# The Development of a Modelling Language for Rich Internet Applications

Jevon M. Wright
School of Engineering and Advanced Technology
Massey University
Palmerston North, New Zealand
j.m.wright@massey.ac.nz

## ABSTRACT
The relatively recent innovation of Rich Internet Applications (RIAs) has introduced important usability and reliability improvements to server-side web applications; however, no existing modelling language for web applications can model the new concepts involved. Our proposed Internet Application Modelling Language aims to provide a simple domain-specific language for RIAs. In this paper, we discuss the ongoing development of both a meta-model for this language and its accompanying CASE tool, which aims to provide a rich modelling environment for the design, development and deployment of RIAs.

## Categories and Subject Descriptors
D.3.2 [**Programming Languages**]: Language Classifications—*Multiparadigm Languages, Design Languages*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Web-based interaction*

## General Terms
Design

## Keywords
Rich Internet Applications, modelling languages, model-driven development

## 1. INTRODUCTION
The introduction of the Internet has arguably been one of the most significant in the world of software applications. The development of software applications is experiencing a considerable shift from desktop software towards distributed *web applications* which are published on the Internet. These applications can then be accessed across a wide range of devices and platforms, lowering the development cost of software applications; however, these applications often experience poor usability and performance, due to poor interactivity, limited bandwidth, and latency issues [16].


NZCSRSC 2010, April 2010, Wellington, New Zealand.

To address the usability and reliability problems of classic web applications, a new generation of web applications called *Rich Internet Applications* (RIAs) have emerged [1]. These combine client-side scripting, background network requests, and a modifiable document object model – technologies collectively known as AJAX [5] – to provide a richer and more reliable user experience.

Software development has arguably been improved by using modelling techniques such as UML [9]; it is clear that web application development should also have this support. Unfortunately, RIAs have not yet achieved the same modelling support as conventional software applications. In this short paper, we describe the development progress of the *Internet Application Modelling Language* (IAML), which aims to provide modelling support for the fundamental concepts of RIAs.

We begin this paper with an introduction on models and model-driven development in Section 2, and illustrate how these concepts may be used in assisting the development of RIAs. In Section 3, we discuss the current and future state of this new modelling language, which is implemented as part of a CASE tool as discussed in Section 4. Finally, we conclude this short paper in Section 5 with a discussion of some of our experiences, and on the future work in this research project.

## 2. MODELS
A model can encompass a wide range of concepts; the most consistent definition is that a model is a simplified abstraction of reality [7]. In our work we consider a model to be an abstract representation of a system under development, which can assist in implementing the real-world system; and in particular, a model representing another model is its *meta-model* [25]. For example, the meta-model of UML class diagrams allow us to design an object-oriented software system in terms of classes, inheritance, relationships, attributes and other artefacts [9].

There is a wide body of existing work which shows that no existing modelling language for web applications, such as WebML [2] or UWE [13], can model RIA concepts sufficiently [3, 17, 23]. Current research on extending existing languages to model RIAs fall short as they do not address the core problem – a RIA modelling language needs first-class modelling support for the fundamental concepts of RIAs, rather than simply providing visual components [22].

As a general-purpose modelling language, UML may be used to model a RIA; in our work, we argue that this approach is not yet suitable for model-driven RIA development. UML is expressive enough to model any software system, yet the language lacks a complete formal specification of the executable semantics of its models. A significant research effort is underway to provide executable semantics for UML models, but it is not yet complete [21].

The publication of semantics for models is a focus of the Model-Driven Engineering (MDE) approach, which argues that models can become first-class artefacts in developing a software application, rather than neglected secondary artefacts only used for documentation or design [19]. In particular, the semantic definition is necessary in order to correctly and consistently translate a given model instance into an executable system [8].

Our research is focused on the development of a domain-specific language for modelling RIAs, and directly providing the executable semantics necessary to generate RIAs from models. Our goal is therefore to design a lightweight language that can be easily implemented in a modelling environment. In particular, we are interested in developing a *platform-independent* modelling language, allowing us to abstract from the wide range of implementation technologies, and adapting to new approaches [25].

While in some situations it is possible to reverse engineer a system implementation into a model, it is not usually feasible to extract the structural intent of the original design from code [15]. Consequently our work is focused on the development of new models, or the translation of existing models from other languages (such as WebML or UWE), rather than reverse-engineering existing systems.

## 3. PROGRESS

After investigating existing modelling approaches and defining a list of core requirements of RIAs [24], we decided to develop a new modelling language for RIAs, rather than through the extension of an existing approach. This approach was chosen over adapting an existing language as no suitable base language was found with the necessary level of support and adaptability [22].

This approach also allows us to directly use RIA concepts as first-class modelling citizens. By re-using concepts from existing modelling languages where possible – such as adapting UML activity diagrams for operational modelling – we hope to improve the accessibility of IAML, whilst remaining platform-independent.

Our previous work into investigating a list of core requirements of RIAs included the definition of a benchmarking application called *Ticket 2.0* [23]. This benchmarking application aims to combine all requirements of RIAs into one single application; as a result, if this one application can be modelled by a particular modelling language, we can argue this same modelling language can model all RIAs.

A major risk in the development of a modelling language for RIAs is that the domain is still rapidly evolving; new technologies such as HTML 5 and offline application support had not been published at the start of this research. By using an iterative and evolutionary development approach, we hope to adapt to this rapidly changing environment, reducing risk and providing rapid feedback [4]. This helps us in adapting to new standards and concepts as they are designed.

Our modelling language, the *Internet Application Modelling Language* (IAML), already supports the definition of some aspects of Rich Internet Applications. We are currently in the fifth iteration of model development, which supports:

1. Operations (using UML activity diagrams).
2. Database modelling (based on UML class diagrams).
3. Events and conditions (based on Event-Condition-Action rules [12]).
4. Database querying and instancing.
5. Limited navigation and user interface modelling.
6. Role-based access control [18].
7. Some higher-level components, such as element synchronisation and user security.

One concern with the development of a modelling language is in balancing the level of detail in its design. Too much abstractness will result in an approach that cannot adapt to different situations; too much flexibility will result in models that are large and unmaintainable. Our previous work on *model completion* allows us to combine the modelling concepts mentioned above into a single meta-model [25]. A discussion of each of the models used in our approach is well outside the scope of this short paper.

After concluding the fourth iteration of model development, we re-evaluated our modelling priorities by attempting to model the *Ticket 2.0* benchmarking application. This evaluation highlighted significant areas of functionality that could not be modelled. This allowed us to develop a list of intended modelling priorities for the fifth iteration of model development, including:

1. Data types and input validation.
2. Listing, browsing and navigating through results.
3. E-mail modelling, possibly with messaging and concurrency models.
4. Richer user interface components.

The implementation of each of these modelling aspects remain future work. In particular, modelling of the e-mail lifecycle is an important web-specific concept that is often neglected.

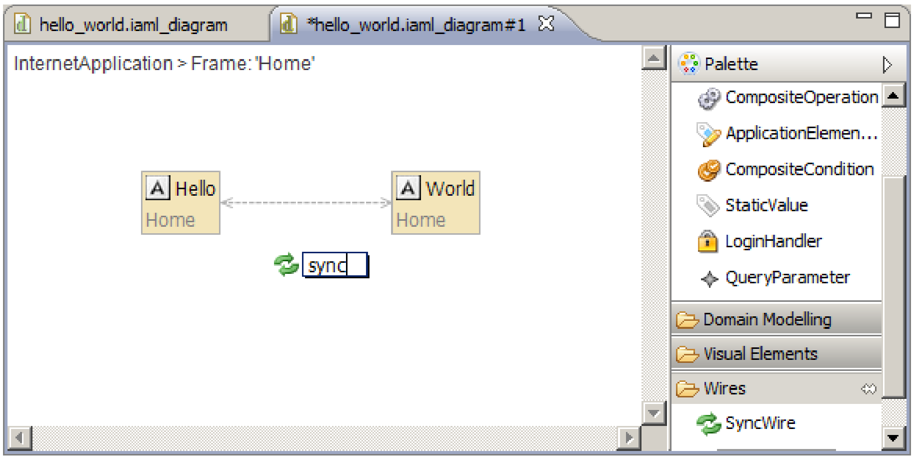


**Figure 1: The proof-of-concept CASE tool for the Internet Application Modelling Language**

## 4. IMPLEMENTATION

Along with the development of IAML, we are simultaneously developing a CASE tool to provide a rich modelling environment for the development of RIAs using this language [22]. This proof-of-concept implementation provides a complete environment for the development of RIAs, including a visual editor, validation framework, code generation and model completion [25].

For the meta-model, IAML is implemented using the meta-modelling Eclipse Modeling Framework (EMF) [20]. EMF allows for the rich definition of a meta-model compliant with the OMG's Meta-Object Facility specification [6], and natively supports serialisation to the XMI interchange format [10]. EMF is very stable and implementations do not depend on the Eclipse platform. A full description of this meta-model is well outside the scope of this paper, but may be viewed online at `http://openiaml.org`.

We use the GMF framework to provide a graphical model editor within the Eclipse environment. As RIAs are inherently visual, we expect that a graphical editor may assist in their design. In Figure 1, we illustrate our implementation of modelling two text fields connected by a *synchronisation wire*; the XMI serialisation of this example is omitted due to space, but may also be viewed online.

The CASE tool also supports basic model validation using an OCL-like syntax, and uses the *OpenArchitectureWare* framework to generate deployable web applications using PHP, HTML and Javascript [22]. In the future, an alternative implementation of this code generation approach into Java/JSP will illustrate the platform independence of our approach, which we expect to be straightforward.

Many other technologies were also investigated. The *Marama* modelling environment is a rich metamodelling tool for creating visual modelling tools, and has been used to develop a range of software development tools [14]. The commercial software *SmartDraw* supports the creation of custom graphical model editors, and the free software *ArgoUML* provides a UML 1.4 environment for defining model instances.

In our implementation, we chose to use EMF and GMF over these other technologies, as these Eclipse-supported approaches were developed by industry with a large developer community, and provide richer opportunities to integrate with other well-supported model-based technologies. The Eclipse framework itself simplifies the development of large software systems, using the concepts of plugins and features.

As we are developing this modelling language with an iterative development process, testing is very important in our approach. Each change to our language is supported by the introduction of additional test models and test cases. In particular, each new modelling concept is introduced along with many new test models and test cases. These tests range from unit tests to integration and acceptance tests, allowing us to verify that our implementation remains consistent throughout the development lifecycle.

Our rich test suite has also been useful in evaluating other aspects of our approach; for example, our test suite of 110 test models was used to evaluate an implementation of the *model completion* concept [25]. This suite may be useful in the development of a library of model examples, as part of the documentation of the language; or as a source of data for discussing the evolution of the modelling language.

## 5. DISCUSSION

In our experience, we have found that developing the IAML meta-model with EMF was a simple, straight-forward process. However, while GMF is still stable, it is still under development, so upgrading the Eclipse environment to a new version has occasionally introduced some incompatibilities. Our rich suite of test cases is invaluable for detecting such issues. The implementation of the code generator for model instances has been straightforward due to our rich suite of code generation test cases.

When using software development frameworks such as EMF and GMF, it is almost certain that some modifications to the generated software will have to be performed. In the development of this CASE tool, we have found that EMF-generated code required very few modifications, but GMF-generated code required many significant modifications. This was partially due to the relative infancy of the GMF project, as not all features have been implemented yet. However, the open source nature of GMF assisted us in making these changes, and developing automated tools to apply them.

One question that needs to be answered in this research is on how to document and publish the semantics of the modelling language. In particular, we need an authoritative source of documentation describing all aspects of the language. This remains an important area of future work.

Along with the publication of model semantics, we may leverage the model-driven development approach to apply additional verification and validation checks on a given model instance. For example, models of software may be evaluated to identify potential non-terminating loops [11]. We are currently investigating a range of techniques to this verification.

The final goal of this research will be the publication of a modelling language and an accompanying CASE tool, which can comprehensively model all RIA concepts, and may be used to develop executable web applications. The implementation and publication of the *Ticket 2.0* application within IAML is an important milestone for this achievement.

The work described in this paper has been implemented as free software under an open-source license, and may be downloaded from the website `http://openiaml.org`.

I wish to acknowledge and thank my supervisor Jens Dietrich for his support and guidance throughout my research.

## 6. REFERENCES

[1] A. Bozzon, S. Comai, P. Fraternali, and G. T. Carughi. Conceptual Modeling and Code Generation for Rich Internet Applications. In *Proceedings of the 6th International Conference on Web Engineering (ICWE '06)*, pages 353–360, New York, NY, USA, 2006. ACM Press.

[2] 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*, pages 137–157, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.

[3] S. P. Christodoulou, G. D. Styliaras, and T. S. Papatheodrou. Evaluation of Hypermedia Application Development and Management Systems. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HYPERTEXT '98)*, pages 1–10, New York, NY, USA, 1998. ACM.

[4] A. M. Davis, E. H. Bersoff, and E. R. Comer. A Strategy for Comparing Alternative Software Development Life Cycle Models. *IEEE Transactions on Software Engineering*, 14(10):1453–1461, 1988.

[5] J. J. Garrett. Ajax: A New Approach to Web Applications. Technical report, 2005.

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

[7] P. Haggett and R. J. Chorley. Models, Paradigms and the New Geography. *Socioeconomic Models in Geography*, 1967.

[8] D. Harel and B. Rumpe. Modeling Languages: Syntax, Semantics and All That Stuff, Part I: The Basic Stuff. Technical report, 2000.

[9] L. Heaton. Unified Modeling Language (UML): Superstructure Specification, v2.0. Technical report, Object Management Group, 2005.

[10] L. Heaton. XML Metadata Interchange (XMI), v2.1. Technical report, Object Management Group, 2005.

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

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

[14] N. Liu, J. Hosking, and J. Grundy. A Visual Language and Environment for Specifying User Interface Event Handling in Design Tools. In *AUIC '07: Proceedings of the Eighth Australasian Conference on User Interface*, pages 87–94, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

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

[16] L. Paulson. Building Rich Web Applications with Ajax. *Computer*, 38(10):14–17, October 2005.

[17] J. C. Preciado, M. Linaje, F. Sanchez, and S. Comai. Necessity of Methodologies to Model Rich Internet Applications. In *Proceedings of the 7th IEEE International Symposium on Web Site Evolution (WSE '05)*, pages 7–13, Washington, DC, USA, 2005. IEEE Computer Society.

[18] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996.

[19] D. C. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31, 2006.

[20] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework*. Addison-Wesley Longman, Amsterdam, 2nd edition, January 2009.

[21] G. Sunye, F. Pennaneac'h, W.-M. Ho, A. L. Guennec, and J.-M. Jezequel. Using UML Action Semantics for Executable Modeling and Beyond. In *Proceedings of the 13th Conference on Advanced Information Systems Engineering*, pages 433–447. Springer, 2001.

[22] J. Wright. A Modelling Language for Interactive Web Applications. In *Proceedings of the Doctoral Symposium at the 24th International Conference on Automated Software Engineering (ASE 2009)*, Auckland, New Zealand, 2009.

[23] J. Wright and J. Dietrich. Requirements for Rich Internet Application Design Methodologies. In *Proceedings of the 9th International Conference on Web Information Systems Engineering (WISE 2008)*, Auckland, New Zealand, 2008.

[24] J. Wright and J. Dietrich. Survey of Existing Languages to Model Interactive Web Applications. In *Proceedings of the 5th Asia-Pacific Conference on Conceptual Modelling (APCCM 2008)*, Wollongong, NSW, Australia, 2008.

[25] J. Wright and J. Dietrich. Non-Monotonic Model Completion in Web Application Engineering. In *Proceedings of the Doctoral Symposium at the 21st Australian Software Engineering Conference (ASWEC 2010)*, Auckland, New Zealand, 2010.