an invaluable model for team use. Further details of these projects and the Visio template used can be downloaded from www.compact-web.org.

References