Survey of Existing Languages to Model Interactive Web Applications

Jevon Wright **Jens Dietrich**

Institute of Information Sciences and Technology,
Massey University,
Palmerston North, New Zealand.
Email: j.m.wright@massey.ac.nz, j.b.dietrich@massey.ac.nz

Abstract

Over the last few years, the web is establishing increased importance in society with the rise of social networking sites and the semantic web, facilitated and driven by the popularity of client-side scripting commonly known as AJAX. These allow extended functionality and more interactivity in web applications. Engineering practices dictate that we need to be able to model these applications. However, languages to model web applications have fallen behind, with most existing web modelling languages still solely focused on the hypertext structure of web sites, with little regard for user interaction or common web-specific concepts.

This paper provides an overview of technologies in use in today's web applications, along with some concepts we propose are necessary to model these. We present a brief survey of existing web modelling languages including WebML, UWE, W2000 and OOWS, along with a discussion of their capability to describe these new modelling approaches. Finally, we discuss the possibilities of extending an existing language to handle these new concepts.

Keywords: web engineering, models, interactivity, AJAX, RIAs, events

1 Introduction

The World Wide Web started out in the early 1990s as an implementation of a globally distributed hypertext system. Primitive pieces of software called web browsers allowed users to render hypertext into visually pleasing representations that could be navigated by keyboard or mouse. These early web sites were generally static pages, and were typically modelled with languages focused on the hypertext structure and navigation of the web site (Garzotto et al. 1993). The full integration of hypertext with relational databases allowed the creation of data-intensive websites, which also necessitated new modelling concepts and languages (Merialdo et al. 2003).

Currently, the most popular modelling languages for web applications are WebML (Ceri et al. 2000) and UWE (Koch & Kraus 2002). Both of these languages represent web applications using conceptual models (data structure of the application domain), navigational models, and presentation models. As such, the ability to express the interactivity of the application is generally restricted to the navigational models, which allow designers to visually represent the components, links and pages of the application.

These languages are excellent at describing older web applications; however recently the increased use of interactivity, client-side scripting, and web-specific concepts such as cookies and sessions have left existing languages struggling to keep up with these Rich Internet Applications (RIAs: Preciado et al. 2005). In this paper we aim to review these existing languages and identify where they are falling short, and how they could be improved.

This paper is organised as follows. Section 2 is an overview of some of the features possible with rich scripting support. To model these new features, we propose in Section 3 some new modelling concepts for interactive web applications. We present a brief survey of the existing modelling languages WebML and UWE in Sections 4 and 5, and discuss their ability to model these new concepts. We briefly mention W2000, OOWS and other potential languages in Section 6; a summary of our language evaluations are presented in Table 2. In the final section, we discuss our findings, provide an overview of related work, and highlight future work of this research project.

2 New Features

Arguably, the most important recent feature of the web is the ability to run scripts on the client (generally through Javascript). Combined with the ability to access and modify client-side Document Object Models (DOM: W3C Group 2004) of the browser, and the ability to compose asynchronous background requests to the web, these concepts together are commonly referred to as AJAX (Garrett 2005). AJAX allows applications to provide rich client-side interfaces, and allows the browser to communicate with the web without forcing page refreshes; both fundamental features of RIAs.

Technologies like AJAX support thin client applications that can take full advantage of the computer power of the clients. These applications reduce the total cost of ownership (TCO) to organisations as they are deployed and maintained on directly manageable servers, and aim to be platform-independent on the client side. To achieve this, AJAX has had to overcome limitations of the underlaying HTTP/HTML protocols, such as synchronous and stateless request processing, and the *pull* model limitation where application state changes are always initiated by the client[1]. This has resulted in rich applications that use the web browser as a virtual machine.

The impact of these technologies has been significant; new services such as Google Docs (Google Inc.

[1] The opposite to the pull model is the *push* model, where state changes are pushed from the server to the client; this is commonly implemented on the web with pushlets (van den Broecke 2002).

2006) are implementing collaborative software solutions directly on the web, based on the *software as a service* philosophy, and to some degree competing with traditional desktop software such as Microsoft Office. RIAs can also be developed in environments such as Flash, which are provided as a *plugin* to existing web browsers, but can reduce accessibility[2].

One popular example of AJAX is to provide an auto-completable *destination address* text field in an e-mail web application. As the user enters characters into this field, the client contacts the server for addresses containing these characters, displaying a list of suggested addresses. This improves usability, potentially reduces the overall bandwidth of network communication, and improves interactivity and responsiveness.

An investigation of some of the most popular AJAX-based websites on the web allows us to identify some of the features that these new technology provides to web applications. This has allowed us to develop a comprehensive selection of use cases for AJAX technologies, which we omit from this paper for brevity. Without going into detail, and removing features that are already addressed in existing modelling languages, new application features that require support include:

1. Storing data on the client and/or server, both volatile and persistent[3];
2. Allowing automatic user authentication based on cookies[4];
3. Allowing form validation to occur on the server, on the client before submission, or in real-time during form entry;
4. Providing different output formats for resources, including HTML, XML, WML, and Flash, possibly based on the user-agent of the visitor;
5. Providing web services and data feeds, and integration with external services and feeds, both on the server and the client;
6. Preventing the user from corrupting the state of a web application, for example by using browser navigation buttons;
7. Providing more natural user actions such as drag-and-drop, keyboard shortcuts, and interactive maps;
8. Describing visual effects of transitions between application states[5];
9. Having scheduled events on either the client or the server;
10. Allowing web applications to be used offline[6];
11. Distributing functionality between the client and the server, based on client functionality, determined at runtime.

[2]Flash requires additional software to be installed on the client machine, and generally has fewer accessibility features than traditional web sites, though this problem has been a focus in recent versions (Regan 2005).

[3]Possibly through cookies, or offline plugin software such as Google Gears (Google Inc. 2007).

[4]Not all of these new requirements are strictly related to scripting; in this case, existing languages lack modelling support for cookies.

[5]Visual effects can improve the implementation of system metaphors, e.g. fast forward/rewinding through a movie clip metaphor of a series of images.

[6]This could be achieved through plugin software such as Google Gears or the Dojo Offline Toolkit (The Dojo Foundation 2007).

These new features are distributed over both the clients and servers of web applications. Existing languages based solely on replacing the entire client-side DOM on each request are clearly no longer appropriate, as scripting permits modifying the DOM at runtime. We require a more dynamic language, which can be extended to handle these new features.

3 Modelling Concepts

Due to the recent development of these new web applications, there is no existing modelling language for representing these interactive web applications, nor even a comprehensive discussion on what concepts would be required in such a language. To fill this important gap, we propose the following modelling concepts, based as much as possible on existing standards.

3.1 Events

Events are an integral part of new Web applications; indeed, document scripting is directly event-driven, achieved by capturing events created by the web browser (part of the DOM Level 2 events specification (W3C Group 2000); examples include the `onClick` and `onMouseOut` event handlers). Instead of a web application simply responding to navigation through a hypertext, it now has to also contend with user interaction, element manipulation, scheduled events, remote client access, exceptions, and more. The most obvious solution to this problem is to promote events to first-class citizens in the model. A web modelling language should have the ability to define, capture, and create a range of events.

These events could be succinctly captured in terms of Event-Condition-Action (ECA) rules, which are quickly becoming standardised and accepted in the modelling community. ECA rules allow for built-in specification mechanisms through pre- and post-conditions, and can already be visually modelled in UML[7]. There are also emerging standards for rule storage and serialisation through R2ML (Reverse Working Group I1 2006) and Reaction RuleML (Paschke et al. 2007).

3.2 Browser Control

Once simply an extension to a low-level hypertext navigation interface, the user's web browser has become an important part of any web application. The browser is the medium for the interactions between the user, the user's host device, and the remote web application; it also allows web applications to take advantage of the clients' computing power to provide a richer interface. A web modelling language should be able to model concepts specific to browsers, which we have identified to include:

1. Navigation: The very nature of the web allows for users to browse applications out-of-order and jump across navigational links. It is important to not let the user corrupt the state of the web application, or repeat significant events out-of-order[8]. Consideration must also be made to the possibility of one user browsing different parts of the web application through multiple windows.
2. Cookies: Cookies allow the web application to save small amounts of data to the users' device,

[7]see R2ML or the Universal Rule Markup Language URML (Tabet et al. 2000) for more information.

[8]We must also not always restrict the user to only one path of navigation, which would be a serious usability issue.

Layer	Scope	Example Use
Component	elements in the DOM	when a page widget is loaded, move focus to a text box
Request	a single HTTP request	transform XML to WML at end of request
Page	one page can contain many requests over AJAX	close an opened window when the parent page has closed
Session	a visitor has at least one session	delete shopping cart on session timeout
Login	can span multiple sessions, without manual re-authentication	save temporary shopping cart to database on logout
User	represents the user itself	when a user closes an account, delete all their blog entries
Application	represents the entire application	when application is created, create temporary administrator accounts

Table 1: Possible lifecycle layers and their potential use

adding some context/state information to an otherwise stateless protocol. These are often used to implement sessions.

3. Opening windows: On some browsing devices, opening additional windows (either new browser windows or virtual windows inside the application itself) allows for more natural representations of some types of interactions.

4. Scripting: A web modelling language should have some support to identify (and handle) scripting support in the browser; this could allow us to switch users to a more accessible site which does not require browser scripting[9].

5. Plugins: Integration with browser plugins is an important extensibility aspect for web applications.

6. User Agent Identification: By identifying the visitor's user agent, we can make decisions on what content to show[10].

3.3 Lifecycles

One new concept that should be addressed is the ability to identify and define lifecycles in a web application. By defining application lifecycles, we can separate out the layers of a web application into multiple, conceptually-distinct layers. With the opportunity to model events at the beginning and end of lifecycles, we can represent complex interactions with simple structures. We present a small selection of potential lifecycles in Table 1; note how each successive layer builds upon the previous one.

3.4 Users and Security

Along with events, we suggest that users are also an integral part of web applications. The majority of web applications are concerned with the creation and definition of user accounts. Users often have roles or permissions, and often are described in a hierarchy. Implementation of the security of these roles and permissions are vital.

Collaborative software such as Google Docs allow many users to work on a single document at once; these user networking concepts should be supported in a model, along with being able to model the interactions between multiple users. Concise and secure integration of these concepts are vital to the development of secure web applications.

[9]One example of this is a RIA web-mail client redirecting to a text-only site if the browser cannot execute scripts.

[10]For example, switching to a mobile-optimised site for mobile devices.

3.5 Databases

The original hypertext applications were no more than navigation through a complex hypertext of static pages. Since then, integration with databases to provide data-intensive web applications have become the industry norm. This provides new modelling challenges for web application models, in particular:

1. Abstraction to a persistent domain model, which can be independently implemented through a database;

2. The ability to upload files to databases or filesystems, which can be considered a type of database;

3. Distribution of data onto clients, through cookies or local offline storage, e.g. Google Gears (Google Inc. 2007), and synchronisation between client and server databases;

4. Being able to reference multiple databases in an application for scalability purposes.

3.6 Messaging

The most common type of event in web applications is the ability for two components to message each other. The obvious example is the HTTP request/HTML response of traditional web applications, but the range of potential messaging scenarios has grown to include technologies such as sending e-mails, text messages to mobiles, invocation of web services using SOAP requests (W3C Group 2007), RSS feeds, and so on. A web modelling language should have concise support for describing these types of messages, and should be extensible in the future.

3.7 User Interface Modelling

Generally, a web application modelling language should try and keep user interface modelling separate from the conceptual model, to promote separation of concerns. However, web applications now require the ability to model user interface components with event support; for example, events linked to a button on a page may change the state of the web application when the button is activated.

Atterer (2005) notes that separate presentation and navigation models tend to remain strongly related, which raises the question over the suitability of having these models separated, especially for interactive web applications.

3.8 Standards

Innovations on the web are typically proliferated through standards; likewise, a modelling language must also adhere to standards to improve acceptance

by the engineering community. A modelling language should be platform-independent, allowing application deployment on a variety of platforms. It should use existing standards where possible, and should have support for modelling with UML (Object Management Group (OMG) 2005*a*).

Any conceptual modelling language should have support for the Model-Driven Architecture concept (MDA: Object Management Group (OMG) 2003), which is an emerging standard that promises tangible benefits for the quality of software engineering. This standard provides a model with a meta-model – often through the existing Meta-Object Facility model (MOF: Object Management Group (OMG) 2006) – which makes model transformations through existing languages such as QVT and ATL easier (Jouault & Kurtev 2006).

By having the ability to transform models into other models, we can easily develop systems in a range of MDA-compliant models, allowing for automated model translation, simplified integration with diverse platforms and quick round-trip engineering; these address a real industry need for agile development. Languages with such support also gain several advantages, including access to existing software tools to edit and visualise models, and code generation frameworks such as Eclipse's JET (Eclipse Foundation 2007*a*); similarly, a model which can be serialised using standards such as XMI (Object Management Group (OMG) 2005*b*) allows easy integration with other CASE tools. The benefits of the model-driven approach are further explained in Kraus & Koch (2002).

3.9 Verification

An important aspect of software engineering is formal verification, with software verification a well-established field (Holzmann 1997). Thus it is also important to apply the same validation objectives to web applications. The structure of the web introduces different validation concerns, including testing for broken links, page reachability, missing resources, load testing and syntax validation (Bellettini et al. 2005). Many modelling language support custom constraints, often represented as pre- and post-conditions safeguarding the execution of behavioral elements. Verification mechanisms are needed to check those constraints, and interfacing these mechanisms with existing model checkers such as Alloy (Jackson 2006) would be beneficial.

The ability to design applications with asynchronous callbacks adds some new challenges. Some of them, like the detection of deadlocks and and resource starvation, reflect general issues in concurrent systems. Others are related to constraints that can be used to specify the interaction of concurrent processes. For instance, consider the auto completion example used earlier. It is possible that earlier asynchronous requests are overtaken by later requests due to the structure of the web – resulting in the list of suggested strings suddenly becoming less precise. On the modelling level, a constraint must be used stating that the order of responses must be the same as the order of requests.

3.10 Software Support

Finally, a web modelling language should be supported by software tools, either through a dedicated CASE tool, or by extending existing frameworks such as the Eclipse Modeling Framework (EMF: Gerber & Raymond 2003). This allows for proof-of-concept and increases acceptance by the software engineering community.

4 WebML

A major existing academic web modelling language is WebML (Ceri et al. 2000). Derived from AutoWeb (Merialdo et al. 2003), WebML provides a platform-independent conceptual model for data-intensive web sites. It is a topic of active research, with extensions proposed for a range of concepts such as web services (Manolescu et al. 2005), process modelling (Brambilla et al. 2006), exceptions (Brambilla et al. 2005) and context awareness (Ceri et al. 2007). The language is also supported through the commercial CASE tool WebRatio (WebRatio Group 2007*a*).

WebML models are composed of five models, which we will briefly summarise for completeness; for more details, the reader is referred to Ceri et al. (2001):

1. Structural model: an Entity-Relationship (ER) model of the data in the system and their relationships;
2. Derivation model: uses an OQL-like syntax called WebML-OQL to specify derived data in the structural model;
3. Composition model: describes the content units of a site, composed of site views, which display object data;
4. Navigation model: provides contextual links between content units;
5. Presentation model: contains XSL stylesheets to transform XML models into the output platform language.

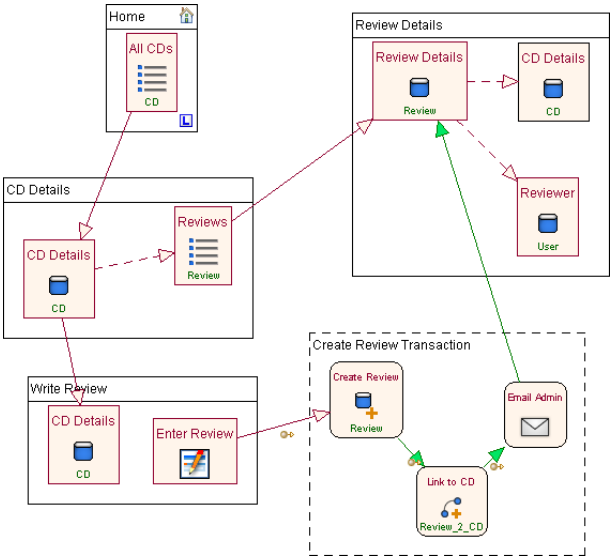

Figure 1: Hypertext model of a simple WebML application

The most important model in WebML is the Hypertext model, which is the combination of the Composition and Navigation models. An excerpt from an example model for a simple online CD store is presented in Figure 1. These models are developed in an iterative waterfall manner.

Note that there appears to be a new AJAX extension under development (WebRatio Group 2007*b*), but details of this extension have not yet been published; it appears to consist of a predefined range of standard AJAX components such as auto-complete fields and drag-and-drop objects.

Feature	WebML	UWE	W2000	OOWS	OOHDM	Araneus
Events	Ok	-	Poor	-	-	-
Browser Control	Poor	-	-	-	-	-
Lifecycles	Poor	Good	Poor	-	-	-
Users	Good	Poor	Poor	Poor	-	-
Security	Ok	Ok	Poor	-	-	-
Databases	Good	Ok	Poor	Poor	Poor	Poor
Messaging	Good	Poor	Ok	-	-	-
UI Modelling	Poor	Ok	Ok	Poor	Ok	Poor
Platform Independent	Excellent	Excellent	Good	Excellent	Good	Ok
Standards	Poor	Excellent	Excellent	Ok	Poor	-
Meta-models	Poor	Excellent	Excellent	Poor	-	-
Model Verification	Ok	Ok	-	-	-	Poor
CASE Tool	Good	Ok	Poor	Ok	-	Ok

Table 2: Feature Comparison of Existing Languages to Model Interactive Web Applications

4.1 Events

Ceri et al. (2002) extends WebML to handle *operations* and *operation chains*, which allows WebML to describe server-side events; this can be seen in the transaction area in Figure 1. Bozzon et al. (2006) further extends the model to allow for distinguishing between client and server operations and objects with the `C` (*Client*) symbol, depicted in Figure 2. Events can only execute explicitly from a request, and cannot spawn additional operations in parallel, except through external web services (Manolescu et al. 2005). Exception support is also proposed in Brambilla et al. (2005) through a variety of event models, however this is not yet supported in WebRatio.

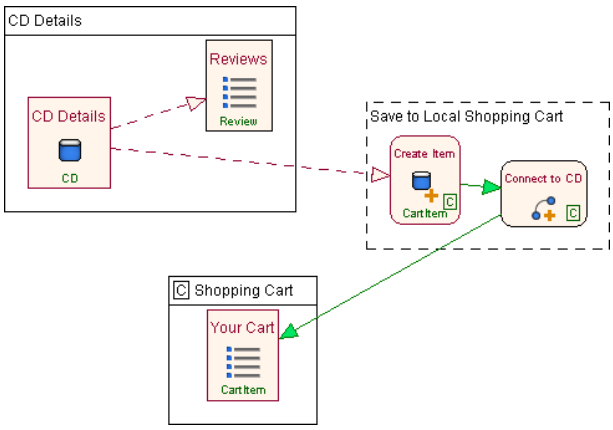

Figure 2: Client/server extensions to WebML

4.2 Browser Control

A WebML model has little support for controlling the browser, with no support for cookies, scripting or identifying the user agent. It allows for scripted components, but the model cannot describe the scripts themselves.

4.3 Lifecycles

The closest thing we can compare lifecycles to is the use of operation chains and navigation links, but these suffer the same issues as event modelling in WebML. An example of this lifecycle support – to delete the users shopping cart at the end of their session – is presented in Figure 3; note that this operation chain only occurs when the user explicitly logs out[11]. WebML does not treat lifecycles as separate conceptual objects, and as such, implementation of lifecycles could add a lot of complexity and redundant information to the model.

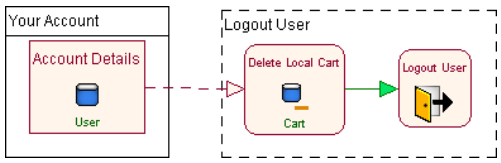

Figure 3: Emulating lifecycle events in WebML

4.4 Users and Security

WebML has inbuilt support for users and groups, but only as members of the structural model. Each user belongs to at least one group, and each group can only access one site view[12]. The model assumes only one user interacts with the model at any one time, so cannot support modelling the interaction between multiple users. The only permissions represented in the model is the ability for groups to view certain pages; additional security permissions must be checked explicitly through operation chains.

4.5 Databases

The use of an ER model in the structural model of a WebML model abstracts the data from the database system; the hypertext models can abstractly access entities in the model. It has limited support for uploading files or handling multiple databases, but should be extensible.

4.6 Messaging

Messaging support is limited to the contextual information passed between hypertext model units; units can also access global parameters. Ceri et al. (2007) adds units to allow the model to access query parameters directly. WebML has excellent support for describing a wide range of web services (Manolescu et al. 2005), and also has good support for sending multi-part e-mails with attachments using its *Power Mail* unit.

4.7 UI Modelling

There is no formal model in WebML for modelling the user interface; the presentation model takes generated XML pages and transforms them using XSL

[11] The *Logout User* unit actually resets the session and redirects the user to the home page; it does not support post-operations.

[12] To switch site views in WebML, the user must explicitly switch into another group.

stylesheets into HTML pages. The CASE tool WebRatio provides a presentation editor which uses a grid-based layout; other presentation attributes have to be composed manually, possibly assisted with external stylesheet design software[13].

4.8 Standards

WebML is platform-independent by design. The generated pages can be in any language, limited only by the implementation of the software tool; WebRatio currently outputs to JSP pages.

The WebML model is generally proprietary, except for the use of ER diagrams for its structural model. It has no support for meta-modelling tools or standards, and the basic WebML model has only recently been implemented as a UML 2.0 profile (Moreno et al. 2006). It lacks comprehensive support for MDA concepts.

4.9 Verification

WebML models have no formal verification process. The WebRatio tool provides a means to search the model for structural and content warnings, but the model itself cannot currently be integrated with any existing model checkers (such as Alloy: Jackson 2006). The model generation process itself has previously been tested with a custom internal prototype in Baresi, Fraternali, Tisi & Morasca (2005).

4.10 Software Support

WebML is actively supported by the development of the CASE tool WebRatio, which provides support for most of the features discussed in this paper. Some extensions, such as exception support, are not yet implemented. The newest version 5.0 extends the popular Eclipse framework (Eclipse Foundation 2007*b*) and is currently in beta, but does not appear to utilise the EMF framework (Gerber & Raymond 2003).

5 UWE

UML-based Web Engineering (UWE) is an extension to UML which aims to provide web application modelling support using standard UML constructs. Like WebML, UWE consists of a number of models; however UWE provides automatic and semi-automatic tools to help the developer progress from model to model, which allows the design process to be more iterative and incremental than WebML's waterfall model.

Most models are generated automatically from previous models, using model transformation techniques such as QVT and ATL; for more information on this process, the reader is referred to Koch (2007). UWE is supported by the CASE tool ArgoUWE (Knapp et al. 2003), based on the open source ArgoUML software (ArgoUML Group 2007). The visual models involved in UWE include:

1. Requirements model: Use cases and activity diagrams are provided which describe the requirements of the system. (Based on WebRE, Cuaresma & Koch 2006, .)

2. Conceptual model: A UML class diagram describing the data structure of the application.

3. Navigation Space model: Constructed from the conceptual model, this specifies *which* data objects can be visited through application navigation.

4. Navigation Structure model: The Navigation Space model is refined manually to specify *how* data objects can be visited through navigation. This adds access element stereotypes such as «index», «guided tour», «query» and «menu».

5. Presentation model: Abstractly specifies the structure of an element in the navigation structure, using stereotyped elements such as «text», «form», «anchor», «image» and so on.

6. Task model: Uses activity diagrams to model actions, and reference models in the presentation model.

7. Deployment model: Describes the structure of the deployed application using a UML deployment diagram.

8. Integration model: This model further refines the navigational structure of the application into a "big picture" model, which is used to generate the platform-specific implementation; for more information, the reader is referred to Koch (2007).

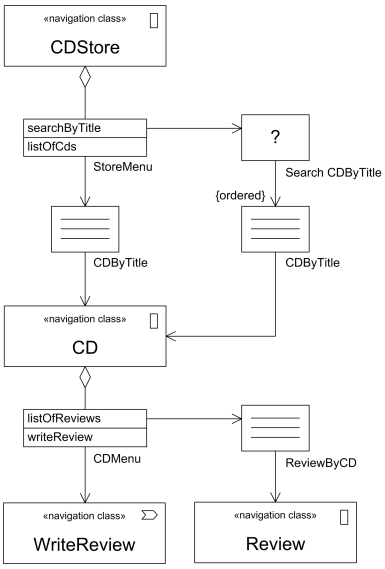

Figure 4: Navigation Structure model of a simple UWE application

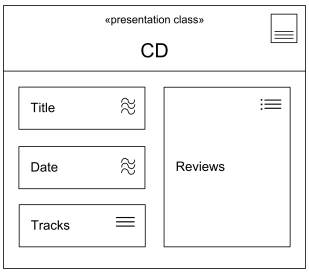

Figure 5: Presentation model of a simple UWE application

By progressing through the incremental models, we can construct a Navigational Structure model for the same simple online CD store application, presented in Figure 4. The access elements in this model can be refined through UML activity diagrams as in

[13]WebRatio comes bundled with the XSL template designer *EasyStyler Presentation Designer* (Brambilla et al. 2006).

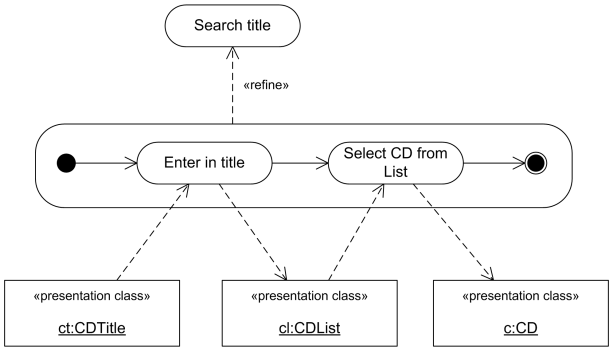

Figure 6: Interaction model of a simple UWE application

the Interaction model presented in Figure 6. These models refer to presentation classes such as the one in Figure 5, which abstractly describe the content and layout of the displayed data object.

As UWE is more recent than WebML, it does not have nearly as many extensions to its functionality; but does have a promising extension to add aspect-oriented modelling (Baumeister et al. 2005). UWE is based on the structure and navigation of web applications, and as such, lacks features such as events that would be required for interactive application modelling.

5.1 Events

UWE has no explicit support for events; as it is modelling only the structure and navigation of web applications, events are ignored and the focus is instead on tasks and activities. The language only considers server-side modelling, so there is no opportunity for client-side activity modelling.

5.2 Browser Control

The client and browser is completely ignored in UWE, with no support for controlling it, or accessing cookies, scripting or plugins.

5.3 Lifecycles

The interaction and task models allow a fairly good representation of interaction lifecycles in UWE, but fall short of specifying comprehensive lifecycle management. These are not real lifecycles, but only interaction structures; any activity happens explicitly. Nevertheless, UWE has promising support for the lifecycle concept.

5.4 Users and Security

Users are treated only as data objects, and as actors for requirements engineering. As such, user permissions and security are ignored, and assumed to be handled in the models themselves. Zhang et al. (2005) suggests modelling access control with aspects; this allows for more expressibility than restricting access based on user permissions, and allows for the construction of fairly complex security, but isn't yet implemented in the CASE tool ArgoUWE.

5.5 Databases

Similar to WebML, the conceptual model of a UWE model abstracts the data from the database system. The navigation and presentation diagrams are based on accessing these objects in the database. It has no support for uploading files, or accessing the filesystem. Since UWE is server-side only, there is no client-side data support.

5.6 Messaging

UWE has no explicit support for concepts such as web services or data feeds, but this would likely be a simple extension. Like WebML, contextual information is automatically passed between navigational links. However it does not support sending e-mails – an important component of web applications.

5.7 UI Modelling

The abstract components used in the presentation model are excellent, allowing the designer to conceptually specify the construction of their page without worrying about implementation details, and simplifies the manual formatting required at the end of the implementation cycle. However, due to a lack of event support, it is not possible to model interaction of the DOM itself.

5.8 Standards

UWE has excellent standards support, as it is based around extending UML using existing standards; all diagrams are UML compliant, and UWE has a MOF-compliant meta-model, which is used extensively in its automated model transformations. UWE's standards support is a natural side effect of following the MDA concept.

5.9 Verification

The CASE tool ArgoUWE currently checks the well-formedness of the model through continuous verification (Knapp et al. 2003). Future work will focus on the integration of UWE with the model checker AGG (Taentzer 2003) to provide model verification; for more information the reader is referred to Koch (2007).

5.10 Software Support

UWE is supported with the CASE tool ArgoUWE, which is a plugin extension of the open source UML modelling tool ArgoUML (ArgoUML Group 2007). It is less refined than WebML's WebRatio tool, and can be difficult to use. This tool currently lacks support for aspect-oriented modelling, but is subject to future work; for more details, the reader is referred to Knapp et al. (2003).

6 Other Languages

There is a wide range of other languages developed by the academic community to model web applications, but WebML and UWE appear to be the most suitable for interactive applications. Most existing languages allow us to specify the data structure (generally through entity-relationship diagrams), the navigational structure, and some aspects of the presentational structure; but generally fail to handle web-specific concepts such as events, scripting or sessions. We will briefly discuss two further languages, W2000 and OOWS, which appear to be the most suitable and popular approaches after WebML and UWE.

6.1 W2000

W2000 (Baresi et al. 2006), derived from the older Hypermedia Design Method (HDM: Garzotto et al. 1993) and developed with principles from UML, is a conceptual web modelling language focused on the development of web applications featuring a business logic layer. Baresi et al. (2000) suggests that web applications add dynamics to the conventional web site dimensions of navigation and information, and argue that operations should be first class citizens – a design goal of W2000.

The use of UML interaction diagrams in its approach of modelling operations is a significant feature, allowing the description of operations and services in the web application. It features a MOF meta-model which provides platform independence, and in theory is easily extended. It allows for an application to be presented in multiple views through presentation models, which can be created from OCL-like business rules (Baresi, Colazzo & Mainetti 2005).

The most significant deficiency for modelling RIAs in W2000 is its lack of client-side support, though it stands to reason this could be achieved through extensions. It provides no explicit support for users or security, except through extending its operational models. A limited lifecycle concept could also be emulated this way.

W2000 is supported by an unreleased prototype Eclipse extension as a CASE tool, but the lack of released software limits W2000's viability as a web application modelling platform.

6.2 OOWS

OOWS (Pastor et al. 2006) is a conceptual web modelling approach focused on identifying users in their systems and their use cases, and the elements within the resulting web pages. Due to its focus as a conceptual (as opposed to a computational) approach, OOWS lacks any support for operational features such as events, web browsers, lifecycles, security, messaging or verification.

Users are first-class citizens in the model, represented in a hierarchy, but the lack of operation support limits the expressibility of their abilities and permissions. The OOWS system supports databases, but only in a directly relationally-mapped way. It provides a basic presentational model through the use of *abstract information units* (AIUs), but the process of translating this to a functional web page appears to require a lot of manual work.

Standards support in OOWS is limited to the use of some UML diagrams, and the OASIS formal language for modelling semantics (Pastor et al. 2006). It is supported with the commercial CASE tool OlivaNova, which aims to be platform-independent over multiple server languages (CARE Technologies 2007).

Clearly OOWS falls short of modelling the majority of the requirements we have identified, and as such, we refer the reader to Pastor et al. (2006) for more information on the language.

6.3 Others

We are aware of many other existing web modelling languages, such as Araneus (Merialdo et al. 2003) and OOHDM (Rossi & Schwabe 2006), but these are generally quite dated and overwhelmingly lack the ability to model the requirements of RIAs. Space constraints in this paper limit us to only mention these in passing, but for comparison purposes, we include our reviews of these other languages in Table 2.

In a similar light, we know that commercial web modelling environments exist, but many of these are proprietary and closed-source; additionally, many lack the same infrastructure to model these requirements. Formal reviews into these commercial tools remains a point of future work.

From a more formal perspective, Schewe (2005) presents web applications ("Web Information Systems") as a triplet of issues: content, navigation and presentation. It allows a system to be described in terms of stories and actions, using user roles for personalisation and security support, and allows defining pre- and post-conditions on actions in the system. By defining the application mathematically, we can formally reason about the system for optimisation and verification, and allows us to detect errors such as dead links. This approach looks promising but currently lacks support for important web concepts such as scripting and sessions, and lacks software or standards support.

6.4 Feature Comparison

We can summarise our major model language reviews into a simple feature matrix, in which we describe each feature with subjective rankings of No Support (indicated with a hyphen) to Excellent. This matrix is presented in Table 2.

7 Discussion

By reviewing our work, we can clearly see that the most urgent features WebML lack are an events model, more control over the web browser, and scripting support. Further research is required to see if the WebML model could handle such a major revision. The lack of meta-modelling support is a serious shortcoming; this lack of support will hinder its integration with other tools and acceptance in the wider community. WebML also requires more development in terms of verification support.

Since it provides less functionality than WebML, UWE appears to require more work to handle our proposed concepts, but initially presents a cleaner model of a web application. It's meta-model support has already proven successful, with the easy integration into the ArgoUML CASE tool. It may handle our extensions more robustly, especially with its ability to successively refine detail in its model. However an extended UWE model may simply become too large, due to its UML roots.

The other languages we have briefly mentioned earlier simply do not have the necessary structure to handle such extensions, and we expect they would require major work or reimplementation to handle these concepts. Generally, the lack of an event model in a web modelling language critically limits the expressibility of the language to describe interactive web applications or RIAs, and modelling the client-side is similarly vital.

7.1 Related Work

Preciado et al. (2005) is similar to our paper, by reviewing existing hypertext and hypermedia languages for their suitability to model RIAs. However, this research is concerned more with hypermedia concepts than hypertext, such as visual continuity and multimedia support; our paper deals with web application concepts, such as session support and document scripting.

Gu et al. (2002) reviews existing languages with respect to a list of functional and informational architecture requirements for web modelling languages. It is concerned only with the visual model, and has no regard for interactive concepts or the distributed web.

The review also misses web-specific concepts such as events and sessions, and mentions emails and users very briefly.

Atterer (2005) reviews the usability of the UWE and OO-H languages, in terms of the usability of the software tool, and the generated websites themselves. It finds that existing languages do not focus on the usability of the individual pages, and current Web Engineering takes place at a much greater level of detail than classical Software Engineering, making the development process more complex and work-intensive.

Schwabe (2001) contains case studies of the implementation of a conference management web application, published as part of IWWOST'01. This study was implemented with many modelling languages, but no overall comparison of the results was made, and such a review would be a useful discussion to have in the future.

Fraternali (1999) is a comprehensive survey of most research projects and commercial tools in 1999, and their ability to model web sites. It covers basic web requirements such as platform independence, code generation and some degree of developing presentational models, but lacks support for all new web concepts, such as sessions, scripting or events.

7.2 Future Work

This paper is the first publication in a doctoral research programme to develop a web modelling language to describe interactive web applications. With our review of existing academic languages, we have identified their respective strengths and weaknesses, which will allow us to decide whether a language extension is appropriate. Another option is to prototype a new language, which would have built-in support for the requirements detailed in Section 3. These two options remain the focus of our future work.

We have focused our review on academic languages only, and likewise, a review of commercial web development tools would be useful. Existing papers that do just this (such as Fraternali 1999) are dated and lack concern for modelling the interactive requirements of RIAs.

One interesting tangential question raised is the appropriate development process for web applications. Languages like WebML advocate an iterative waterfall process, followed by deployment and maintenance; it remains to be seen if this remains appropriate, given the current approach of releasing web applications early and updating regularly, which is strikingly similar to Agile development.

References

ArgoUML Group (2007), 'ArgoUML'.
URL: *http://argouml.tigris.org*

Atterer, R. (2005), Where Web Engineering Tool Support Ends: Building Usable Websites, *in* 'SAC '05: Proceedings of the 2005 ACM symposium on Applied computing', ACM Press, New York, NY, USA, pp. 1684–1688.

Baresi, L., Colazzo, S. & Mainetti, L. (2005), First Experiences on Constraining Consistency and Adaptivity of W2000 Models, *in* 'SAC '05: Proceedings of the 2005 ACM symposium on Applied computing', ACM Press, New York, NY, USA, pp. 1674–1678.

Baresi, L., Colazzo, S., Mainetti, L. & Morasca, S. (2006), W2000: A Modelling Notation for Complex Web Applications, *in* E. Mendes & N. Mosley, eds, 'Web Engineering', Springer, pp. 335–364.

Baresi, L., Fraternali, P., Tisi, M. & Morasca, S. (2005), Towards Model-Driven Testing of a Web Application Generator, *in* 'ICWE', pp. 75–86.

Baresi, L., Garzotto, F. & Paolini, P. (2000), From Web Sites to Web Applications: New Issues for Conceptual Modeling, *in* 'ER '00: Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling', Springer-Verlag, London, UK, pp. 89–100.

Baumeister, H., Knapp, A., Koch, N. & Zhang, G. (2005), Modelling Adaptivity with Aspects, *in* 'ICWE', pp. 406–416.

Bellettini, C., Marchetto, A. & Trentini, A. (2005), TestUml: User-Metrics Driven Web Applications Testing, *in* 'SAC '05: Proceedings of the 2005 ACM symposium on Applied computing', ACM Press, New York, NY, USA, pp. 1694–1698.

Bozzon, A., Comai, S., Fraternali, P. & Carughi, G. T. (2006), Conceptual Modeling and Code Generation for Rich Internet Applications, *in* 'ICWE '06: Proceedings of the 6th international conference on Web engineering', ACM Press, New York, NY, USA, pp. 353–360.

Brambilla, M., Ceri, S., Comai, S. & Tziviskou, C. (2005), Exception handling in workflow-driven Web applications, *in* 'WWW '05: Proceedings of the 14th international conference on World Wide Web', ACM Press, New York, NY, USA, pp. 170–179.

Brambilla, M., Ceri, S., Fraternali, P. & Manolescu, I. (2006), 'Process Modeling in Web Applications', *ACM Trans. Softw. Eng. Methodol.* **15**(4), 360–409.

CARE Technologies (2007), 'OlivaNova: Products Overview'.
URL: *http://www.care-t.com/products/index.asp*

Ceri, S., Daniel, F., Matera, M. & Facca, F. M. (2007), 'Model-driven development of context-aware Web applications', *ACM Trans. Inter. Tech.* **7**(1), 2.

Ceri, S., Fraternali, P. & Bongio, A. (2000), Web Modeling Language (WebML): A Modeling Language for Designing Web Sites, *in* 'Proceedings of the 9th international World Wide Web conference on Computer networks', North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, pp. 137–157.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S. & Matera, M. (2002), *Designing Data-Intensive Web Applications*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ceri, S., Fraternali, P., Matera, M. & Maurino, A. (2001), Designing Multi-Role, Collaborative Web Sites with WebML: a Conference Management System Case Study, *in* 'IWWOST'01 Workshop', Valencia, Spain, pp. 130–152.

Cuaresma, M. J. E. & Koch, N. (2006), Metamodeling the Requirements of Web Systems, *in* 'WEBIST: Proceedings of the Second International Conference on Web Information Systems and Technologies', pp. 310–317.

Eclipse Foundation (2007*a*), 'Eclipse Modeling: Java Emitter Templates'.
URL: *http://www.eclipse.org/emft/projects/jet/*

Eclipse Foundation (2007*b*), 'Eclipse.org Home'.
URL: *http://www.eclipse.org*

Fraternali, P. (1999), 'Tools and approaches for developing data-intensive web applications: a survey', *ACM Comput. Surv.* **31**(3), 227–263.

Garrett, J. J. (2005), Ajax: A New Approach to Web Applications, Technical report.
URL: *http://www.adaptivepath.com/publications/essays/archives/000385.php/*

Garzotto, F., Paolini, P. & Schwabe, D. (1993), 'HDM – A Model-Based Approach to Hypertext Application Design', *ACM Trans. Inf. Syst.* **11**(1), 1–26.

Gerber, A. & Raymond, K. (2003), MOF to EMF: There and Back Again, *in* 'eclipse '03: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange', ACM Press, New York, NY, USA, pp. 60–64.

Google Inc. (2006), 'Google Docs'.
URL: *http://docs.google.com*

Google Inc. (2007), 'Google Gears'.
URL: *http://gears.google.com*

Gu, A., Henderson-Sellers, B. & Lowe, D. (2002), Web Modelling Languages: The Gap Between Requirements and Current Exemplars, *in* 'Proceedings Of The Eighth Australian World Wide Web Conference'.

Holzmann, G. J. (1997), 'The Model Checker SPIN', *Software Engineering* **23**(5), 279–295.

Jackson, D. (2006), *Software Abstractions: Logic, Language, and Analysis*, Cambridge, Mass.

Jouault, F. & Kurtev, I. (2006), On the architectural alignment of ATL and QVT, *in* 'SAC '06: Proceedings of the 2006 ACM symposium on Applied computing', ACM Press, New York, NY, USA, pp. 1188–1195.

Knapp, A., Koch, N., Moser, F. & Zhang, G. (2003), ArgoUWE: A Case Tool for Web Applications, *in* 'First Int. Workshop on Engineering Methods to Support Information Systems Evolution (EMSISE 2003)'.

Koch, N. (2007), 'Classification of Model Transformation Techniques Used in UML-based Web Engineering', *Software, IET* **1**(3), 98–111.

Koch, N. & Kraus, A. (2002), The Expressive Power of UML-based Web Engineering, *in* 'IWWOST'02', pp. 105–119.

Kraus, A. & Koch, N. (2002), Generation of Web Applications from UML Models using an XML Publishing Framework, *in* 'IDPT-2002: Integrated Design and Process Technology'.

Manolescu, I., Brambilla, M., Ceri, S., Comai, S. & Fraternali, P. (2005), 'Model-driven design and deployment of service-enabled web applications', *ACM Trans. Inter. Tech.* **5**(3), 439–479.

Merialdo, P., Atzeni, P. & Mecca, G. (2003), 'Design and Development of Data-Intensive Web Sites: The Araneus Approach', *ACM Trans. Inter. Tech.* **3**(1), 49–92.

Moreno, N., Fraternalli, P. & Vallecillo, A. (2006), A UML 2.0 profile for WebML modeling, *in* 'ICWE '06: Workshop proceedings of the sixth international conference on Web engineering', ACM Press, New York, NY, USA.

Object Management Group (OMG) (2003), Model-Driven Architecture Guide, v1.0.1, Technical report.
URL: *http://www.omg.org/cgi-bin/doc?omg/03-06-01*

Object Management Group (OMG) (2005*a*), Unified Modeling Language (UML): Superstructure Specification, v2.0, Technical report.
URL: *http://www.omg.org/cgi-bin/doc?formal/05-07-04*

Object Management Group (OMG) (2005*b*), XML Metadata Interchange (XMI), v2.1, Technical report.
URL: *http://www.omg.org/technology/documents/formal/xmi.htm*

Object Management Group (OMG) (2006), Meta Object Facility (MOF) Core Specification, v2.0, Technical report.
URL: *http://www.omg.org/cgi-bin/doc?formal/2006-01-01*

Paschke, A., Kozlenkov, A., Boley, H., Tabet, S., Kifer, M. & Dean, M. (2007), Reaction RuleML, Technical report.
URL: *http://ibis.in.tum.de/research/ReactionRuleML/*

Pastor, O., Fons, J., Pelechano, V. & Abrahão, S. (2006), Conceptual Modelling of Web Applications: The OOWS Approach, *in* E. Mendes & N. Mosley, eds, 'Web Engineering', Springer, pp. 277–302.

Preciado, J. C., Linaje, M., Sanchez, F. & Comai, S. (2005), Necessity of Methodologies to Model Rich Internet Applications, *in* 'WSE '05: Proceedings of the Seventh IEEE International Symposium on Web Site Evolution', IEEE Computer Society, Washington, DC, USA, pp. 7–13.

Regan, B. (2005), Best Practices for Accessible Flash Design, Technical report.
URL: *http://www.adobe.com/resources/accessibility/best_practices/best_practices_acc_flash.pdf*

Rewerse Working Group I1 (2006), R2ML – The REWERSE I1 Rule Markup Language, Technical report.
URL: *http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=R2ML*

Rossi, G. & Schwabe, D. (2006), Model-Based Web Application Development, *in* E. Mendes & N. Mosley, eds, 'Web Engineering', Springer, pp. 303–333.

Schewe, K.-D. (2005), The Challenges in Web Information Systems Development in 15 Pictures (Invited Talk), *in* 'ISTA', pp. 204–215.

Schwabe, D., ed. (2001), *First International Workshop on Web-Oriented Software Technology.*

Tabet, S., Bhogaraju, P. & Ash, D. (2000), Universal Rule Markup Language, Technical report.
URL: *http://home.comcast.net/ stabet/urml.html*

Taentzer, G. (2003), AGG: A Graph Transformation Environment for Modeling and Validation of Software, *in* 'Proc. Tool Exhibition at Formal Methods 2003'.

The Dojo Foundation (2007), 'The Dojo Toolkit'.
URL: *http://dojotoolkit.org/*

van den Broecke, J. (2002), Pushlets White Paper, Technical report, Just Object B.V.
URL: *http://www.pushlets.com/doc/whitepaper-all.html*

W3C Group (2000), Document Object Model (DOM) Level 2 Events Specification, Technical report, W3C Recommendation 13 November 2000.
URL: *http://www.w3.org/TR/DOM-Level-2-Events/*

W3C Group (2004), Document Object Model (DOM) Level 3 Core Specification, Technical report, W3C Recommendation 07 April 2004.
URL: *http://www.w3.org/TR/DOM-Level-3-Core/*

W3C Group (2007), Simple Object Access Protocol (SOAP) Version 1.2, Technical report, W3C Recommendation 27 April 2007.
URL: *http://www.w3.org/TR/soap12-part1/*

WebRatio Group (2007*a*), 'WebRatio'.
URL: *http://www.webratio.com*

WebRatio Group (2007*b*), 'WebRatio AJAX Extension'.
URL: *http://www.webratio.com/WebRatio-AJAX.do*

Zhang, G., Baumeister, H., Koch, N. & Knapp, A. (2005), Aspect-Oriented Modeling of Access Control in Web Applications, *in* '6th Int. Workshop Aspect Oriented Modeling (AOM'05)', Chicago, USA.