Ph.D. Confirmation Report

A Modelling Language for Interactive Web Applications

Jevon Wright

Supervisor: Dr Jens Dietrich

Institute of Information Sciences & Technology
Massey University

March 7, 2008

Abstract

Web applications are quickly becoming a significant aspect of software development, especially with the recent development of interactive, client-side web applications, frequently identified as part of *Web 2.0*. Consequently, these new web technologies are increasing web application complexity and adding new concepts to an already complicated environment. The use of formal modelling approaches is frequently expected to improve the reliability, usability and security of web applications, as well as reducing development costs, promoting openness, standards and platform-independence, improving maintainability, and advancing the field of code generation.

Despite the existence of a diverse range of web application modelling languages, no approach has become widely accepted in industry. Some researchers suggest that the development of new modelling approaches and development processes would address these problems. This Ph.D. research aims to develop a platform-independent modelling language, within the conceptual framework of Model Driven Architecture (MDA) and the Meta Object Facility (MOF) standard, which will reunite the conceptual gap between these new technologies and their implementation. This outcome will be empirically validated with a proof-of-concept CASE tool, and compared with existing methods to discuss the expressibility and performance of our contributions.

In this confirmation report, we will discuss the state of model-driven web development and its implications on industry. We will introduce the research aims of this Ph.D., along with our expected contributions and how we are planning to achieve them. We will conclude with a summary of our progress and discuss our future direction.

This is a doctoral confirmation report submitted in part to fulfill the requirements of progressing to full registration in the Ph.D. programme in Computer Science at Massey University, Palmerston North, New Zealand. This Ph.D. has been supervised

by Dr Jens Dietrich at the Institute of Information Sciences and Technology, soon to become the School of Engineering and Advanced Technology. My research into this Ph.D. was supported by the Massey University Doctoral Scholarship.

Contents

Abstract **i**

1 Introduction **1**

2 Research **3**
- 2.1 Research Questions . . . 3
- 2.2 Major Contributions . . . 4
- 2.3 Research Method . . . 5

3 State of the Art **7**
- 3.1 Existing Languages . . . 7
  - 3.1.1 WebML . . . 7
  - 3.1.2 UWE . . . 8
  - 3.1.3 UML Extensions . . . 9
  - 3.1.4 Older Languages . . . 10
  - 3.1.5 Formal Models . . . 10
  - 3.1.6 Commercial Approaches . . . 12
- 3.2 Frameworks . . . 12
- 3.3 Other Literature . . . 13
- 3.4 New Functionality . . . 14
- 3.5 Our Critical Review . . . 15
- 3.6 Summary . . . 16

4 Enabling Technologies **17**
- 4.1 Models . . . 17
- 4.2 Metamodels . . . 18
- 4.3 Metamodel Support . . . 20

4.4 Transformations and Code Generation 21
4.5 Verification 22

5 Progress **23**
5.1 Publications 23
5.2 Future Direction 24
5.3 Final Thesis 26

A Glossary **27**

B Interactive Web Applications **29**

C Web Application Requirements **31**

D Final Thesis Structure **33**

List of Figures

2.1 Iterative Development Lifecycle 6

3.1 Hypertext Model of a simple WebML application 8
3.2 Navigation Structure Model of a simple UWE application 9
3.3 Overview Class Diagram of Conallen's UML extension 11
3.4 Simultaneously editing a UML model and it's code in Apollo 12

4.1 Programming languages in MOF 19
4.2 Levels of abstraction in MDA 19
4.3 The conceptual gap in web development 20
4.4 The GMF Framework Wizard in Eclipse 21

Chapter 1

Introduction

The invention of the Internet has ushered in a new era of connectivity and information availability. From anywhere in the world, any person with a working Internet connection and a browsing device – whether it be Personal Computer, TV set, mobile phone or PDA – can instantly browse the information on the Internet. This has been augmented with the rise of interactive web applications, which provide rich interfaces for anybody to interact with a diverse range of services, or publish their own content. Desktop software applications[1] are now becoming increasingly superseded with web-based applications, in part to the lower overall Total Cost of Ownership [1] of centralised web applications.

Software application development has arguably been improved significantly by the development of standard modelling languages such as UML [2], providing a clear, consistent way to document and model software systems. As web applications can be considered a form of software, it is clear that web application development should have a similar level of support [3].

Despite claims that the web development industry is experiencing a quality and methodology crisis [4, 5, 6], surveys [7] show that talk of such a crisis is unfounded, with most developers following structured in-house methods. However, the use of formal methodologies is exceedingly rare [7, 8, 4], despite industry expectation that these methods improve the reliability, usability, security [7] and maintainability [9] of web applications, and decrease development costs and risks [9].

Web development tends to focus solely on the low-level implementation [9]

[1]Refer to Glossary in Appendix A for definitions.

using a diverse range of technologies [3], but at the same time must be accessible to a diverse range of platforms and devices [3]. It is this *conceptual gap* between application requirements and the implementation that needs to be addressed. Some researchers argue that some level of abstraction in the form of models and tools will bridge this gap [10, 5], and consequently many approaches have been proposed. These approaches are either extensions to UML [2] aiming to capitalise on its existing support and standards [11, 12], or the development of entirely new domain-specific models [13, 14] which instead aim on expressibility and implementation support.

Despite these attempts, existing solutions are woefully inadequate for modelling web applications [3]. In particular, solutions lack support for interactivity; cannot support dynamic content generation; ignore existing standards and frequently have no implementation support [8]. It is clear that since existing solutions cannot model web applications, industry will hesitate to use them.

This situation is made worse by the recent rise of new types of user-oriented Rich Internet Applications (RIAs) [15]. These applications place some computational requirements on the client-side which cannot be expressed by any existing web application model [16]. Along with the unique requirements of web applications [9], there is now a significant gap between any existing methodology and the implementation of web applications.

It is the task of modelling this conceptual gap between requirements and implementation details that is the focus of this Ph.D. project.

Chapter 2

Research

This absence of a suitable modelling language or methodology, along with the increasing complexities offered today by user-oriented interactivity in rich web applications, leads us to our thesis statement:

> *The development of a new interactive web application metamodel will allow us to solve the challenges faced in Web Engineering; in particular, improving the reliability, usability and security of interactive web applications, and simplifying the development process in industry.*

2.1 Research Questions

This thesis statement is naturally broken down into five key questions, which we aim to address in the course of this Ph.D.:

1. What are the challenges that interactive web applications provide?
2. What issues exist with existing web application modelling languages?
3. Can we develop a modelling language to handle these challenges?
4. How do we prove that such a modelling language provides web application developers with greater reliability, usability and security?
5. Similarly, how do we prove that such a modelling language improves the development process, in terms of speed, simplicity and consistency?

2.2 Major Contributions

Our answers to these key questions will provide the knowledge on which to develop such a web application metamodel, leading us to the major expected contributions of our work:

1. We will identify the range of features and requirements of web applications, and condense them into some core requirements (Appendix C).
2. We will identify the existing approaches and models available, and investigate to what extent they can describe these new features, as well as document their extensibility.
3. We will propose a *benchmark web application* to allow the evaluation and comparison between different web engineering approaches.
4. We will propose an interactive web application modelling language, which;
   (a) Is expressive enough to describe the interactive, asynchronous update of web content in web applications; in particular, support for AJAX-based application designs and Web 2.0 user collaboration.
   (b) Is expressive enough to describe the diverse range of possible events on the web, such as client requests, session lifecycles and scheduled server events.
   (c) Is expressive enough to describe the majority of existing interactive web applications currently on the web.
   (d) Is suitable for round-tripping in development, with the ability to generate significant executable code from the model, and vice versa.
   (e) Is platform-independent and standards-compliant, using existing meta-models where appropriate.
   (f) Fits in with the Model Driven Architecture [17] concept and complies with the Meta-Object Facility [18] approach.
   (g) Can be used for formal verification purposes, with an interface to an existing model checker such as Alloy [19].
   (h) Is open-source and promoted in industry, with a prototype proof-of-concept CASE tool implementation.

5. We will evaluate our modelling language empirically with existing approaches, using our benchmark web application, and discuss the suitability and effectiveness of our approach.

2.3 Research Method

Existing modelling approaches have been attempted in the past, ranging from domain-specific languages [13] to extensions of UML [11], but these approaches have generally lacked independent support[1] in industry [7], and more importantly, expressibility. We think this is due to some key issues:

1. Commonly, models are aimed at an older version of web technology, and cannot express client-side user interactivity.
2. Most models are not easily extensible, which prevents developers from adding support for new technologies.
3. Many models have no CASE tool support or are proprietary, which hinders industry adoption, making it unclear if such a model is expressive enough, and preventing model extensibility.

One other major issue we expect to encounter is the speed of web technology development; it is more than likely that by the end of this project there will be some significant new concepts in web technology. In particular, upcoming browser platforms such as Firefox 3 offers revolutionary new technologies such as offline browsing and better graphics support [21]; the upcoming HTML 5 standard offers offline support, new content elements, a new event source element, and significant changes to the client API [22].

Both of these proposals did not exist at the start of this Ph.D., and similarly we expect the development of other new technologies within the next 24 months, which would create significant risk in such a research project. To handle this risk, we are undertaking research in a iterative[2] fashion as illustrated in Figure 2.1 (adapted from [24]). A timetable of our iterations is provided in Table 5.1.

[1]Notwithstanding industry examples which are under direct influence from the authors themselves [20].

[2]A similar option, often confused with iterative development, is incremental development; however, this approach is too agile for our project, which requires dedicated research. See [23] for a discussion of these two approaches and their respective benefits.

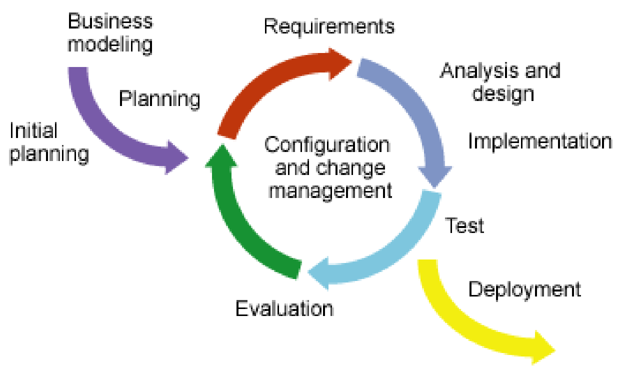


Figure 2.1: Iterative Development Lifecycle

Iterative development is known to focus development and emphasise the development of deliverable artefacts. Compared with conventional approaches, this reduces project risk and increases the participation of stakeholders [25]. This approach also allows us to handle changing requirements, resolving the challenges imposed by the volatile nature of the web. The focus on deliverable artefacts will also implicitly provide us with sufficient documentation, experience in extendibility, a detailed and robust current model, and a functional proof-of-concept CASE tool. Researchers argue that such a focus on usable, deliverable tools and processes is vital to transform industry [26, 27].

Chapter 3

State of the Art

By simplifying a system in an abstract form (a model), new knowledge about the system can be discovered, and in some cases, allows simulation of the original system [28]. In regards to software development, the practical effects of having an abstract model of a complicated systems allows stakeholders to collaborate and communicate more effectively. A common modelling platform to use in this situation is UML [2], which can also be extended to model any domain through its extensibility functionality [29]. Using a model increases the conceptual understanding of a system; and potentially, provides the ability to share and integrate models using interchange formats such as XMI [30].

3.1 Existing Languages

The benefits of using models combined with the apparent methodology crisis has sparked the development of a significant number of modelling approaches. We have found that they tend to either be abandoned, poorly implemented, and lack the features required to model industry-standard web applications.

3.1.1 WebML

As the most-researched and commercialised academic web application language, WebML [13] provides for the description of web applications through five models, as summarised in [16]. One of these models is presented in Figure 3.1. These models are entirely server-side, and as a consequence, it is difficult to describe client-side

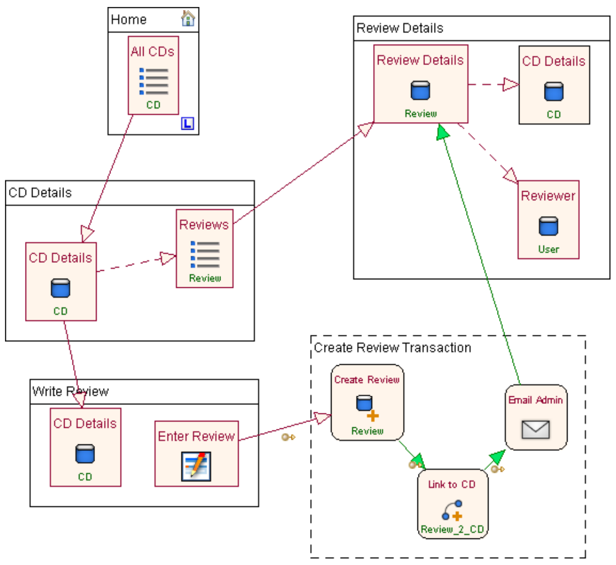


Figure 3.1: Hypertext Model of a simple WebML application

functionality such as client-side events or offline toolkits. WebML also has poor support for web concepts, being unable to directly represent sessions, cookies[1], or scripting.

One upcoming client-side extension [32] aims to provide client-side objects through the use of additional stereotypes in the Hypertext Model. Another upcoming AJAX extension [33] aims to address the use of interactive components, but this is implemented as a new library, rather than solving the main challenges in distributed components. User support is not built into the model, possibly compromising security. The model and CASE tool is closed source suggesting extensibility will be difficult.

3.1.2 UWE

UML-based Web Engineering [11] is an extension to UML which aims to provide a web-centric modelling approach to web applications. UWE consists of more individual models than WebML, but provides automatic and semi-automatic tools[2], implemented in it's ArgoUML-based CASE tool ArgoUWE [35], to simplify the

[1]Through an extension, one can access *global variables* which could be considered part of the session or cookie [31].
[2]Using QVT model transformation technology [34].

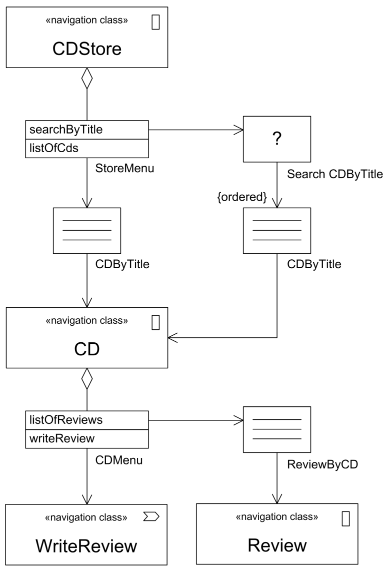


Figure 3.2: Navigation Structure Model of a simple UWE application

model development process. One of these models is presented in Figure 3.2, based on [11]. All of the models are directly UML models or simple extensions based on stereotypes, as summarised in [16].

Like WebML, these models are entirely server-side, and cannot describe client-side functionality. Users and security are also ignored, assuming the developer will deal with these directly in the application logic. UWE has less support for web concepts than WebML, also ignoring e-mails and browser identification. However, it's standards-based approach and automatic model transformation processes are promising in regards to future extensibility.

3.1.3 UML Extensions

Conallen's UML extension [36] aims to describe web applications through the use of four major UML models – the use case diagram, activity diagrams, sequence diagrams and extended class diagrams. The resulting model is fairly complex (see Figure 3.3, based on [36]) due to its UML heritage. It is apparently supported by the Rational CASE tool, but this implementation could not be found. It is very platform-dependent, and ignores the client side completely. It lacks security and database modelling[3] concepts, as well as browser concepts and even sending e-mails.

The extension initially appears promising and is frequently cited, but severely lacks the expressibility required to describe interactive web applications. It is possible that this could be addressed by creating new stereotypes and models, but due to it's complexity this work will only remain a source of inspiration.

Web Unified Modelling Language (WUML) [12] aims to address both web application design and customisation concerns in one UML extension, but neglects aspects related to web applications, and instead focuses on the modelling of context information in UML and applying it to core UML concepts such as packages.

The UML profile for **Schedulability, Performance and Time Specification** (SPT) [37] is an official OMG profile to add events, actions and concurrency to UML packages. Whilst this could be a useful source of inspiration towards developing an actual model, the SPT extension is focused too much on the issues of concurrency, shared resources, resource management and performance analysis, and consequently it may prove beneficial to not include the entire profile in a modelling language.

3.1.4 Older Languages

Earlier approaches to model web applications have also been investigated, such as Araneus [38], HDM [39], Autoweb [40] and OOHDM [14], but these are dated and cannot be used in any practical context.

3.1.5 Formal Models

Rather than providing a visual model, work has been done on describing web applications from a more formal perspective, with roots in algebra and graph theory. Schewe [41] describes **Web Information Systems** as a triplet of issues: content, navigation and presentation, and goes on to formally describe the functionality of each triplet in the system. However, the approach lacks the same support for web content as older languages, and cannot handle interactive functionality.

The clear benefit of using a formal model for describing web applications is that the model also allows us to formally verify the functionality of the system as described; however, these are much more difficult for developers to understand, and

[3]Except through referencing DCOM-style interfaces.

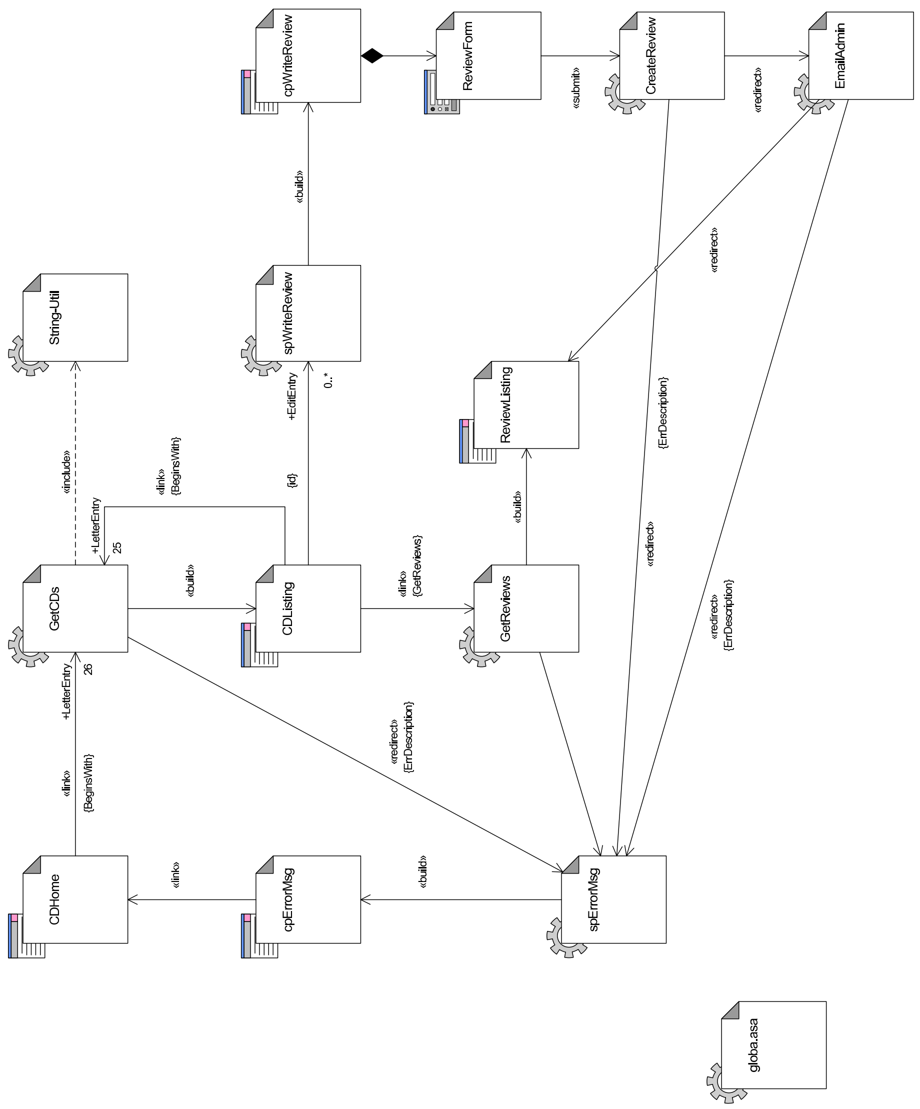


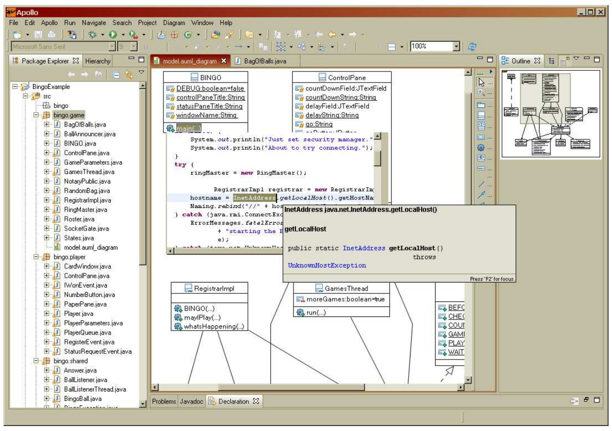
Figure 3.3: Overview Class Diagram of Conallen's UML extension

without a usable CASE tool or real-world implementation, will never be accepted by industry [42].

3.1.6 Commercial Approaches

Web application development software tend to all follow similar approaches; they act as code generators based on textual source code, and do not use an actual model. This may be due to their history as IDEs and not model-based tools. Software such as Adobe Dreamweaver, NetBeans Visual Web Pack, Microsoft Visual Studio and Zend Studio fall into this category.

CASE tools approach the problem from a different direction; Apollo[4] allows users to edit code and the UML model at the same time [43], as shown in Figure 3.4. Apollo lacks web support, but the approach is promising and inspiring for future development, as it allows the model to stay synchronised with the implementation, an identified major problem with existing tools [44].



```
        System.out.println("Just set security manager."
        System.out.println("About to try connecting.");
    }
    try {
        ringMaster = new RingMaster();

            RegistrarImpl registrar = new RegistrarIm
        hostname = InetAddress.getLocalHost().getHostNa
        Naming.rebind("//" + hos
    } catch (java.rmi.ConnectExc
        ErrorMessages.fatalError
            + "starting the B
            e);
```

```
InetAddress java.net.InetAddress.getLocalHost()

getLocalHost

public static InetAddress getLocalHost()
                          throws
UnknownHostException

Press 'F2' for focus.
```

Figure 3.4: Simultaneously editing a UML model and it's code in Apollo

3.2 Frameworks

A practical solution to answering the conceptual gap in web development is the use of frameworks, to apply standard solutions to common problems. A framework is "a reusable, 'semi-complete' application that can be specialised to produce custom

[4]A successor to the popular Poiseidon UML 1.4 editor.

applications" [45]. Popular frameworks in industry include Ruby on Rails, Symfony for PHP, Struts for JSP, Seaside for Smalltalk, Spring for Java, and Spring.NET for ASP.Net. By taking a very practical approach, frameworks are popular in industry due to rapid development time and stability.

Frameworks could be seen as extensions to existing programming languages, which only provide boiler-plate code specific to a particular problem domain. However, depending on the complexity of the framework, the lines between mere boiler-plate code and full-fledged language instantiation becomes blurred; for example, Symfony provides a custom database interface which is radically different from PHP and could be considered a new language. The only issue surrounding the use of frameworks is in the difficulty in extending the framework to different problem domains; in this case, more complexity must be built into the framework, or a new approach must be developed entirely.

3.3 Other Literature

Web application modelling has been extensively researched in the past, with [8] an excellent survey of the fundamental problems with existing approaches. Other reviews concern themselves with evaluating the functional requirements of languages [46, 47], and consistently find that existing languages are inadequate due to deficiencies in expressibility, usability or implementation support. Some reviews are too dated for serious consideration [48, 49].

Research in describing *interactive* web applications can be derived from the work done in hypermedia applications [26, 15], notwithstanding hypermedia's preference to proprietary user interfaces such as CD-ROM applications [27]. Web applications can be considered low-level hypermedia applications [7]; however the requirements of web applications adds unique challenges that existing surveys tend to neglect, both in terms of technical issues and usability issues [9].

Discussions on the suitability of model-driven approaches are found widely in existing literature, with the best overviews by [30] and [50]. [9] discusses the use of MDA and it's natural advantages to address the unique problems introduced by web applications, such as improving usability and future maintenance.

3.4 New Functionality

Interactive web applications are primarily focused on the AJAX concept [51], which allows for a web application to spawn background network requests asynchronously, and access and modify the existing HTML DOM correspondingly with the use of client-side scripting. This allows the application to minimise the network traffic and latency and improve efficiency, usability and responsiveness.

Key innovations in the area of web technologies include:

1. **Client-side Scripting:** The use of Javascript/ECMAScript [52] on the client side allows for the browser to be more responsive to user interactions, for example highlighting the current selection, or checking a form before it is submitted to the remote server.

2. **Document Object Model:** Client-side scripting can access and modify the DOM [53], a tree-based representation of the current web page. This allows the browser to dynamically create new images, dialogs and content based on user interaction, improving usability and responsiveness.

3. **Web Services:** By providing a publicly-accessible interface[5] to a remote network service, external applications can access these resources and combine them in new and innovative ways, also known as **mashups** [54]. These services can be described and accessed using standards such as SOAP [55] and XML-RPC [56].

4. **Data Feeds:** Web applications can provide a live or delayed data feed to their resources through the use of RSS [57], allowing clients to regularly retrieve new data. This is commonly used by newsreaders to collate many diverse news sources together, and is a core feature of Yahoo Pipes [58].

5. **Pushlets:** The Internet is generally a request-oriented medium; by keeping a connection persistently open through the use of AJAX, and frequently parsing this stream for new events, it is possible to simulate an event-based open connection on the client side [59][6].

[5]Also known as an API.

[6]HTML 5 has proposed a new element called *event-source*, which will provide similar functionality.

6. **Offline Toolkits:** External plugins[7] can allow the web application to store data and resources (images and scripts) on the client-side, allowing the application to be used offline, which can be useful for unreliable Internet connections. The upcoming Firefox 3 [21] and HTML 5 [22] releases are also promoting this functionality.

One important concern of AJAX use is that it can be overused – web applications compressed into one single scripted page can take a long time to load, and can reduce accessibility[8]. Similarly, a web application which submits an overwhelming number of AJAX requests can create substantial network traffic which will actually reduce efficiency and responsiveness, and possibly crash the remote server.

3.5 Our Critical Review

By identifying a selection of key interactive web applications (Appendix B), we used our knowledge of these applications to construct a list of 62 use cases, all of which are possible with technology available today (Appendix C). These use cases are also the requirements for any modelling language to address, and were grouped into key areas (Table 3.1, adapted from [16]) and evaluated against existing approaches.

Our review found that not only do existing approaches neglect support for interactive web applications, the majority of these languages also have poor support for the very web concepts they should be supporting already, such as sessions, events, browsers, e-mails, users and security. This critical review was the basis of our first paper [16] and the results are summarised according to a subjective ranking scale (Table 3.2) as a feature matrix in Table 3.1.

Some common themes we can note is that existing modelling languages have no direct support for events or interactive functionality, and instead provides this functionality through component libraries that are difficult or impossible to extend. They tend to ignore the web browser and common concepts such as users and security, which are fundamental concepts in web development. There is generally no support for model verification or CASE tool support, and some approaches do not provide a real-world implementation. We believe that a modelling language for the web will require software support, not only as a proof of concept, but also to encourage discussion and industry support.

[7]Such as the Dojo plugin [60] and Google Gears [61].

[8]The accessibility and usability of RIAs are two important issues which need to be resolved [62].

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	-	-	-
Database Support	Good	Ok	Poor	Poor	Poor	Poor
Messaging	Good	Poor	Ok	-	-	-
UI Modelling	Poor	Ok	Ok	Poor	Ok	Poor
Platform Independence	Excellent	Excellent	Good	Excellent	Good	Ok
Standards Support	Poor	Excellent	Excellent	Ok	Poor	-
Use of Metamodels	Poor	Excellent	Excellent	Poor	-	-
Verification	Ok	Ok	-	-	-	Poor
CASE Tool Support	Good	Ok	Poor	Ok	-	Ok

Table 3.1: Feature Comparison of Existing Modelling Languages

Rating	Concept Support
-	No support at all
Poor	Very limited support, difficult to implement
Ok	Some support, some aspects cannot be implemented
Good	Most aspects can be implemented with ease, generally complete
Excellent	Ideal implementation of the concept

Table 3.2: Feature Comparison Measurements

3.6 Summary

We can safely say that based on our review, along with the other reviews discussing some of the more fundamental features of web applications, no existing language fully satisfies our requirements. There is an obvious conceptual gap between the high-level web system concepts and the low-level technologies required to support its implementation. This gap is highlighted by the arrival of new technologies, adding additional complexity that needs to be simplified. Along with a CASE tool implementation, a modelling language for interactive web applications is clearly required.

Chapter 4

Enabling Technologies

Our desire to develop a new modelling language is centered around the recent arrival of key technologies, which we expect will simplify the model development process significantly.

4.1 Models

A model's abstract simplification of reality implies that it cannot fully capture all of the semantic details of an actual system. Consequently, models in industry tend to focus on only the documentation and initial design of a system; many models are abandoned early due to insychronity between the model and the implementation [44] without the use of a sophisticated CASE tool [43].

To solve a problem as complex as modelling web applications, one may instead define a Domain Specific Language (DSL) which allows the architect to define solutions at the level of abstraction of the problem domain [29], allowing domain experts to understand them easily. The reduced abstraction improves productivity compared to using general-purpose modelling languages such as UML, which is more commonly used for documentation and high-level system design purposes.

UML [2] is the standard software engineering modelling language, but its general purpose approach directly impacts on its suitability as a domain-specific modelling language. Common web concepts such as sessions, timed events and e-mails are difficult to describe in classic UML models, and extension through UML 2.0 profiles [63] are required. Existing extensions such as UWE provide some web-based extensions to UML but as identified earlier, these generally ignore the same web concepts.

The OMG recommends extending UML through the use of three key extension mechanisms; stereotypes, tagged values, and constraints [29]. In practice, only stereotypes are supported by existing CASE tools [64], and constraints are poorly supported. This extension approach can only add new subclasses and constraints; you cannot remove any existing UML notations or create new operations without creating an entirely new UML derivative language [29]. Some developers note that DSLs provide more precision and less complexity, and recommend extending UML only if the majority of your concepts easily map onto existing UML concepts [29].

4.2 Metamodels

One approach which is enjoying a lot of attention is the Meta-Object Facility (MOF) [18] approach. The MOF approach is based on four layers [50] from M0 (representing real-world things) to M3 (representing the meta-metamodel, the abstract frameworks and languages describing the metamodel). A practical example of these four layers is presented in Figure 4.1, adapted from [50], as programming languages are themselves a model of a real system, with well-defined metamodels (the programming language grammar) and meta-metamodels (the use of EBNF textual syntax [50]). By creating your own domain-specific models, and describing them in a standard way, architects gain both the expressibility and ease-of-use benefits of a custom language, and the portability and tool support of a standardised model architecture.

A similar concept, which is also attracting attention, is the Model-Driven Architecture (MDA) [17] approach. MDA attempts to simplify the conceptual load of multiple models by proposing three levels of abstraction; the *Computation Independent Model* (CIM), the *Platform-Independent Model* (PIM) and the *Platform Specific Model* (PSM), as illustrated in Figure 4.2 (adapted from [65]). This allows us to understand the interactions between multiple models, as technically every metamodel (in M2) can also be a model (M1) and a real-world thing (M0) [50]. The three layers are less clearly defined than in MOF, as a model can belong to any layer, depending on context. They are also less strict, as note how a PIM can be realised by another PIM.

To understand these two initially separate approaches, we can combine these into

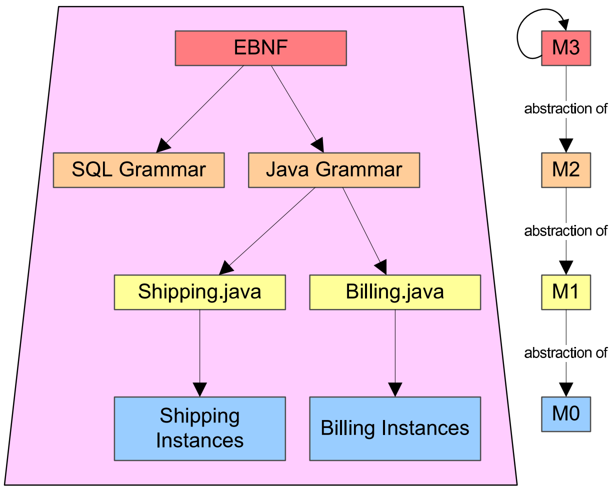


Figure 4.1: Programming languages in MOF

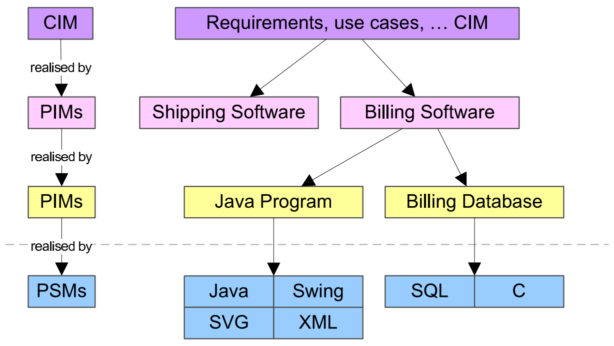


Figure 4.2: Levels of abstraction in MDA

a matrix, as shown in Figure 4.3[1]. This illustrates the relationships between all of the layers, and allows us to clearly illustrate the conceptual gap in web application development. Currently the two models highlighted in red do not exist, and the transformation from CIM to PSMs is manually done by the developer.

Our work is to create the *Web Application PIM* in this model, which is the source of the conceptual gap discussed earlier. Without this layer, the work required to translate the computationally independent layer (requirements and use cases) to the actual implementation layers (SQL statements, Java code) is significant. Also note the diverse range of M2 and M3 models used in the development of web applications;

[1]We are not yet sure what the real-world version of a platform-independent model would represent.

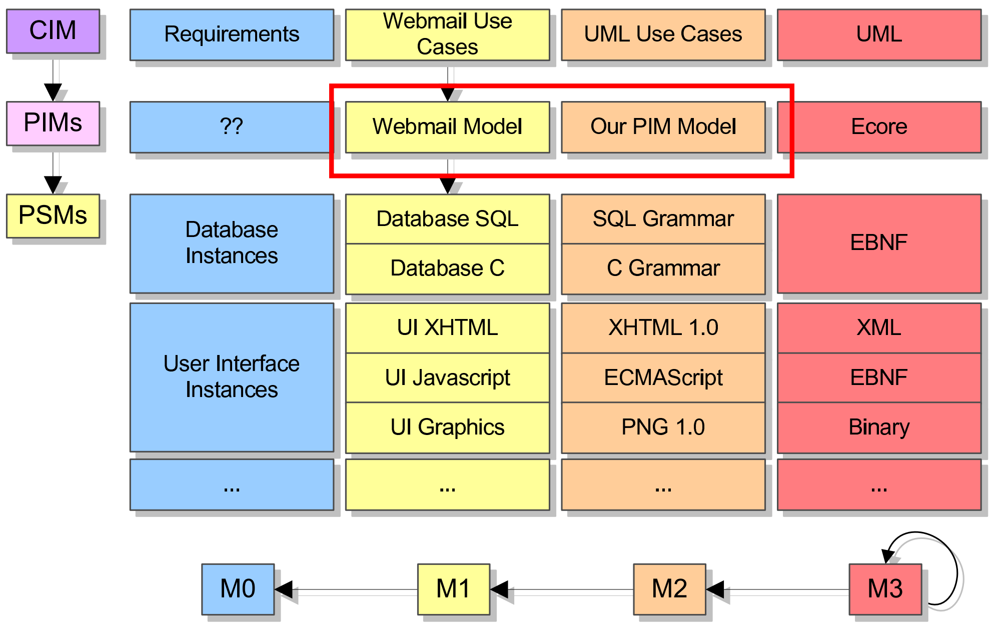


Figure 4.3: The conceptual gap in web development

it is specifically this diversity which adds such difficulty to the development process.

4.3 Metamodel Support

Another aspect to consider is the availability of computer-aided software engineering (CASE) tools. CASE tools improve the development process for architects in terms of reliability, stability and communication, and the use of these tools to also generate source code from the model is becoming more popular [30].

An important benefit of using a standard metamodel for model development is the developer may gain an almost-free CASE tool. Existing platforms such as Eclipse's plugin framework, Microsoft's DSL toolkit[2], and NetBean's extension mechanism allows for developers to easily create a fully-functional graphical CASE tool, often with code generation facilities, directly from a metamodel specification.

We are specifically interested in Eclipse's EMF framework (also known as the Graphical Modeling Framework, GMF) [66]. It is hosted in the wildly successful Eclipse platform [67], which is open source and well documented. It is supported

[2]Codenamed *Whitehorse*.

by the software community and commercially by IBM, and is a very active topic of research [68, 69] and development [70]. It provides a metamodel editor (both graphically and textually) and can also import other metamodel formats [66, 71]. Through a wizard (Figure 4.4), this metamodel can be converted into a fully functional CASE tool plugin with code generation functionality in a few easy steps [72].

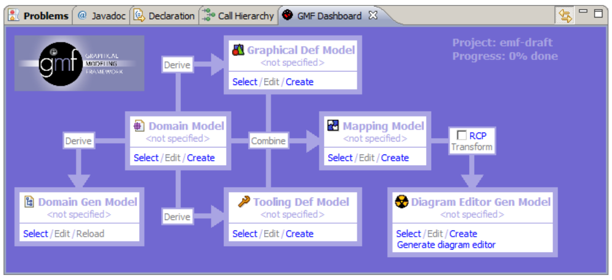


Figure 4.4: The GMF Framework Wizard in Eclipse

4.4 Transformations and Code Generation

Another benefit of models is the concept of *transformations*; these allow us to systematically transform an existing model into a new target model (or models), following a set of transformation rules [73]. For example we can use transformations to convert a WebML model into a UWE model, or to translate a DSL into the XMI standard format for use in a CASE tool. These transformations can be achieved through XML stylesheets (XSLT: [74]), or less commonly through visual mapping tools [73].

A technically similar concept is the use of transforming models into actual code, known as *code generation*. Whilst still transforming an existing model into another model, code generation is focused on creating a concrete, detailed implementation in a specific target platform that can be executed. Code generation techniques aim to simplify development and improve portability and platform independence. A model-driven application provides benefits for future maintenance, reducing costs and increasing stability in the future.

It is common to split an application into the three MVC layers, as first pioneered in Smalltalk [75]: presentation (user interface), domain logic, and the data itself. This reduces coupling and dependencies, and improves the conceptual separation between the design and the implementation. It is especially common in web development, as it simplifies complexity and increases availability and security [3]. Similarly, by abstracting the code generation into separate layers using *templates*, we can achieve a more modular and clearly defined approach to code generation. EMF natively supports code generation through the use of Java Emitter Templates (JET) [76], which follows this template-based approach. It uses a textual syntax similar to scripted server-side pages for the generation of code in any language [77].

Another approach to model transformations is the upcoming standard of QVT (MOF Query/Views/Transformations [78]), as recommended by the OMG. QVT aims to bridge different models together in the MOF heirarchy by providing a standard transformation language. The standard is currently being finalised and CASE tools are beginning to support this technology [79, 80].

4.5 Verification

Verification of a model can initially be achieved through using the Object Constraint Language (OCL) [81] to define constraints on the composition of the model, however this only verifies that the model is valid, and cannot verify that the system will satisfies higher-level requirements.

An important feature we aim to address is to verify the implementation of the model and its execution, through the use of formal model verification. Run-time testing of implemented applications remain the standard for verifying application correctness [82], but much work has been done in validating a model derived from the application implementation [10, 83, 84], or more appropriately in our situation, the original model itself [34].

By verifying the model directly instead of the implementation, we can evaluate functional properties of web applications, such as concurrency issues and the interaction between multiple users. We can also evaluate non-functional requirements such as broken links, syntax validation, load testing, page response time [85], security, and accessibility.

Integration of model verification into our anticipated CASE tool would significantly improve the visibility of web application verification. Some work has been done in EMF to integrate model validation directly into its framework [86], however the integration of an existing industrial-strength validator such as Alloy [19] would likely improve the viability of model verification.

Chapter 5

Progress

Our first three iterations have been completed. In these iterations we have identified key examples of interactive web applications on the Internet (Appendix B); concentrated their features into a comprehensive set of requirements for a modelling language (Appendix C); critically evaluated existing approaches (Table 3.1); and prototyped development in EMF to identify it's suitability as a target development platform. Along with this work we have presented a number of publications and seminars to promote our work.

5.1 Publications

In the last year we have authored two papers for two Australasian conferences, with both accepted and presented by myself.

- In November 2007 I presented my first paper, *Towards an Interactive Web Modelling Language* [87], at SIENZ 07 in Auckland. In this paper I presented and discussed the research programme to software engineers and postgraduate students.
- In January 2008 I presented our second paper, *Survey of Existing Languages to Model Interactive Web Applications* [16], at the Asia-Pacific Conference on Conceptual Modelling 2008 in Wollongong, Sydney. In this paper we presented our model requirements, a survey of five existing academic languages to model interactive web applications, and discussed our future work.

- In March 2007 I presented a short seminar at Massey University on WebML as part of the Software Engineering seminar series in 2007. In this seminar I presented my research proposal along with a preliminary investigation into WebML.
- Early this year in February I gave an informal presentation at Victoria University, Wellington, where I once again presented our research proposal and our early prototyping work on the metamodel. The feedback I obtained from this presentation was invaluable. This presentation will be followed up with a formal computer science seminar in March.

Future conferences which I plan to submit papers or attend in the next 12 months (dependent on funding) include NZCSRSC 2008 in Christchurch; WISE 2008 in Auckland; MODELS 2008 in Toulouse, France; APCCM 2009 in Wellington; ICWE 2009; VL/HCC 2009; ASWEC 2009; and WebStock 2009.

5.2 Future Direction

An important concern with developing an interactive modelling language is to quantitatively measure its suitability against other approaches; however, unlike other areas such as rule modelling[1] and enterprise software development[2], there is no standard industry benchmark for interactive web applications. Di Martino et al [90] note the lack of publicly available data for empirical evaluations – in their case, for web cost estimation – and instead collected data from private industry projects. [49] covers the implementation of a standard conference management web application in multiple methodologies, but no comparison or benchmark is provided, and this application is too trivial for our work.

As such, our third iteration will focus on the development of an appropriate benchmark for use in evaluation, along with developing some quantitative measures to compare with. Measures may be inspired by the work on the Tukutuku database [91], such as the number of new and reused web pages, the number of high-effort functions, and the actual total effort used in developing the application.

The development of this benchmark is crucial, as a benchmark will allow us to quantitatively evaluate existing languages; evaluate our extensions; evaluate the

[1]Rule modelling has the EURent fictional case study benchmark [88].

[2]Enterprise software has the Java Pet Store fictional scenario benchmark [89].

development of our own language; and understand more about the potential issues faced with new technology. The benchmark will be feature-based and inspired by popular web applications[3] and projects from industry; additional input from other researchers is similarly important. Once this benchmark application is developed, we can attempt to extend an existing language and evaluate our extensions according to this benchmark. This will allow us to conclusively decide if it is possible or sensible to extend an existing language, or if it would be beneficial to develop our own.

Following this, the language will be comprehensively designed and implemented along with a proof-of-concept CASE tool. Our work will then be re-evaluated with the same benchmark application to identify progress. Any issues discovered will be used to refine and finalise our final approach. We will also spend some time on working on model verification, and its integration with existing model checkers, as well as investigating potential integration into the development platform.

Some thought has gone into considering potential visual metaphors for the web application model, such as the use of implicit model elements to simplify the model; simplifying communication channels between common objects (such as database to interface); dynamic implementation stereotypes based on user agent capabilities; and drilling deeper into a model for specialisation. Research is needed to justify these interface decisions, but we note that this Ph.D. is focused more on developing a metamodel than the visual interface; this could be a good project for user interface research.

Discussion on the usefulness of the modelling language and its corresponding CASE tool support would be strengthened through evaluation of the model with industry developers. This could only be achieved once the project has matured enough to be user-friendly, but would nicely coincide with the public release of the tool. It would be important to evaluate the experiment in terms of the features of the model rather than the efficiency of the user interface surrounding it.

The final and most important goal in this project is to implement at least one real-world application, and use our approach to deploy it into at least two different platforms. This is an important validation step to show our approach is practical and suitable [26, 27], and will be a good source of discussion for the evaluation of our design. One piece of work that would also be useful is a simple translator to convert existing models in languages such as WebML into our new model.

[3]We have noted that most public sites, particularly those in Appendix B, use relatively few new technologies at once; to ensure that we are feature-complete, we will have to create our benchmark by combining all of them together.

#	Iteration	End Date	Status
0	**Requirements**	July 2007	Complete
1	**Existing Evaluation**	November 2007	Complete
2	**Prototype Solution**	January 2008	Complete
3	**Develop Benchmark**	March 2008	In Progress
4	**Language Extension and Evaluation**	May 2008	
5	**Model Development**	August 2009	
6	**CASE Tool Development**	November 2009	
7	**Evaluate Model with Benchmark**	February 2009	
8	**Verification and Integration**	May 2009	
9	**Model and CASE Tool Development**	July 2009	
10	**Real World Evaluations**	October 2009	
11	**Final Evaluations**	December 2009	
12	**Thesis Writeup**	February 2010	

Table 5.1: Proposed Timetable

5.3 Final Thesis

A conceptual structure for the final thesis is provided in Appendix D. Thanks to the deliverable-oriented approach of iterative development used in our research, we also plan to provide these final artefacts at the conclusion of this Ph.D.:

- A functional and practical metamodel (the modelling language)
- An industry-inspired interactive web application benchmark specification
- A functional CASE tool as a proof-of-concept application
- Code generation from the model to at least two platforms
- Model verification integration with existing model checkers
- Transformations between other languages
- Real world case study implementation and discussion

Appendix A

Glossary

AJAX
Asynchronous Javascript and XML; a set of technologies that allow web applications to execute client-side scripting, send asynchronous network requests, and access the client-side document object model (DOM) to provide a richer user interface.

Computer-Aided Software Engineering Tool (CASE tool)
Software tools used in the design, development and maintenance of software.

Desktop Software Application
Conventional software as that which requires the download and installation of the majority of the application onto a local machine.

Flash
A free, proprietary client-based software component that provides a richer interface to users than conventional web browsers, through the use of animation, rich 2D and 3D graphics, sound and video.

Interactive Web Application
See *Rich Internet Application*

Metamodel
A model of a model, highlighting the domain concepts and properties of the model itself.

Model
A representation of the essential apsects of an existing system (or a system to be constructed) which presents knowledge of that system in a usable form [92].

Rich Internet Application (RIA)
A web application which features a richer interface than conventional web applications, through the use of client-side scripting, asynchronous events and improved browser functionality, usually through *AJAX* or *Flash* [32, 15].

Software Application
Software performing productive tasks for users, such as word processors and calendar applications.

Web Application
Software with the same goal as desktop software applications, but instead execute primarily through web browsers connected to the Internet, and require little to no installation.

Web 2.0
An intangible concept which can be said to emphasises interaction, community and openness in web applications [93]; most popularly realised through the combination of *social networks* and *Rich Internet Applications*.

Appendix B

Interactive Web Applications

As part of our requirements discovery, we identified the most visible expressions of interactivity in web applications, and decomposed their features down into common requirements in Appendix C. These web applications are:

1. **Gmail**: A web-based e-mail client, including an integrated chat-box, POP3/SMTP integration and rich text capabilities.
   `http://www.gmail.com`
2. **Writely**: A collaborative web-based word document editor, allowing multiple users to edit a document at once. Now part of Google Docs.
   `http://docs.google.com`
3. **Google Calendar**: A calendar application, featuring mobile phone notifications and using external calendar data sources.
   `http://calendar.google.com`
4. **Google Maps**: A drag-and-drop interface to worldwide maps, which also allows mashups with other applications.
   `http://maps.google.com`
5. **YouTube**: A video-sharing site, allowing users to upload, view, rate and share their own videos.
   `http://www.youtube.com`
6. **Live Search Images**: A rich image-searching application with a fluid interface.

`http://images.live.com`

7. **Digg**: A collaborative news sharing site.
   `http://digg.com`

8. **Flickr**: A collaborative image sharing site.
   `http://www.flickr.com`

9. **Google Reader**: A *feed reader*, retrieving data feeds of content from other sites, which also features offline reading through Google Gears [61].
   `http://reader.google.com`

10. **Google Pages**: A rich visual editor for producing and publishing static web sites.
    `http://www.googlepages.com`

11. **iGoogle**: A personalised home-page with rich, scripted widgets contributed by developers.
    `http://www.google.com/ig`

12. **Last.fm**: A social networking site integrated with music, allowing users to share their music tastes and listen to personalised Internet radio.
    `http://www.last.fm`

13. **Facebook**: A social networking site providing a public API which applications can use to harness the power of the social network.
    `http://www.facebook.com`

Appendix C

Web Application Requirements

1. Update Data
2. View Data
3. Pagination
4. User Action Auditing
5. Server Transaction Support
6. Local Data Storage
7. Server Data Access
8. Persistent Client Data
9. Temporary Server Data
10. Uploading Files
11. User Authorisation
12. Password Reset
13. Session Support
14. Account Registration
15. Automatic User Auth
16. Static Views (HTML)
17. Async Form Validation
18. Client Form Validation
19. Server Form Validation
20. Multiple Browser Support
21. Mobile Phone Support
22. Remote Data Source
23. Active Remote Data Source
24. Data Feeds
25. Web Service User
26. Back Button Control
27. Opening New Windows
28. Client-Side Application
29. Mobile Phone Communication
30. E-mailing Users
31. E-mail Unsubscription
32. Persistent Errors
33. User Content Security
34. User Collaboration
35. Rich Help Tips
36. Interactive Map
37. Drag and Drop

38. Client Timer Support
39. Server Timer Support
40. Page Caching
41. Offline Application Support
42. Loading Time Support
43. Flash MP3 Support
44. Flash Support
45. Internationalisation Support
46. Logout Control
47. Single Sign-In Solutions
48. User Redirection
49. Keyboard Shortcuts
50. Undo/Redo Support
51. Browser-Based Chat
52. Pop-up Window Support
53. Incompatible Client Warning
54. Store Data in Local Database
55. Store Resources Locally
56. Communication with Plugins
57. Scheduled Events
58. Web Service Provider
59. Runtime Interface Updates
60. Out-of-Order Events
61. Backwards-Compatible Scripting
62. Spellchecking

Appendix D

Final Thesis Structure

1. **Abstract, Preface**
2. **Introduction**
   (a) Problem statement
   (b) Objectives
   (c) Approach
   (d) Outline of Thesis
3. **State of the Art**
   (a) The Need for engineering
   (b) Existing Languages
   (c) New Technologies
   (d) Existing Reviews
      i. Literature Review
   (e) Our Critical Review
   (f) Summary
4. **Enabling Technologies**
5. **Contributions**
   (a) Approach
   (b) Features
   (c) Goals
      i. Case study
      ii. User evaluation
      iii. Metrics
6. **Review**

   (a) Feature Evaluation
   (b) Benchmark Application
   (c) Case Study 1: benchmark
   (d) Case Study 2: real world project
   (e) User Evaluation
   (f) Metrics

7. **Evaluation**
   (a) Original Contributions
   (b) Discussion
8. **Conclusion**
9. **Appendices**
   (a) Interactive Web Applications
   (b) Language Requirements
   (c) Benchmark Details
   (d) Modelling Language Details
   (e) Implementation Details
   (f) Case Study 1 Implementation
   (g) Case Study 2 Implementation
10. **References**

Bibliography

[1] A. T. Manes, *Web Services: A Manager's Guide*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[2] Linda Heaton, "Unified Modeling Language (UML): Superstructure Specification, v2.0," Object Management Group, Tech. Rep., 2005. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/05-07-04

[3] J. Offutt, "Quality Attributes of Web Software Applications," *IEEE Software*, vol. 19, no. 2, pp. 25–32, 2002.

[4] E. Mendes, N. Mosley, and S. Counsell, "The Need for Web Engineering: An Introduction," in *Web Engineering*, E. Mendes and N. Mosley, Eds. Springer, 2006, pp. 1–27.

[5] H.-W. Gellersen and M. Gaedke, "Object-Oriented Web Application Development," *IEEE Internet Computing*, vol. 3, no. 1, pp. 60–68, 1999.

[6] M. J. Taylor, J. McWilliam, H. Forsyth, and S. Wade, "Methodologies and Website Development: A Survey of Practice," *Information & Software Technology*, vol. 44, no. 6, pp. 381–391, 2002.

[7] M. Lang and B. Fitzgerald, "Hypermedia Systems Development Practices: A Survey," *IEEE Software*, vol. 22, no. 2, pp. 68–75, 2005.

[8] S. S. Selmi, N. Kraïem, and H. H. B. Ghézala, "Toward a Comprehension View of Web Engineering," in *ICWE*, 2005, pp. 19–29.

[9] R. Gitzel, A. Korthaus, and M. Schader, "Using established Web Engineering knowledge in model-driven approaches," *Sci. Comput. Program.*, vol. 66, no. 2, pp. 105–124, 2007.

[10] C. Bellettini, A. Marchetto, and A. Trentini, "WebUml: reverse engineering of web applications," in *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2004, pp. 1662–1669.

[11] N. Koch and A. Kraus, "The Expressive Power of UML-based Web Engineering," in *IWWOST'02*, 2002, pp. 105–119.

[12] G. Kappel, W. Retschitzegger, and W. Schwinger, "Modeling Ubiquitous Web Applications: The WUML approach," in *International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, Yokohama, Japan, 2001.

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

[14] G. Rossi and D. Schwabe, "Model-Based Web Application Development," in *Web Engineering*, E. Mendes and N. Mosley, Eds. Springer, 2006, pp. 303–333.

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

[16] J. Wright and J. Dietrich, "Survey of Existing Languages to Model Interactive Web Applications," in *Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM 2008)*, Wollongong, NSW, Australia, 2008.

[17] J. Mukerji and J. Miller, "Model-Driven Architecture Guide, v1.0.1," Object Management Group, Tech. Rep., 2003. [Online]. Available: http://www.omg.org/cgi-bin/doc?omg/03-06-01

[18] Linda Heaton, "Meta Object Facility (MOF) Core Specification, v2.0," Object Management Group, Tech. Rep., 2006. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[19] D. Jackson, *Software Abstractions: Logic, Language, and Analysis*, Cambridge, Mass., 2006.

[20] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri, and E. Tosetti, "Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA," in *ICWE*, 2007, pp. 539–544.

[21] Mozilla Foundation, "Firefox 3 for developers," 2008. [Online]. Available: http://developer.mozilla.org/en/docs/Firefox_3_for_developers

[22] W3C Group, "HTML 5: A vocabulary and associated APIs for HTML and XHTML," W3C Working Draft 26 February 2008, Tech. Rep., 2008. [Online]. Available: http://www.w3.org/html/wg/html5/

[23] A. A. Cockburn, "The impact of object-orientation on application development," *IBM Syst. J.*, vol. 38, no. 2-3, pp. 308–332, 1999.

[24] Milena Litoiu, "Reduce complexity with model-driven development, Part 1: Use the IBM Software Development Platform to develop end-to-end solutions," IBM, Tech. Rep., 2004. [Online]. Available: http://www.ibm.com/developerworks/ibm/library/i-modev1/

[25] Eric Lopes Cardozo, "The seven habits of effective iterative development," IBM, Tech. Rep., 2002. [Online]. Available: http://www-128.ibm.com/developerworks/rational/library/1742.html

[26] J. C. Preciado, M. Linaje, F. Sanchez, and S. Comai, "Hypermedia Systems Development: Do We Really Need New Methods?" in *IS2002: Proceedings of the Informing Science + IT Education Conference*, Cork, Ireland, 2002.

[27] C. Barry and M. Lang, "A Survey of Multimedia and Web Development Techniques and Methodology Usage," *IEEE MultiMedia*, vol. 8, no. 2, pp. 52–60, 2001.

[28] R. Frigg and S. Hartmann, "Models in Science," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., 2008. [Online]. Available: http://plato.stanford.edu/archives/spr2008/entries/models-science/

[29] J. Bruck and K. Hussey, "Customizing uml: Which technique is right for you?" Tech. Rep., 2007. [Online].

Available: http://www.eclipse.org/modeling/mdt/uml2/docs/articles/Customizing_UML2_Which_Technique_is_Right_For_You/article.html

[30] T. O. Meservy and K. D. Fenstermacher, "Transforming Software Development: An MDA Road Map," *IEEE Computer*, vol. 38, no. 9, pp. 52–58, 2005.

[31] M. Matera, M. Matera, A. Maurino, S. Ceri, and P. Fraternali, "Model-driven design of collaborative web applications," *Softw. Pract. Exper.*, vol. 33, no. 8, pp. 701–732, 2003.

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

[33] WebRatio Group, "WebRatio AJAX Extension," 2007. [Online]. Available: http://www.webratio.com/WebRatio-AJAX.do

[34] N. Koch, "Classification of Model Transformation Techniques Used in UML-based Web Engineering," *Software, IET*, vol. 1, no. 3, pp. 98–111, 2007.

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

[36] J. Conallen, *Building Web applications with UML*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[37] Linda Heaton, "UML Profile for Schedulability, Performance, and Time, v1.1," Object Management Group, Tech. Rep., 2005. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/2005-01-02

[38] P. Merialdo, P. Atzeni, and G. Mecca, "Design and Development of Data-Intensive Web Sites: The Araneus Approach," *ACM Trans. Inter. Tech.*, vol. 3, no. 1, pp. 49–92, 2003.

[39] F. Garzotto, P. Paolini, and D. Schwabe, "HDM – A Model-Based Approach to Hypertext Application Design," *ACM Trans. Inf. Syst.*, vol. 11, no. 1, pp. 1–26, 1993.

[40] P. Fraternali and P. Paolini, "Model-driven development of Web applications: the AutoWeb system," *ACM Trans. Inf. Syst.*, vol. 18, no. 4, pp. 323–382, 2000.

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

[42] P. H. Carstensen and L. Vogelsang, "Design of Web-Based Information Systems - New Challenges for Systems Development?" in *Proceedings of the 9th European Conference on Information Systems (ECIS)*, 2001.

[43] Gentleware, "Apollo for Eclipse," 2007. [Online]. Available: http://www.gentleware.com/apollo.html

[44] T. Massoni, R. Gheyi, and P. Borba, "A model-driven approach to formal refactoring," in *OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. New York, NY, USA: ACM, 2005, pp. 124–125.

[45] M. Fayad and D. C. Schmidt, "Object-oriented application frameworks," *Commun. ACM*, vol. 40, no. 10, pp. 32–38, 1997.

[46] A. Gu, B. Henderson-Sellers, and D. Lowe, "Web Modelling Languages: The Gap Between Requirements and Current Exemplars," in *Proceedings Of The Eighth Australian World Wide Web Conference*, 2002.

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

[48] P. Fraternali, "Tools and approaches for developing data-intensive Web applications: a survey," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 227–263, 1999.

[49] D. Schwabe, Ed., *First International Workshop on Web-Oriented Software Technology*, 2001. [Online]. Available: http://www.dsic.upv.es/~west2001/

[50] D. Djuric, D. Gasevic, and V. Devedzic, "The Tao of Modeling Spaces," *Journal of Object Technology*, vol. 5, no. 8, 2006.

[51] J. J. Garrett, "Ajax: A New Approach to Web Applications," Tech. Rep., 2005. [Online]. Available: http://www.adaptivepath.com/publications/essays/archives/000385.php/

[52] ECMA International, "ECMA-262: ECMAScript Language Definition, Edition 3," ECMA International, Tech. Rep., 1999. [Online]. Available: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

[53] W3C Group, "Document Object Model (DOM) Level 3 Core Specification," W3C Recommendation 07 April 2004, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/TR/DOM-Level-3-Core/

[54] A. Ankolekar, M. Krötzsch, T. Tran, and D. Vrandecic, "The Two Cultures: Mashing up Web 2.0 and the Semantic Web," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2007, pp. 825–834.

[55] W3C Group, "Simple Object Access Protocol (SOAP) Version 1.2," W3C Recommendation 27 April 2007, Tech. Rep., 2007. [Online]. Available: http://www.w3.org/TR/soap12-part1/

[56] D. Winer, "XML-RPC Specification," UserLand Software, Tech. Rep., 1999. [Online]. Available: http://www.xmlrpc.com/spec

[57] RSS Advisory Board, "RSS 2.0 Specification," RSS Advisory Board, Tech. Rep., 2007. [Online]. Available: http://www.rssboard.org/rss-specification

[58] Yahoo! Inc., "Yahoo Pipes," 2007. [Online]. Available: http://pipes.yahoo.com/

[59] J. van den Broecke, "Pushlets White Paper," Just Object B.V., Tech. Rep., 2002. [Online]. Available: http://www.pushlets.com/doc/whitepaper-all.html

[60] Dojo Foundation, "The Dojo Toolkit," 2007. [Online]. Available: http://dojotoolkit.org/

[61] Google Inc., "Google Gears," 2007. [Online]. Available: http://gears.google.com

[62] W3C Group, "Accessible Rich Internet Applications (WAI-ARIA) Version 1.0," W3C Working Draft 4 February 2008, Tech. Rep., 2008. [Online]. Available: http://www.w3.org/TR/wai-aria/

[63] L. Fuentes-Fernández and A. Vallecillo-Moreno, "An Introduction to UML Profiles," *The European Journal for the Informatics Professional*, vol. 5, no. 2, April 2004. [Online]. Available: http://www.upgrade-cepis.org/issues/2004/2/up5-2Editorial.pdf

[64] A. Schleicher and B. Westfechtel, "Beyond stereotyping: Metamodeling approaches for the UML," in *Proc. 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, R. H. Sprague, Jr., Ed. IEEE Computer Society, 2001.

[65] D. Pilone and N. Pitman, *UML 2.0 in a nutshell*. 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA: O'Reilly Media, Inc., 2005.

[66] A. Gerber and K. Raymond, "MOF to EMF: There and Back Again," in *eclipse '03: Proceedings of the 2003 OOPSLA Workshop on Eclipse Technology eXchange*. New York, NY, USA: ACM Press, 2003, pp. 60–64.

[67] Eclipse Foundation, "Eclipse.org Home," 2007. [Online]. Available: http://www.eclipse.org

[68] K. Ehrig, C. Ermel, S. Hänsgen, and G. Taentzer, "Towards Graph Transformation Based Generation of Visual Editors using Eclipse," *Electr. Notes Theor. Comput. Sci.*, vol. 127, no. 4, pp. 127–143, 2005.

[69] U. Shani and A. Sela, "OO Design Methodology of a DSL using EMF: (demonstration for the telco revenue assurance domain)," in *OOPSLA '06: Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. New York, NY, USA: ACM, 2006, pp. 698–699.

[70] Eclipse Foundation, "Eclipse Projects," 2008. [Online]. Available: http://www.eclipse.org/projects/listofprojects.php

[71] A. Schauerhuber, M. Wimmer, and E. Kapsammer, "Bridging existing Web modeling languages to model-driven engineering: a metamodel for WebML," in *ICWE '06: Workshop proceedings of the sixth international conference on Web engineering*. New York, NY, USA: ACM, 2006, p. 5.

[72] Catherine Griffin, "Using EMF," IBM, Tech. Rep., 2003. [Online]. Available: http://www.eclipse.org/articles/Article-Using%20EMF/using-emf.html

[73] S. Sendall and W. Kozaczynski, "Model transformation: the heart and soul of model-driven software development," *Software, IEEE*, vol. 20, no. 5, pp. 42–45, Sept.-Oct. 2003.

[74] W3C Group, "XSL Transformations (XSLT) Version 2.0," W3C Recommendation 23 January 2007, Tech. Rep., 2007. [Online]. Available: http://www.w3.org/TR/xslt20/

[75] G. E. Krasner and S. T. Pope, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," *Journal of Object Oriented Programming*, vol. 1, no. 3, pp. 26–49, 1988.

[76] Eclipse Foundation, "Eclipse Modeling: Java Emitter Templates," 2007. [Online]. Available: http://www.eclipse.org/emft/projects/jet/

[77] Remko Popma, "Introduction to jet," Azzurri Ltd, Tech. Rep., 2004. [Online]. Available: http://www.eclipse.org/articles/Article-JET/jet_tutorial1.html

[78] Mariano Belaunde, "Meta Object Facility (MOF) 2.0 Query/View/Transformation 1.0 Beta 2 Specification," Object Management Group, Tech. Rep., 2007. [Online]. Available: http://www.omg.org/cgi-bin/doc?ptc/2007-07-07

[79] Borland Software Corporation, "Borland Together," 2008. [Online]. Available: http://www.borland.com/us/products/together/index.html

[80] T. Clark, T. Gardner, C. Griffin, and L. Tratt, "QVT technologies and Eclipse," Tech. Rep., 2003.

[81] Linda Heaton, "Object Constraint Language Specification, v2.0," Object Management Group, Tech. Rep., 2006. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/2006-05-01

[82] H. Miao and H. Zeng, "Model Checking-based Verification of Web Application," in *ICECCS '07: Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 47–55.

[83] A. Deutsch, L. Sui, and V. Vianu, "Specification and Verification of Data-driven Web Services," in *PODS '04: Proceedings of the twenty-third ACM SIGMOD-*

SIGACT-SIGART symposium on Principles of database systems. New York, NY, USA: ACM Press, 2004, pp. 71–82.

[84] E. D. Sciascio, F. M. Donini, M. Mongiello, and G. Piscitelli, "AnWeb: a system for automatic support to web application verification," in *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*. New York, NY, USA: ACM, 2002, pp. 609–616.

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

[86] Eclipse Foundation, "Eclipse Modeling Framework Project (EMF): Validation Framework," 2008. [Online]. Available: http://www.eclipse.org/modeling/emf/?project=validation

[87] J. Wright, "Towards an Interactive Web Modelling Language," in *Proceedings of the 2007 SIENZ Workshop*, Auckland, New Zealand, 2007.

[88] Business Rules Group, "Defining Business Rules – What Are They Really?" Tech. Rep., 2001. [Online]. Available: http://www.businessrulesgroup.org/first_paper/br01c0.htm

[89] Dean Wampler, "Cat Fight in a Pet Store: J2EE vs. .NET," ONJava.com, Tech. Rep., 2001. [Online]. Available: http://www.onjava.com/pub/a/onjava/2001/11/28/catfight.html

[90] S. D. Martino, F. Ferrucci, C. Gravino, and E. Mendes, "Comparing Size Measures for Predicting Web Application Development Effort: A Case Study," *ESEM 2007: First International Symposium on Empirical Software Engineering and Measurement*, pp. 324–333, 2007.

[91] E. Mendes, S. D. Martino, F. Ferrucci, and C. Gravino, "Cross-company vs. single-company web effort models using the Tukutuku database: An extended study," *Journal of Systems and Software*, 2007.

[92] P. Eykhoff, *System Identification, Parameter and State Estimation*. London: Wiley, 1974.

[93] D. E. Millard and M. Ross, "Web 2.0: Hypertext by any other name?" in *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*. New York, NY, USA: ACM, 2006, pp. 27–30.