Visual Methods for Web Application Design

Robert Chatley, Jeff Kramer, Jeff Magee, Sebastian Uchitel
Dept of Computing, Imperial College London
{rbc,jk,jnm,su2}@doc.ic.ac.uk

Abstract

The paper outlines a tool-supported approach to the design of Web applications. Behavioural models are augmented with web-based simulations of user interfaces to permit validation and usability assessment of systems by end users in advance of implementation. The goal is to correct architectural design decisions that adversely impact usability early in the design cycle when correction is relatively inexpensive. The behavioural model of a system captures the interactions between the different users roles and the set of components that constitute the application. A visual scenario-based language is used to specify interactions and the tool LTSA-MSC is used to synthesise the required behavioural model. The tool supports a visual representation of this model that is animated in response to user-interaction with the simulated Web interface. The combination of these facilities permits agile incremental elaboration of a system design.

1. Introduction

Usability is increasingly considered a key software quality attribute, particularly crucial in web-based applications where competition is literally one click away [12]. To effectively design usable web applications, usability issues should be addressed early in the development process. To this end, high user involvement is crucial, and so is support for rapid iterative improvements based on user feedback. In addition, as web applications are commonly distributed, concurrent and multi-user, usability assessment should not be restricted to single user observation but should also consider how usability is affected by concurrent users collaborating (or interfering) with one another.

In recent years a number of techniques for usability assessment have been developed [1][2]. These techniques are mainly based on observing users interactions with the systems and as such require a working implementation of the system. This means that usability assessment can only be carried out late in the development process. At this stage it is very expensive to go back and make major architectural changes to the design [3]. It also makes an agile iterative improvement of the application's usability difficult to achieve.

Behaviour models are precise, abstract descriptions of the intended behaviour of a concurrent and possibly distributed system. We use a behaviour model to capture the interaction between users and the different components that form the target application. This model is simulated to allow end-users of the system to interactively explore the specified behaviour for the purposes of both validation – is this the required behaviour, and usability – is this the best way of performing the interaction/task. The behaviour models we use in our approach take the form of labelled transition systems (LTS)[5] –examples of which are shown below in Figure 1.

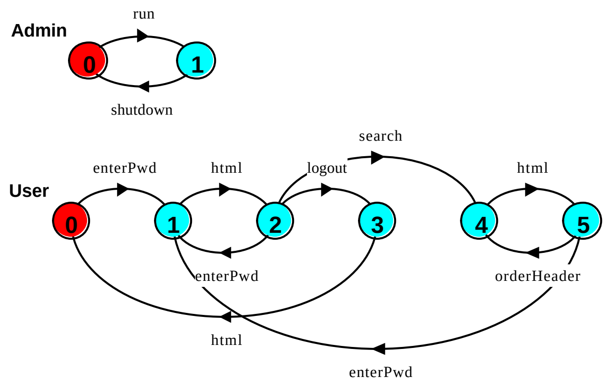

Figure 1 : LTS for Admin and User components.

We have developed the tool *Labelled Transition System Analyser – Message Sequence Chart version* LTSA-MSC [4, 6, 9] to construct these LTS models from a set of scenarios specified in a visual message sequence chart notation – an example of this notation is illustrated in Figure 2. The automated synthesis procedure that builds the LTS behavioural models such as those depicted in Figure 1 from message sequence charts [11] is described in

[7]. Behavioural models of web-applications built using the tool are unsurprisingly easier and faster to build than the system itself since the model focuses only on interaction. Essentially interaction behaviour is captured abstractly as a set of sequences of symbols that the model accepts or generates. Each symbol models an input, output or a processing action that the system performs. However, one cannot ask non-technical end-users to validate that the behavioural model satisfies the requirements by presenting them with traces of these abstract symbols.

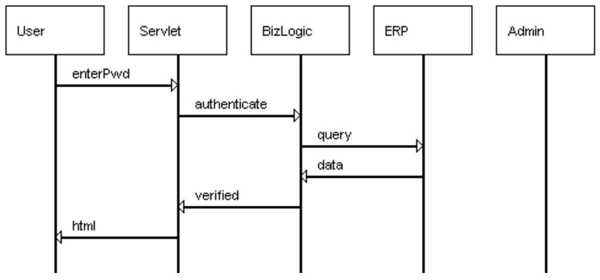

Figure 2: *Login* Message Sequence Chart.

To address this problem, we have extended our existing behaviour analysis tool LTSA-MSC [6] with web server capabilities and mechanisms for associating web page elements (graphics, buttons, etc.) with behaviour model symbols. This representation of model symbols as web-page elements permits users to explore the behaviour model using a standard web browser.

2. LTSA-MSC/Web Browser interaction

The LTSA tool has been extended using a "plugin" to allow it to provide the engine for simulations of web applications. The plugin provides an interface so that the model can be viewed in a standard web browser. It adds a mini webserver to the LTSA so that it can communicate with a standard web browser by means of the HTTP protocol. Essentially, the web animator plugin allows us to associate fragments of HTML with different possible actions. These can be hyperlinks, buttons or any other interactive element commonly found on web pages. The plugin dynamically composes a web page from these fragments and serves it to a web browser to display.

When the user is presented with such a webpage, they can click on any of the links or buttons on the page, which will cause the browser to send an HTTP request back to the server. The server analyses this request to detect what action the user has requested and triggers an appropriate transition in the LTS. A new webpage is served to the user with a new set of available actions.

Extra decision logic has been added so that it is possible to make a distinction between actions that are carried out by different parties. This allows us to distinguish between actions performed by users and those that are carried out by components of the system without any user intervention.

The tool has been extended further to allow multiple users to interact with the model concurrently. The threading model in the web server was extended to be able deal with multiple concurrent requests from different web browsers.

2.1. Configuring a Simulation

The LTSA produces an XML document describing the available transitions each time that a new state is reached. An XSLT [8] transformation is applied to this XML document based on an XSL stylesheet. This stylesheet describes a transformation from XML to HTML which defines the visual appearance of the web pages. This HTML is then sent over the network via HTTP to the browser where it is rendered. A separate stylesheet is written for each application that is simulated, as their visual appearance and their alphabet of actions will differ. References to images and all the standard HTML elements can be included in the XSL. Figure 3 shows a fragment of XSL which gives the HTML to output when the *enterPwd* action is enabled. The figure also shows the result of rendering this HTML, which includes an image button.

```
<xsl:when test="name='enterPwd'">
  <input type="text" name="login"/>
  <input type="password" name="passwd"/>
  <input name="{number}" value="Login"
   type="image" src="/enter.jpg" />
</xsl:when>
```

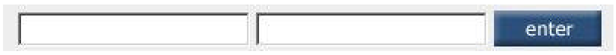

Figure 3: A fragment of XSL and the rendered HTML

XSL stylesheets are a standard way of expressing a transformation from XML to another data representation, in this case HTML. This technique is itself commonly used in web and e-commerce applications. Because the output is standard HTML, we can achieve an interface which is very close to that that might be used in the final system.

2.2 Case Study

To validate these modelling techniques in action and provide examples of the sort of problems that can occur with concurrent use of web applications, we applied the techniques described above to an industrial case study.

This is a model of the eSuite[1] e-commerce application that permits multiple users. The eSuite application has a typical tiered architecture comprising a web server hosting a Java servlet for presentation, a business logic layer and an ERP (Enterprise Resource Planning) system that is effectively treated as a database. In a previous model [10], a single user interacted with the system to perform tasks such as logging in to the system or searching for an order that a customer had placed. The new multi-user facilities of the LTSA-MSC tool have allowed us to introduced a new class of user that interacts with the system, an administrator who controls the operation of the system. The examples of Figures 1,2 & 3 are taken from the Case Study.

5. Conclusions

In summary, we have outlined an approach for simulating the behaviour of web-applications based on a model of the system at the architectural level. By providing a web-based mock-up of the final user interface for controlling the simulation, both validation against end user requirements and usability assessments of the system can be carried out earlier in the design process than is usually the case. This means that design changes informed by this usability assessment can be made at this early stage. By providing a visual representation of the behaviour models that drive the simulation and a visual scenario-based language for specifying these models, design changes can be introduced on the spot, allowing for an agile incremental elaboration of the system design. By allowing multi-user simulation we can identify undesirable system behaviour that is the result of unexpected user interaction.

The LTS-MSC tool and Web plugin can be fo und at: http://www.doc.ic.ac.uk/ltsa/

Acknowledgements

We gratefully acknowledge the support of the European Union under grant STATUS (IST-2001-32298), and the partners working on the STATUS project.

References

.

[1] Preece, J., Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey. *Human-Computer Interaction*. Addison Wesley, 1994.

[2] Constantine, L.L., L.A.D. Lockwood. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley, New York, NY, 1999.

[3] Brooks, Jr., F.P., 1995: *The Mythical Man-Month: Essays on Software Engineering, Twentieth Anniversary Edition,* Reading, MA: Addison-Wesley

[4] S. Uchitel, J. Kramer and J. Magee. *Negative Scenarios for Implied Scenario Elicitation*, Proceedings of 10th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE'02)

[5] Magee J., and J. Kramer, *Concurrency – State Models and Java Programs*. John Wiley & Sons, March 1999

[6] S. Uchitel, R. Chatley, J. Kramer and J. Magee. *LTSA-MSC: Tool Support for Behaviour Model Elaboration Using Implied Scenario*, Proceedings of TACAS 2003, LNCS 2619, p597-602, April 2003.

[7] S. Uchitel, J. Kramer and J. Magee. *Synthesis of Behavioural Models from Scenarios*, IEE TSE, Vol 29, No 2, p 99-115, Feb. 2002

[8] J. Clark, “XSL Transformations (XSLT) Version 1.0”, http://www.w3.org/TR/xslt

[9] Uchitel, S., J. Kramer, and J. Magee. *Detecting Implied Scenarios in Message Sequence Chart Specifications* in Joint 8th European Software Engineering Conference (ESEC'01) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE'01). 2001. Vienna: ACM Press

[10] R. Chatley, S. Uchitel, J. Kramer and J. Magee, *Model-based Simulation of Web Applications for Usability Assessment* to appear in ICSE 03 workshop "Bridging the Gaps Between Software Engineering and Human-Computer Interaction", May 3-4, 2003 in Portland, OR.

[11] ITU, *Message Sequence Charts*, 1996, International Telecommunications Union. Telecommunication Standardisation Sector.

[12] Nielsen J., *Designing Web Usability*, New Riders Publishing, Indianapolis, 2000

[1] eSuite is a product developed by LogicDIS, one of the partners in the STATUS project (European Union grant IST-2001-32298).