Requirements for Rich Internet Application Design Methodologies

Jevon M. Wright and Jens B. Dietrich

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

Abstract. Rich Internet Applications (RIAs) are quickly becoming the *de facto* standard for interactive web applications on the Internet, featuring rich interfaces that increase user usability and efficiency. These technologies increase the complexity of implementing web applications, making it difficult to address non-functional requirements such as application quality and reliability. There is much activity in developing modelling languages for web applications, but RIAs introduce additional concerns for application developers. Without identifying the requirements of interactive web applications, we cannot quantitatively compare different formal methodologies nor suggest they are robust enough for industry.
In this paper we present a comprehensive list of web application modelling requirements, derived from previous work and existing real-world interactive web applications. We use these requirements to then propose an industry-inspired benchmarking application, which allows us to evaluate approaches to handling the complexity of modelling real-world applications.

Key words: interactive web applications, Rich Internet Applications, web engineering, requirements, benchmark

1 Introduction

For the last decade, web applications are increasingly becoming the standard for communication and interaction, allowing any connected user on the Internet to browse information using standardised protocols and a web browser. Many approaches to model these web applications have been proposed in the past, such as WebML [1], UWE [2] and W2000 [3]. Recently, the concept of *Rich Internet Applications* [4] has arguably redefined the environment of web applications – advocating rich user interfaces and improving user participation – and is transforming users from content consumers to providers [5].

This has increased the complexity of web development, and consequently web developers have found a greater need for the use of formal methodologies to assist in the development and deployment of these interactive web applications [4]. However, very little work has been done in evaluating existing methodologies,

or proposing a comprehensive list of requirements of RIAs. This paper aims to satisfy these real needs by defining the expressive requirements of interactive web applications, and demonstrating the use of these requirements by proposing a sample benchmarking application.

We provide a brief background and our motivation for this work in Section 2. We then propose our requirements in Section 3, along with a discussion on their development. We combine these requirements into a fully featured benchmarking application, *Ticket 2.0*, in Section 4. A discussion of our contributions and future work is presented in Section 5, and we finally conclude our work in Section 6.

2 Motivation

Software development is a complex activity, and it is expected by industry that the use of formal methodologies and modelling languages to abstract away from this complexity increases the reliability, usability, security and maintainability of this software [6,7]. Web applications are a form of software that presents additional unique challenges and requirements to desktop software [7], and many modelling language approaches have been proposed to solve this additional complexity.

Despite this complexity, developers tend away from using such formal methodologies, instead advocating for proprietary or outdated approaches, even though formal methodologies are expected to be beneficial [6,8]. This may be a symptom of existing modelling languages being unable to express the unique requirements of web applications [7,9]; consequently, a methodology which is more expressible with regards to web applications should be beneficial to the web development community.

Past work on identifying the requirements of RIAs have tended to focus on the qualities of the methodology surrounding it or its software support [4,10], with less focus on the functional requirements of these applications. While web applications may be considered a primitive form of hypermedia [6], they have qualities that cannot be addressed with existing hypermedia modelling approaches [4]. In our paper we propose a comprehensive list of RIA requirements.

Along with correlating these requirements with existing work, we may also prove their validity by highlighting their actual usage in existing web applications. We amplify this step by consolidating all of these requirements into a proposed *benchmarking application*. Similar approaches are used in the domains of business rules [11] and enterprise software modelling [12], and this implementation is crucial to prove the real-world suitability of a formal methodology [6,13].

3 Requirements

Due to its relative infancy, there is little work on identifying the functional requirements of RIAs, with most research to date spent on identifying the technical and process requirements of appropriate methodologies. Whilst it is important

to consider the requirements of the process surrounding a modelling language, this is generally more flexible than the issues raised by the expressiveness of the modelling language itself. This is especially valid with web applications, as web concepts such as sessions and e-mails are largely ignored in existing approaches [14]. Consequently, we chose to develop our RIA requirements by studying real-world examples in industry:[1]

1. **Gmail**: Web-based e-mail by Google.
   `http://www.gmail.com`
2. **Calendar**: Google Calendar, a collaborative online calendar.
   `http://calendar.google.com`
3. **Reader**: Google Reader, an offline-enabled feed reader.
   `http://reader.google.com`
4. **Docs**: Google Docs, a collaborative office suite.
   `http://docs.google.com`
5. **Last.fm**: A social network-enabled music site.
   `http://www.last.fm`
6. **Pages**: Google Page Creator, an online web publishing suite.
   `http://pages.google.com`
7. **Facebook**: A social networking platform.
   `http://www.facebook.com`

In Tables 1 and 2 we present our proposed 59 core requirements of interactive web applications. Each requirement is based on an actual feature of RIAs, and is presented along with an example of their usage. They are grouped into six categories solely for ease of reference. We have purposefully ignored some basic data and presentation requirements;[2] these trivial aspects are covered by requirements such as *View Data* and are omitted for clarity.

These proposed requirements are ideal for evaluating and comparing different web modelling languages, and this approach has been taken before in evaluating older web modelling languages [4,9,10]. Indeed, it would be very useful to evaluate these requirements in a similar manner to Christodoulou et al [10]. As our previous work is concerned with a similar evaluation [14], we instead focus our attention on suitable methods to create and validate modelling language to address these requirements.

4 Benchmarking Application

Benchmarking applications[3] are a technique that may be useful in identifying the expressiveness of different technologies, and this concept has been used before

[1] The interested reader will note that most of these applications are developed by Google; indeed, Google has focused their business model significantly around RIAs.
[2] Such as the ability to use an external database, linking between pages, or being able to display the content in HTML.
[3] Instead of the classic definition of a performance benchmark, this is instead a *functional benchmark*.

#	Requirement	Example
Data		
D1	Static Pages	Gmail: Static help pages
D2	View Data	Gmail: View an e-mail
D3	Update Data	Gmail: Create an e-mail
D4	Pagination	Gmail: Display e-mails in pages
D5	Provide Data Feed	Last.fm: Provide RSS feed of recommendations
D6	Use Web Services	Calendar: Use external iCal feed
D7	Offline Data	Reader: Download new feeds before going offline
D8	Offline Resources	Reader: Download resources before going offline
D9	Web Service Provider	Facebook: Provide Facebook application using API
D10	Uploading Files	Gmail: Adding attachments
D11	Access Server Data	Gmail: Download new message headers
D12	Local Variables/Data	Docs: Download document source to client
D13	Cookies	Gmail: Recall last input language
Events		
E1	Scheduled Events	Calendar: Event reminders on client and server
E2	Client Timer Support	Gmail: Check server for new e-mails
E3	Server Timer Support	Gmail: Check POP3 servers for new e-mails
E4	Async Form Validation	Last.fm: Check in entered event artist data
E5	Client Form Validation	Gmail: Warn user if subject is missing
E6	Server Form Validation	Gmail: Sending an e-mail to an invalid address
E7	User Collaboration	Docs: Two users can work on the same document
E8	Browser-Based Chat	Gmail: Google chat
E9	Out-of-Order Events	Docs: Dealing with edit events with multiple users
E10	Server Transaction Support	Gmail: Purchasing more storage space
Users and Security		
S1	User Authorisation	Gmail: Sign in
S2	Session Support	Gmail: Sign in
S3	User Logout	Gmail: Sign out
S4	Automatic User Auth	Gmail: Log in automatically
S5	User Security	Calendar: Only certain users can access a calendar
S6	Group Security	Calendar: Shared calendars secured to certain groups
S7	Security Levels	Calender: Read/write/change sharing permissions
S8	Single Sign-In Solutions	Google Services; OpenID
S9	Personalisation	Calendar: Display a custom timetable format
User Agents		
A1	Browser Identification	Gmail: Redirect user if user agent fails requirements
A2	User Redirection	Gmail: Redirect to e-mail web links
A3	Multiple Browser Support	Gmail: Load different interfaces depending on agent
A4	Multiple Outputs	Calendar: Provide a feed in iCal, XML, HTML
A5	Client-Side Application	Gmail: Webmail application
A6	Load Additional Scripting	Gmail: *Contacts* menu loads another script
A7	Back Button Control	Gmail: A user cannot go back once logged out
A8	Plugin Support	Gmail: Play MP3 attachment
A9	Plugin Communication	Last.fm: Clicking on a track updates the Flash player
A10	Navigation Control	Gmail: Update URL fragment identifier

Table 1. Interactive Web Application Requirements (1)

#	**Requirement**	**Example**
Interaction		
T1	E-mailing Users	Gmail: Can send e-mails
T2	E-mail Unsubscription	Facebook: User can unsubscribe from all e-mails
T3	Mobile Phone Communication	Calendar: Can send text message reminders
T4	Internationalisation Support	Last.fm: Different locales
T5	Multiple Domain Support	Last.fm: Different domains display different locales
User Interface		
U1	Presentation	Calendar: Displaying a particular user interface
U2	Client-side Scripting	Gmail: Home page displaying available space
U3	Drag and Drop	Calendar: Can drag and drop events
U4	Loading Time Support	Gmail: Switch to HTML view after 30 seconds
U5	Keyboard Shortcuts	Calendar: Can browse using keyboard
U6	Opening New Windows	Pages: Open links in new windows
U7	Pop-up Dialog Boxes	Gmail: Can compose an e-mail in a new window
U8	Runtime Interface Updates	Gmail: Update *Unread Mails* in real time
U9	Static Views (HTML)	Gmail: Provide a static HTML view
U10	Modal Dialogs	Pages: Inserting an image shows a modal dialog
U11	Use External Components	Facebook: Transitions with *script.aculo.us*
U12	Provide External Libraries	Gmail, Calendar: A consistent calendar input box

Table 2. Interactive Web Application Requirements (2)

in a variety of different domains [11,12]. In the search for such a benchmarking application for web development however, we have not yet found any web application that matches all of our requirements simultaneously. We suspect this is due to the complexity such an application would burden on a development team, and it is precisely this reason that a structured formal methodology would be appropriate. This also means that there is no suitable application from which to build upon.

Ideally a benchmarking application for RIAs would involve the fields of social networking, e-commerce, web services, scheduled events, business integration and consumer interaction. It would be difficult to adapt common academic scenarios such as library or student applications to address all of these requirements. A sensible option would be adapting an existing web application, but existing applications are designed primarily for user simplicity and not feature usage. Extending an existing application may entangle too much additional complexity[4], which is important to consider when realising that a poorly designed modelling language may require significant model duplication[5] in order to fulfill the benchmark.

[4] Consider re-implementing Gmail from scratch, compared with implementing only a single client-side application.

[5] Consider that an application with client-side, server-side and mobile interfaces may require at least three separate but functionally identical models.

Whilst combining all RIA features into one application will exponentially increase its complexity, it is this complexity that will be a valuable learning exercise into how a methodology handles semi-realistic web applications. As such, we propose that a simple event ticketing application, combined with social networking features, is ideal. This proposed application meets all of the requirements we proposed in Section 3, as shown in Table 4. We also argue that while developing our own benchmarking application is definitely a challenge, it will be less complicated and more accessible in the long term than trying to extend an existing web application.

In the rest of this section we present our social networking-enabled, event ticketing application titled *Ticket 2.0*. Its business goal is to provide a rich interface for users to browse upcoming events and book tickets using a credit card. They may interact with other users on the site through friends lists and chat rooms on the event detail pages themselves, permitting open discussions and user interaction. It also aims to provide a unified interface for event managers, allowing them to schedule upcoming events and track their progress.

The conceptual structure of the application is presented in Figure 1, and the ticket booking application flow is shown in Figure 2. These figure have been purposely presented without using any existing modelling notations to try and be as independent as possible. Elements shaded gray indicate features that are navigable from every page,[6] likely as part of a common navigation header. Due to space restrictions this is not a complete formal specification, and the following sections will become quite technical, however straight forward for an experienced developer to implement. The full specifications for *Ticket 2.0* are available online at `http://openiaml.org/`.

4.1 Application Properties

The site is provided in two locales, with two separate domains selecting the appropriate display language. The user may be automatically logged in through cookie identification, if this feature is selected by the user. If the user visits with a mobile phone, a smaller set of the application is presented (highlighted in light gray), which does not include manager or administrator functionality. Pages marked with an asterisk may be visited offline if the user has appropriate technology[7] installed. The application is divided into three secure sections, which only particular types of users may access. The *Book Ticket* page involves a client-side application, and is described with detail in Section 4.6.

4.2 Public Pages

Home The home page describes the application, and allows the user to switch locales. It also allows the user to switch between the mobile and full versions of the application.

[6] These pages may be called *landmarks* [15].

[7] Such as Google Gears: `http://gears.google.com`.

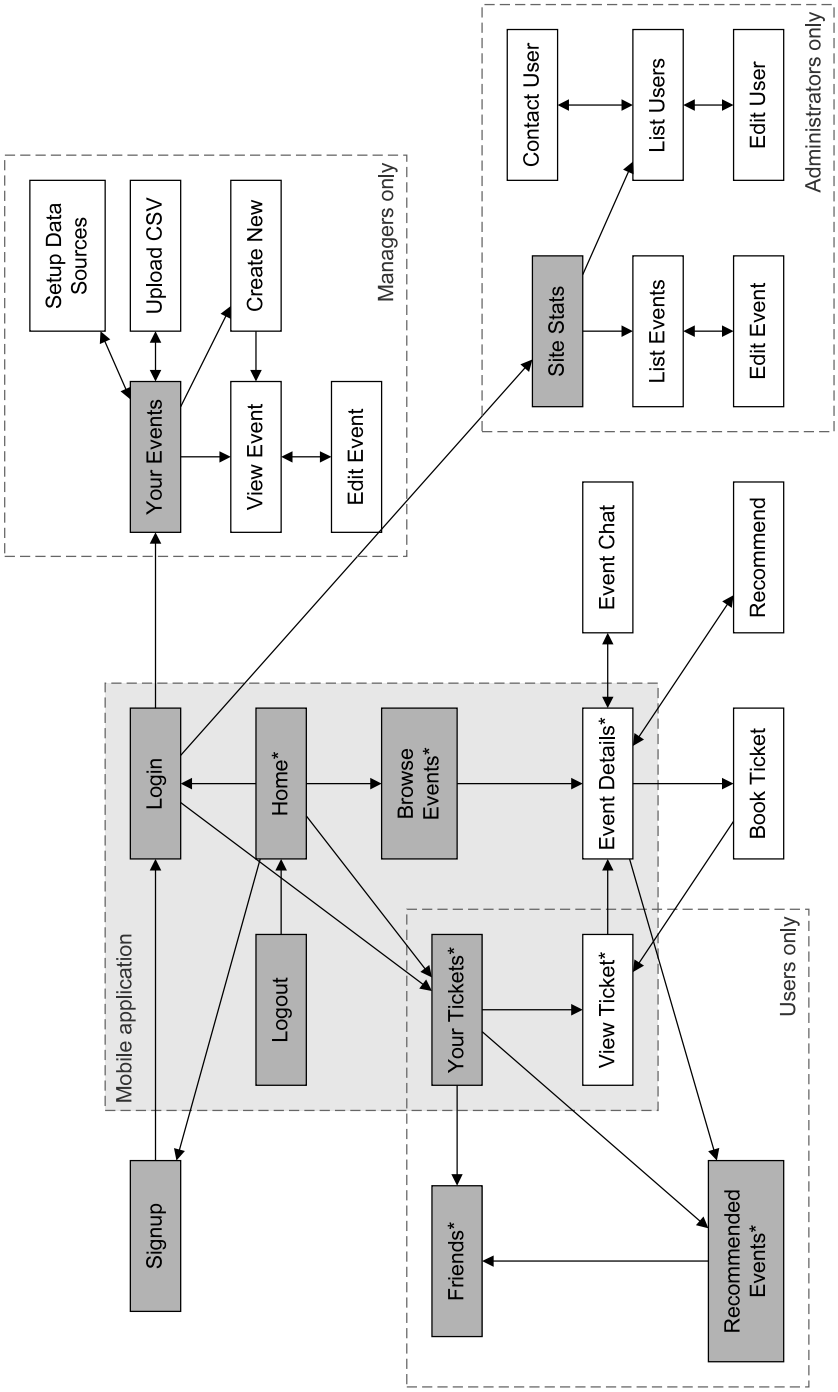

Fig. 1. Ticket 2.0 Application Page Structure

Signup, Login, Logout Allows the user to signup, login, or logout. These pages are secured through HTTPS. When logging in the user is presented with an option to remember their authentication details.

Browse Events Lists all events in the system, presents a Google Maps mashup [16] of events, and plays MP3 samples for selected events. It also provides a public API for event listings. The user may browse the listings using keyboard shortcuts, and uses a standard search widget. If not logged in, it uses cookies to recall the last browsed location.

Event Details Provides a Google Maps mashup of the event location. It may play an MP3 file uploaded by the manager related to the event. External links are opened in new windows. From here the visitor may purchase a ticket as described in Section 4.6.

4.3 User Pages

Recommended Events Displays events the application recommends to the user, in a manner similar to *Browse Events*. This page provides an external API to retrieve a certain users' recommended events, using a token key as authentication. The events are selected both by user recommendations and the system. It provides a drag-and-drop interface to delete unwanted recommendations. If offline, the deletions are saved and submitted once the application is online again. Once per week, the application sends out an automated e-mail to the user, listing new event recommendations.

Recommend Allows a user to recommend any event to another user or e-mail address. They may enter in multiple targets. If the e-mail address exists on the system and is not currently a friend of the user, a friend request is sent as well. If the target user is currently logged into the application, the recommendation is displayed in a popup window as well on the targets machine.

Event Chat A simple interactive chat popup window, which provides rudimentary communication between users. Visitors do not have to be currently logged in, but Users may add other Users as *friends*. It uses *script.aculo.us*[8] for event chat transitions.

Friends Allows the user to view, add and remove friends.

Browse Tickets, View Ticket Allows the user to browse and view previously purchased event tickets.

4.4 Manager Pages

After logging in conventionally, managers may authenticate themselves additionally using OpenID [17] before any changes are applied.

[8] A Javascript library providing rich object transitions and interactions: `http://script.aculo.us`.

Create New The manager may upload an MP3 file for the event. The event description is presented in a rich text editor, and the venue may be selected using an auto-completed text field. This page provides an external API to schedule new events, using a token key as authentication.

Your Events, View Event, Edit Event The manager may view the events they have added, and edit them with an interface similar to *Create New*.

Upload CSV The manager may upload multiple events in one file, according to a CSV format.

Setup Data Sources The manager may set up an RSS feed for new events. This RSS feed is checked daily for new events, which are then imported into the system. The manager may specify an additional OpenID authentication server.

4.5 Administrator Pages

This section is restricted to administrators only, allowing them to modify the core content of the site.

List/Edit Events/Users Lists the events or users on the system, and the administrator may edit their properties or remove them.

Site Stats Displays a simple overview of the traffic statistics for the site in a client-side application. Updates the fragment identifier while browsing between dynamic pages, which allows the administrator to bookmark the application state.

Contact User The administrator may send one or many users an e-mail or text message.

4.6 Ticket Booking Process

This process is described graphically in Figure 2. This operates as a client-side application that loads additional scripting to define the display of the venue details, and the user cannot navigate using their back button or history. Before the user reaches this, the user agent is checked for compatibility; if it cannot execute client-side scripting, the user is redirected to a separate static version of the booking process which is functionally similar, but lacks the richness of the client-side user interface.

The application displays a warning message if it takes longer than one minute to load the application. It has a three-minute client-side timeout, and a fifteen-minute server-side timeout; the current timeout situation is displayed in real time, along with the current connection status. If the client-side timeout occurs, the user is warned that the connection has failed; if the server-side timeout occurs, the booking process is cancelled and any reserved seats released. Input validation occurs both on the server and the client, as well as asynchronously in the background, with the client displaying errors in modal popup boxes on failed client-side validation.

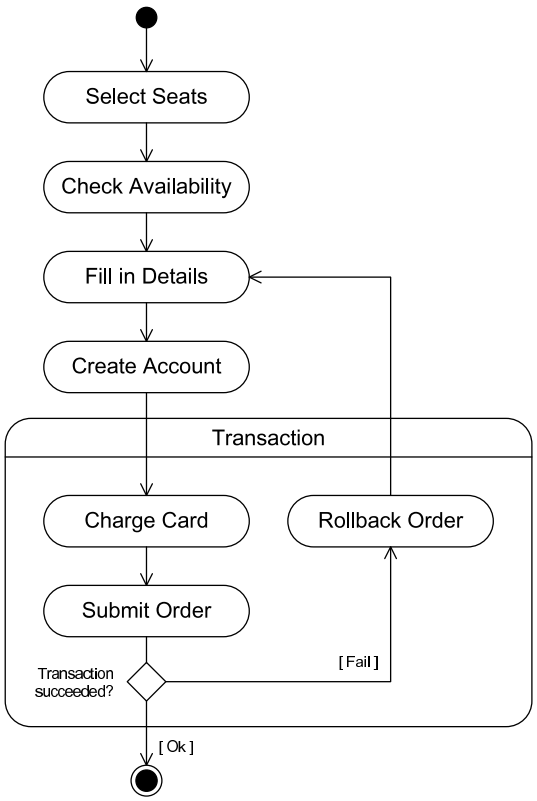

Fig. 2. Ticket 2.0 Booking Process

Select Seats Provides a graphical interface to select seats in the venue, displaying a map of the venue and clickable seat regions. If whilst selecting seats a seat region is expended on the server, all clients are notified and the region disabled on their displays.

Check Availability After successfully finding some seats in the desired region, the seats are booked. These are reserved until the user completes the process or the server timeout occurs.

Fill in Details, Create Account The user enters in the details for purchasing the tickets, and if the user is not currently logged in, they are asked to create an account, or authenticate themselves.

Charge Card Interacts with an external credit card billing provider, charging the user for the tickets. If either the card charging or order submission steps fail, the transaction is rolled back and the user is asked to confirm their details.

Submit Order Once an order is submitted, the user is e-mailed a confirmation, and a message is also sent to their mobile phone. The tickets are printed to PDF on the server, which are posted manually every morning by the printing office.

5 Discussion

Feature	**Matching Requirements**
Interaction	E4, U2, U3, U5
Multimedia	A8
Tool CASE	*n/a*
Visual continuity	A5, U2, U8
Synchronization	E2, E3
N-Tier development	*n/a*
Dynamic data retrieval	D7, D12
Parallel requests to different sources	D6, E4
Personalisation	A3, T4, S9
Interactive collaboration	E7, E9

Table 3. Comparison of Requirements Proposed by Precadio et. al [4]

These proposed requirements fit in well with existing discussions of Rich Internet Applications and hypermedia systems. Precadio et al. [4] propose ten RIA requirements, and finds that both web modelling languages and hypermedia languages are functionally lacking. Similarly, Christoudoulou et al. [10] present fifty hypermedia methodology requirements, and Gu et al. [18] propose sixteen specific web application requirements; many of these relate to the development process. While space limitations prevent us from including a full comparison of our work with existing requirements, we do summarise in Table 3 how our requirements match with those proposed by Precadio et al [4].

We do note that some of our requirements, such as keyboard shortcuts, automatic user authentication, client timer support and browser identification do not yet match up with any previously published requirements; we suggest that these are the new functional properties of RIAs that are currently neglected by existing approaches.

As mentioned in the beginning of this paper, much work has been accomplished in identifying the technical and methodological requirements of web modelling languages. Whilst identifying the functional requirements in this paper, some common themes emerged of these other requirements, such as the importance of a CASE tool; the ability to model using patterns; platform independence; integration with the business model; and management of the development lifecycle [10,18]. Gitzel et al. [7] adds an excellent discussion on the unique non-functional requirements of web applications, such as consistency and predictability. We acknowledge that while expressiveness in a modelling language is

Feature	**Requirements Fulfilled**
Application Properties	
Entire Application	D2 D3 E3 S1 S2 S4 S5 T4 T5 U1 U2 U12
Offline	D7 D8
Automated	D6 E1 E10 T1 T2
Mobile	A3 A4 E1
Public Pages	
Home	D1
Signup	E4-6 T1
Login	D13 S1 S2
Browse Events	A8 A9 D4 D5 D13 U5 U11
Event Details	A2 A8 U6 S9
Book Ticket	A1 A5-7 D6 D11 D12 E2-6 E9 E10 T1 T3 U4 U8-10
Recommend	T1 U8 S9
Event Chat	D11 D12 E7-9 U7 U8
Users Only	
Logout	S3
Recommended Events	A8 A9 D4 D5 U3 U5 U8 U11
Friends, Your Tickets, View Ticket	S6
Managers Only	
Your Events	D4 S6 S7 S8
Upload CSV	D10
Setup Data Sources	D6
View Event	A2 A8 U6
Create New	D9 E1 E4-6
Edit Event	E1 E4-6
Administrators Only	
Site Stats	A5 A10 D11 S6 S7
List Events/Users	D4
Edit Event/User	E5 E6
Contact User	T1

Table 4. Matching Ticket 2.0 Features to RIA Requirements

vital, it must satisfy these additional requirements in its implementation to be successful [13].

With respect to the ability of existing modelling languages to fulfill these requirements, previous work has already shown that no existing language is expressive enough for RIAs [4,10,14], so we omit such an evaluation in this paper. Future research in this area includes work on extending existing languages in order to address their shortcomings.

In the web application domain, previous benchmarking applications have included conference management systems [19], travel agencies [20] and movie databases [21]. Our contribution is a web application which is clearly aligned with the industry interests of interactive web applications, and specifically fulfills the requirements of RIAs (Table 4).

Other than simply implementing a benchmarking application, it is important to develop metrics to enable quantitative comparisons. This would allow a more precise comparison of different methodologies, and hopefully focus research efforts on addressing these real-world requirements. A discussion of suitable metrics is beyond the scope of this paper, but existing projects such as *Tukutuku* [22] should provide a source of inspiration.

Since this paper was submitted, we have successfully implemented the *Ticket 2.0* application using the Symfony framework for PHP. The implementation of this benchmark has already pointed out some interesting challenges that modelling languages would need to address, such as being able to model the intricate details of interacting with Google Maps and rich text editors, and keeping client-side and server-side interfaces synchronised. The next step in this research is to investigate these challenges and use our findings to improve the growing field of web application modelling.

To interact with a demonstration of this application, the interested reader is referred to the project website at `http://openiaml.org/`. This implementation may be considered as a reference for future research in this field. The reader is also encouraged to retrieve a copy of the benchmark specifications and implement them in their development platform of choice.

6 Conclusion

In this paper we have investigated existing real-world applications, and present a comprehensive list of requirements for Rich Internet Applications. We also propose a benchmarking application called *Ticket 2.0* which embodies all of these requirements in a familiar domain. We believe that this approach is an important step in being able to develop modelling approaches to handle the complexities of interactive application development and compare them quantitatively. By basing this benchmarking application in the same domain as existing industry applications, we also believe this contribution will prove favourable for industry discussion and support.

References

1. Ceri, S., Fraternali, P., Bongio, A.: 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) 137–157
2. Koch, N., Kraus, A.: The Expressive Power of UML-based Web Engineering. In: IWWOST'02. (2002) 105–119
3. Baresi, L., Colazzo, S., Mainetti, L., Morasca, S.: W2000: A Modelling Notation for Complex Web Applications. In Mendes, E., Mosley, N., eds.: Web Engineering. Springer (2006) 335–364
4. Preciado, J.C., Linaje, M., Sanchez, F., Comai, S.: 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) 7–13
5. Millard, D.E., Ross, M.: 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) 27–30
6. Lang, M., Fitzgerald, B.: Hypermedia Systems Development Practices: A Survey. IEEE Software **22**(2) (2005) 68–75
7. Gitzel, R., Korthaus, A., Schader, M.: Using established Web Engineering knowledge in model-driven approaches. Sci. Comput. Program. **66**(2) (2007) 105–124
8. Taylor, M.J., McWilliam, J., Forsyth, H., Wade, S.: Methodologies and Website Development: A Survey of Practice. Information & Software Technology **44**(6) (2002) 381–391
9. Selmi, S.S., Kraïem, N., Ghézala, H.H.B.: Toward a Comprehension View of Web Engineering. In: ICWE '05: Proceedings of the 5th international conference on Web engineering. (2005) 19–29
10. Christodoulou, S.P., Styliaras, G.D., Papatheodrou, T.S.: Evaluation of Hypermedia Application Development and Management Systems. In: HYPERTEXT '98: Proceedings of the ninth ACM conference on Hypertext and hypermedia, New York, NY, USA, ACM (1998) 1–10
11. Giurca, A., Wagner, G.: Rule Modeling and Interchange. Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on (26-29 Sept. 2007) 485–491
12. Dean Wampler: Cat Fight in a Pet Store: J2EE vs. .NET. Technical report, ONJava.com (2001)
13. Preciado, J.C., Linaje, M., Sanchez, F., Comai, S.: Hypermedia Systems Development: Do We Really Need New Methods? In: IS2002: Proceedings of the Informing Science + IT Education Conference, Cork, Ireland (2002)
14. Wright, J., Dietrich, J.: 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)
15. Nielsen, J.: Hypertext and hypermedia. Academic Press Professional, Inc., San Diego, CA, USA (1990)
16. Ankolekar, A., Krötzsch, M., Tran, T., Vrandecic, D.: 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) 825–834

17. Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In: DIM '06: Proceedings of the second ACM workshop on Digital identity management, New York, NY, USA, ACM (2006) 11–16
18. Gu, A., Henderson-Sellers, B., Lowe, D.: Web Modelling Languages: The Gap Between Requirements and Current Exemplars. In: AusWeb02: Proceedings of the eighth Australian World Wide Web conference. (2002)
19. Schwabe, D.: A Conference Review System. In: First International Workshop on Web-Oriented Software Technology. (2001)
20. MDWE 2005 Workshop: The Travel Agency System Example. Technical report (2005)
21. van der Sluijs, K., Houben, G.J., Broekstra, J., Casteleyn, S.: Hera-S: Web Design using Sesame. In: ICWE '06: Proceedings of the 6th international conference on Web engineering, New York, NY, USA, ACM (2006) 337–344
22. Mendes, E., Martino, S.D., Ferrucci, F., Gravino, C.: Cross-company vs. single-company web effort models using the Tukutuku database: An extended study. Systems and Software (2007)