Model based web interfaces' analysis

José Creissac Campos

Universidade do Minho, Departamento de Informática
Campus de Gualtar, 4710-057 Braga, Portugal
jose.campos@di.uminho.pt

Abstract

Tools exist that look at the usability of sites from a static perspective – for example, looking for broken links. Other tools analyse web server's logs to understand how users behave when using the site. While these tools are helpful, they can only be applied after development and deployment.

This paper investigates how a model checking based approach can be applied to web sites' design to reason about their behavioral properties from the early stages of development. The goal is that analysis of possible interactions between users and a site might be performed before actual development and deployment of the site.

1 Introduction

Usability plays a relevant role in the success of a web site. The open nature of the web means users are more likely to abandon the interaction with a web site, and move somewhere else, when they come upon difficulties. Particularly relevant is the issue of navigating through a web site.

Tools exist that look at the usability of sites from a static perspective. They do this by using a specified set of usability guidelines, and testing the web pages against them. Examples of such tools are WebSAT (NIST 2005) or A-prompt (UToronto 2005). These tools are an improvement over HTML or link validation (which only consider the structural quality of the site), and are able to give insight into the quality of the sites' design. However, they give little indication on how the site will actually be used by users to achieve their goals. This happens because they do not take into full consideration the dynamic aspects of user behaviour

For large or complex sites, it is important to define a navigation structure that supports users in an adequate manner. Tools such as Watchfire's webQA (Watchfire 2005) address this by simulating the use of the site in order to validate specific use behaviours. This means, however, that only pre-determined behaviours are analysed. Unfortunately, this leaves out, for example, opportunistic interactions that might happen when navigating the site.

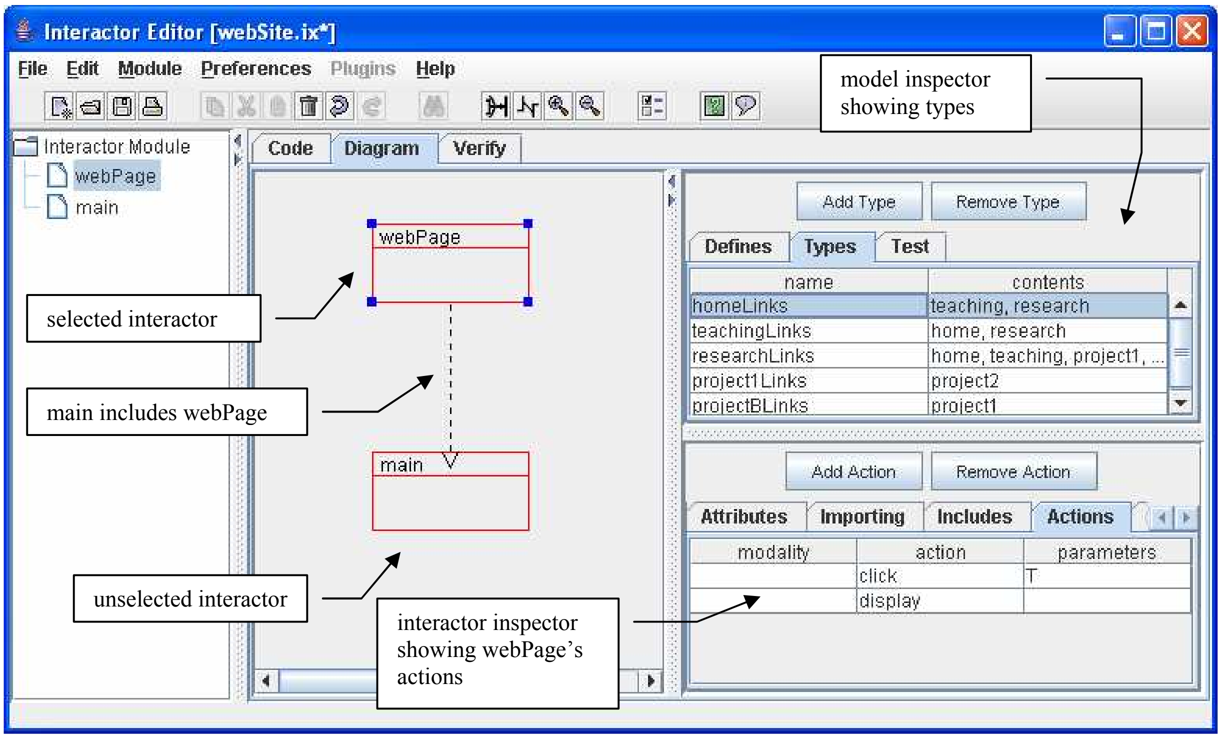
Web traffic analysis tools analyse web server's logs to understand how users have actually behaved when using web sites. ErgoBrowser (ErgoLabs 2005), for example, records a number of parameters of use during web site testing for later analysis. Another possibility is to analyse web server logs in order to get data from real usage of the site. More recently packet sniffing has been introduced as a way of avoiding the need for logs by collecting information directly from the TCP/IP packets that are sent to and from the web server.

While web traffic analysis tools are helpful, they can only be applied after development and deployment. At this stage making changes to the site's structure can be costly both in terms on time and money. One possible approach to address this is to use model based approaches to reason about the design before actual development and deployment. This paper investigates how this type of approach can be applied to web sites' design. The goal is to perform analysis of possible interactions between users and a site before actual development and deployment of the site.

2 Model based usability analysis

Model checking has become an established verification technique for the model based analysis of behavioural properties of reactive systems (Clarke, Grumberg & Peled, 1999). In the last decade the applicability of automated reasoning tools (and especially model checking) to the analysis of usability issues from interactive systems models has been investigated by a number of authors. Examples of work in this field are (Paternò, 1995), (Bumbulis, 1996), (Campos, 1999) or (Loer, 2003).

A tool has been developed (see Figure 1) that supports a semi-automated model-based analysis process of behavioural issues of the interaction (Campos & Harrison, 2001; Campos, 2004). Models are built around the notion of interactor as put forward in (Duke & Harrison, 1993). Each interactor defines a number of typed attributes and actions on those attributes. Modalities (for example, visible or audible) can be assigned to attributes or actions. The effect of the actions in the attributes is expressed in a Propositional Production System (PPS) like style. For each action we state what the values for the attributes are after the action occurs. The use of a PPS like style of specification to model user interfaces has been found useful and easy to use (Curry & Monk, 1995). Interactors can be composed to build complex models.



name	contents
homeLinks	teaching, research
teachingLinks	home, research
researchLinks	home, teaching, project1, ...
project1Links	project2
projectBLinks	project1

modality	action	parameters
	click	T
	display	

Figure 1: The interactors editing environment

At the moment the editor provides two editing modes. In code mode models are written textually in a domain specific languages defined specifically to that effect (MAL interactors). A description of the MAL interactors language falls outside the scope of this paper, readers are referred to (Campos & Harrison, 2001). In diagram mode models are built graphically. Interactors and their relationships are described in a class diagram-like notation, and inspectors are used to edit each interactor.

Once the models are developed they can be translated into the SMV model checker input language (Clarke et al., 1999) for verification.

3 Modelling of web sites

As stated in the introduction, our goal is the analysis of behavioural aspects of the interaction between a site and its users. To achieve this goal, web sites are modelled using MAL interactors.

Models are built from a generic model of a web page. To allow for this we have define an interactor webPage. The webPage interactor captures simply whether the page is being displayed or not (through a Boolean attribute – rendered), and defines two actions (see Figure 1): click models the action of clicking an available link; display models the action of rendering the page to make it visible. The click action is parameterized with an enumerated type. This type has a value for each page to which the current page has a link to (see Types in Figure 1).

A model of the web site's navigation structure can be composed from webPage interactors. An interactor is included for each page/frame set in the site, and then axioms are defined that model the links between the pages/frame sets in the site. For each link between two pages in the site, a axioms is created. To simplify the writing of the models a shorthand notation has been introduced into the MAL interactors language. The expression link(p1, p2) means that a hyperlink exists from page p1 to page p2. The tool replaces this expression using the following axioms:

- per(p1.click(p2)) → p1.rendered – the hyperlink can only be clicked if the page where it resides is visible;
- [p1.click(p2)] !p1'.rendered & p2'.rendered – if the hyperlink to p2 is clicked, then page p2 gets displayed.

Additionally, axioms must be introduced to define when a page can become rendered (basically, whenever a hyperlink to the page is clicked).

Interactor webPage provides a very basic model for web pages. This has the advantage of smaller and easier to write models, but little information can be expressed for each page (only the links to other pages are modelled). If a more complete version of the model is needed, we can specialize webPage in order to include the needed information. For brevity we address the simpler case only in this paper.

The models above can be obtained in one of two ways: by a modelling process prior to the actual development of the site, or by reverse engineering the models from an already existing site.

3.1 Forward engineering

Forward engineering is supported by the tool presented in Section 2. We illustrate the approach with an example. For brevity sake we will use a simple example, not a complete web site analysis. Nevertheless it will be enough to illustrate the main steps in the approach. The web site's structure is presented in Figure 2. The figure represents the different pages in the site and the hyperlinks between them.

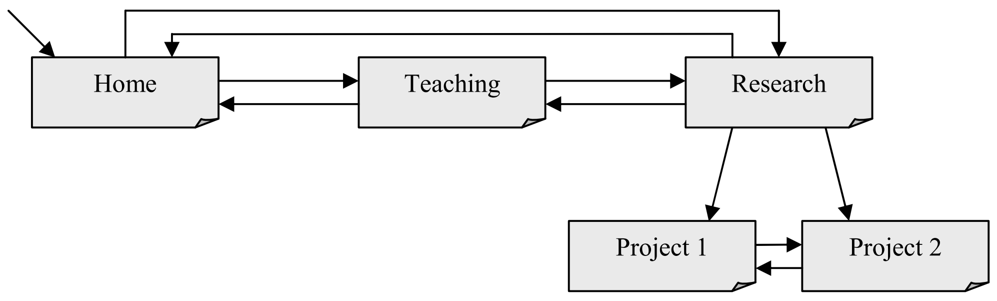


Figure 2: Example web site's structure

In order to model this site, the different sets of pages that can be reached from each page must first be defined. In the current case they are:

- homeLinks = {teaching, research}
- teachingLinks = {home, research}
- researchLinks = {home, teaching, project1, project2}
- projet1Links = {project2}
- project2Links = {project1}

The web site's model can then be built by aggregating five webPage interactors, each parameterized by one of the types above. Finally, the links between pages must be defined using axioms. For the case of the Home page the following axioms would be needed:

- links(home, teaching)
- links(home research)

3.2 Reverse engineering

In the case of reverse engineering, the models are deduced directly from the code of the web site. For the basic situation of considering the link structure only, the process is relatively straight forward. The above process of identifying the hyperlinks of each page and building the model can be performed automatically.

For the case where more expressive power is needed from the model, then a user guided approach must be used. First the site's structure is deduced automatically, then user intervention is needed in order to identify salient attributes of each page.

We have started the development of a new version of the tool in which this reverse engineering process will be supported.

4 Analysis

After the modelling is done the analysis can begin. Typically questions will be asked about the possible behaviours of the model. These questions must be expressed in terms of reachable states in the model, and are written using CTL (Clarke et al., 1999). CTL (Computational Tree Logic) is the logic used by SMV to express properties for verification. The SMV tool is used to check the model for states that violate the property. If any such states are found, a counter-example is produced (if possible) that illustrates a behaviour violating the property. These counter-examples are useful in identifying scenarios where usability problems might arise.

As an example, consider that we would like to test whether the home page is always accessible during navigation of the site. To achieve this we could test the following property:

AG(EF(home.rendered))

which reads: it is always the case that we can get to a state where the home page is being displayed. When the verification of the property is attempted, it fails. The trace produced indicates that the problem lies in page Project 1. In fact, both page Project 1 and page Project 2 miss links to allow navigation to the upper level pages. In the current example the problem is fairly obvious. Note, however, that we could have a larger subset of pages with links between them but not to other parts of the site. That situation would be harder to spot manually but would be as easily detected by the verification process as in the present case.

We can correct the above problem by adding a link from pages Project 1 and Project 2 to page Research. In that situation the above property checks without problem. We could then refine our requirement and test whether the home page of the site is always reachable in a specified number of clicks. If we want the home page to always be available with one click, the property to check is:

AG(home.rendered | EX(home.rendered))

which can be read as: it is always the case that either we are viewing the home page or we can do so in one single step. In this case the property fails. In fact, from Project 1 and 2 the minimum number of clicks needed to reach the home page is two.

This type of analysis enables the identification of structural problems in a web site's design (for example, is a link missing), but also the analysis of more subtle properties of the dialogue induced by that design (for example, will the user be able to quickly return to the home page of the site is (s)he gets lost?). The two properties presented were simple since, due to space limitations, the goal was to give a flavour of the approach. More complex analysis is possible (see next section).

5 Conclusions and Future work

We have developed a tool for the semi-automated analysis of behavioural issues of interactive systems. In this paper the first steps of an effort to enable the application of the tool to web sites' analysis have been put forward. The language has been enriched with notation specific to web site's modelling, a strategy to encode web sites using MAL interactors has been devised, and a reverse engineering tool is being planed.

The approach described has been one of 'monkey at the keyboard'. That is, the system is analysed with no restrictions regarding what the user will do. When such restriction become needed they can be encoded in the properties (see Campos & Harrison, 2001). The analysis that are possible at the moment include, besides the presented approach, the possibility of encoding knowledge about the tasks the users will perform into the model (Campos 2003) in order to assess whether the system reacts appropriately to expected user behaviour. Conversely, if we extend the model with more information about salient attributes of each page, we can investigate how the information presented at the interface enables (or not) the system to support the users' tasks (by relating actions with the information that supports their execution).

We believe that for large sites this type of approach, when supported by an reverse engineering tool, might prove useful in the analysis of how the web sites under developed will be used. The web site's structure, however, is not the only factor influencing the dialogue. Browsers' features such as bookmarks and back/forward buttons also have an influence. The feasibility of including such aspects in the models is under consideration.

Acknowledgements

The author wishes to thank José Mira for his work on the interactors' editor module.

References

Bumbulis, P. (1996) Combining Formal Techniques and Prototyping in User Interface Construction and Verification. PhD thesis, University of Waterloo.

Campos, J. C. (1999) Automated Deduction and Usability Reasoning. D.Phil. thesis, Department of Computer Science, University of York.

Campos, J. C. (2004) Análise de usabilidade baseada em modelos. In *Interacção 2004 - 1a. Conferência Nacional em Interacção Pessoa-Máquina*, 171-176, Grupo Português de Computação Gráfica, July. *(in portuguese)*

Campos, J. C., & Harrison, M. D. (2001) Model checking interactor specifications. *Automated Software Engineering*, 8(3-4), 275-310.

Clarke, E. M., Jr., Grumberg, O., & Peled, D. A. (1999) Model Checking. The MIT Press.

Curry, M. B., & Monk, A. F. (1995) Dialogue modelling of graphical user interfaces with a production system. *Behaviour & Information Technology*, 14(1), 41-55.

Duke, D. J. & Harrison, M. D. (1993) Abstract interaction objects. *Computer Graphics Forum*, 12(3), 25-36.

ErgoLabs (2005) ErgoBrowser. Retrieved on April 1, 2005 from http://www.ergolabs.com/

Loer, K. (2003) Model-based Automated Analysis for Dependable Interactive Systems. PhD thesis, Department of Computer Science, University of York.

NIST (2005), WebSAT. Retrieved on April 1, 2005 from http://zing.ncsl.nist.gov/WebTools/WebSAT/

Paternò, F. D. (1995) A Method for Formal Specification and Verification of Interactive Systems. D.Phil. thesis, Department of Computer Science, University of York.

UToronto (2005) A-Prompt. Retrieved on April 1, 2005 from http://aprompt.snow.utoronto.ca/

Watchfire (2005) WebQA. Retrieved on April 1, 2005 from http://www.watchfire.com/products/desktop/webqa/default.aspx