SIMPLE TECHNIQUES FOR COMPLEX WEB APPLICATIONS

Gary Griffiths, Mike Lockyer, Briony Oates, Barry Hebbron
School of Computing
University of Teesside
Middlesbrough,
TS1 3BA  UK

ABSTRACT
In this paper we present an integrated approach to the development of Complex Web Applications. The principal philosophy of the development is to support simplicity, consistency, completeness and traceability. We present details of the method, process model, notations and tool, and draw conclusions about the suitability of such an approach for inexperienced developers.

KEYWORDS
Web Development, Web Engineering, CASE.

1. INTRODUCTION
In this paper we explore techniques and tools to support the analysis, design and implementation of web systems that are simple enough to encourage their use. Our principal focus is on an approach that would be ideal for inexperienced users in industry or for students. There are many novice web developers in industry, and that is likely to be true for some time given the forecasts of the extra numbers needed in the next few years [Andersen, NCC]. So, simple methods are in the interests of commercial developers.

This is backed up by recent research [Barry] that showed that current methods are not being used in industry, but that the simpler techniques (like storyboards and navigation charts) are popular, usually within an in-house method. It has often been the case that education is neglected when considering methods [Lockyer, 1989] and simple, easy-to-learn methods would also be appropriate for students.

2. TYPES OF WEB SYSTEM
Conallen [Conallen] suggests that there are essentially two types of web-based systems – Web Sites and Web Applications. The former are information sites, the latter are more like traditional information systems that operate over the Internet. Within Web Applications, we want to distinguish between simple and complex systems. Simple Web Applications might typically involve some content pages plus some ‘shopping’ pages involving some straightforward database interaction. Complex Web Applications are really full-blown information systems, with complex database interactions, using the Internet as a user interface.

We have considered the development of Web Sites and Simple Web Applications in another paper [Griffiths, 2002], so for the remainder of this paper we will concentrate on Complex Web Applications. Complex systems could be developed using traditional Information Systems methods like Yourdon [Yourdon] or UML [Rational] augmented with techniques for web page design. Alternatively, specialised web methods could be used, like WebML [Ceri] or Conallen’s adaptation of the UML [Conallen]. We wanted to explore the appropriateness of structured methods [Yourdon, Ashworth, Griffiths 1998] for web development for simplicity and for teaching.

1068
3. PHILOSOPHY

In order to provide a simple method for developing Complex Web Applications (CWAs), we propose the Simple Web Method (SWM): a strong integration between three support mechanisms.

1. A Process Model – provides the steps
2. A Web Notation – provides a set of modelling notations
3. A Software Tool – automates modelling and performs consistency and completeness checks.

3.1 The Process Model

Before we look at the Design and Build stages of SWM for Complex Web Applications in section 5, we should briefly consider the Analysis stage. Planning, analysis and maintenance would be very similar for Web Sites and Web Applications. In fact, the only difference is the Content Analysis and Modelling step in the Analysis stage. For Complex Web Applications, we are not trying to model content so the step is inappropriate. However, the essence of the step – to define the scope of the system – is still appropriate. We do this in a similar way to defining the scope of a development in structured methods. We draw a data flow diagram to model the whole system and then draw a dotted line to denote the ‘Web Boundary’ (rather than the ‘Automation Boundary’ traditionally).

3.2 The Web Notation

The Web Notation essentially comprises three models: an event list, an Entity Relationship Diagram and a Web Flow Diagram. The first two of these are well known [Yourdon]. The notation used within our Web Flow Diagram uses elements of Data Flow Diagrams [Yourdon] and WebML [Ceri]. To visualise richer concepts, and yet control the complexity of those concepts, we have extended and enhanced the original notations [Lockyer, 2002].

3.3 The Software Tool.

The software tool ASCENTW is a direct descendent of ASCENT [Lockyer 1989, Hoggarth]. It supports the web notation defined above and provides code generation to ASP, PHP and ASP.NET. It is important that the tool supports simplicity. Consequently the tool helps developers control the complexity by providing explosions and drill down mechanisms enabling them to perceive the system at an appropriate level of abstraction required at any particular point in the process. The tool filters the models to present the most appropriate view of the model. It allows the developer to take a user view or a data only, structure only, event only view. When the tool allows the model to be compartmentalised on a particular event, then the response thread for that event can be isolated, presented and animated. (See [Lockyer, 2002].)

4. DESIGN AND BUILD STEPS FOR COMPLEX WEB APPLICATIONS

The design and build of Complex Web Applications are described briefly below. More detail can be found in [Lockyer, 2002].

List events and entities - a rough list of events and entities within the system are made. This is not an exhaustive list at this stage and the process is more akin to brainstorming than to a methodical approach. Relationships are not identified for entities until later.

Consider events methodically - the rough event list is considered in detail and broken down. Events may be composed of sub-events, which we call tasks. Also, each task may have system responses. All of these are recorded at this stage. Sometimes storyboards are used, not to map the layout of the pages, but to help think about events, tasks and responses. To consider the events in this level of detail helps discover and focus on missing events, missing entities, attributes for entities, data administration and referential integrity.

Verify events with users - the full list of events, tasks and responses are discussed with the users for verification. This usually results in some changes to the list.
Complete attributes for entities - some attributes for entities will have been recorded when considering the events methodically. All of the entities must now be examined in turn and a complete list of attributes recorded for each.

Generate test plan from full event list - it is relatively straightforward to define an initial test plan from the full event list by defining a test for each event, checking that each task is accomplished successfully.

Build entity relationship diagram (or class diagram) - a hierarchy of entities would indicate an OO approach and would lead to modelling the data in a class diagram. A complex structure with many to many relationships would indicate a structured approach and would lead to modelling the data in an entity relationship diagram. Relationships are added to the entities to build a full entity relationship diagram. Types of user, obligatory/non-obligatory relationships and referential integrity are all considered to complete the model.

Build the database - the database is implemented from the entity relationship diagram, defining required fields and data types. Stored procedures and the database access layer are also considered. Stored procedures are SQL statements or queries that are stored in the database, not in separate pieces of code. These are more efficient and more secure, but difficult to debug. These would not be used in the first prototype, but might be used subsequently. The database access layer is a set of functions that allow database access without SQL. It is particularly useful with an OO approach although there are substantial benefits for any system. It needs to be decided at this point because it is difficult to change later.

Draw storyboards by going through each event - each event could generate many storyboards (or pages). Exceptions should be identified and the delivery technology for the web front end to the system should be considered e.g. HTML, Flash. The links to the database from each page should be considered.

Develop page templates - this step is interleaved with the previous step to draw storyboards. It is useful to consider if the templates form a set or a hierarchy at the outset as this can lead to coding sharing and simplifications. When developing the templates, ‘controls’ may be developed which sit in parts of the page and provide the links, content or functionality. Visual style is considered and style information is defined, like in a Cascading Style Sheet. Navigation is also considered.

Verify storyboards and page templates with users - storyboards and page templates are shown to users and amendments made based on their feedback. The presentation to users could be on paper, in a word processor or with an HTML mock-up created quickly with something like Dreamweaver.

Integrate storyboards for all events - the sequence of pages and database table accesses for each event should be modelled graphically. The resulting diagram is not unlike a web-oriented data flow diagram.

Prepare for code generation - for most pages we need to define the purpose of the page (or sub page), give it a pagetype and define the SQL statement. The types of page (or sub page) and implementation strategy is system & language dependent. For instance, in Active Server Pages (ASP) and PHP we would need to implement a number of separate server-side pages to perform adding data, updating data and sorting data. In Microsoft’s .NET version of ASP, only one page needs to be implemented as all that functionality can now exist in a single page.

Implement access for different types of user - we need to model the different types of user that the system needs to respond to and to define types of users’ access rights to pages. This needs to be supported at the code generation stage.

Refine the HTML - having written or generated the server-side pages needed to run the application, it is usually necessary to refine the HTML in order to enhance the user-interface.

5. CONCLUSION

In this paper we have distinguished between different types of web system and argued for the use of simple techniques. We have revisited our Simple Web Method (SWM), which was formulated with the development of Web Sites and Simple Web Applications in mind, and considered the appropriateness of the stages and steps for developing Complex Web Applications. All of the stages, and most of the steps, are appropriate for both types of web system.

We have proposed the philosophy for our Simple Web Method that it exists as a strong integration between three support mechanisms: the Process model, the Web Notation and the Software Tool. We have set out the activities that are different for Complex Web Applications. In doing this SWM has moved towards
being a process model rather than a simple method. That is, SWM does not now set out a sequence of steps to be followed in developing any web system. Rather, it defines a menu of possible steps, and a method (or process) would be instantiated from this to develop a particular system.

There are a number of areas for further work, particularly method guidance and integrating the Windows-based ASCENTW with our web-based diagram editor to enable a complete internet based development environment.

REFERENCES

NCC, 2000, *Salaries and Staff Issues in IT 2001*, NCC.