META-DESIGN PARADIGM BASED APPROACH FOR ITERATIVE RAPID DEVELOPMENT OF ENTERPRISE WEB APPLICATIONS

Athula Ginige
AeIMS Research Group, University of Western Sydney, Sydney, Australia

Keywords: Enterprise web application development, Meta-design paradigm, Iterative development, Auto generation of web applications.

Abstract: Developing enterprise software or web applications that meet user requirements within time and budget still remains a challenge. The success of these applications mostly depends on how well the user requirements have been captured. The literature shows progress has been made on two fronts; improving ways requirements are captured and increasing interaction between users and developers to detect gaps or miscommunication of requirements early in the lifecycle by using iterative rapid development approaches. This paper presents a Meta-Design paradigm based approach that builds on work already done in the area of Model Driven Web Engineering to address this issue. It includes a Meta-Model of an enterprise web application to capture the requirements and an effective way of generating the application.

1 INTRODUCTION

Today enterprises are depending heavily on web based business applications to stay competitive. This class of applications are broadly categorised as Enterprise Web Applications. Still there are many challenges to developing Enterprise Web Applications within time and budget that meet user requirements (Krigsman 2008). At the start approaches that were taken to develop enterprise web applications came from software engineering. Since then these have evolved to meet specific characteristics of Web based applications. Yet Software and Web development approaches have many things in common.

Many of the studies on software project failures have identified difficulties of accurately capturing user requirements as a major contributor to failure of software development projects (Standish_Group 1995; Sauer, Gemino et al. 2007). Researchers have found that this equally applies for web applications as well (Escalona and Arago´n 2008).

Much effort has been spent on developing better ways to accurately capture requirements. Yet the various survey results on success rate of software development projects show still there are major challenges to accurately capturing the requirements (Standish_Group 1995; Sauer, Gemino et al. 2007; Escalona and Arago´n 2008).

A good approach to find whether the developed system meets the user requirements is by giving the system to the users as quickly as possible and to get the users to use the system. If there was loss of information in capturing the requirements or if the requirements have evolved since these were initially captured then by rapidly developing the system and getting users to test it will bring these to light.

Thus we need better ways to capture requirements as well as approaches to rapidly develop the Web applications. In this paper we propose an approach based on Meta-Design paradigm which consists of an efficient model based approach to capture the requirements and a model transformation approach to rapidly generate the application.

2 RELATED WORK

The literature shows in recent times progress has been made to address the issues related to requirements capture on two fronts; first by improving ways requirements are captured (Escalona and Koch 2007; Escalona and Arago´n

Ginige A. (2010).
META-DESIGN PARADIGM BASED APPROACH FOR ITERATIVE RAPID DEVELOPMENT OF ENTERPRISE WEB APPLICATIONS.
In *Proceedings of the 5th International Conference on Software and Data Technologies*, pages 337-343
DOI: 10.5220/0003014003370343

2008). The second approach is to use rapid iterative application development methods that enable users to frequently see parts of the physical system as the application get developed and provide feedback. Getting frequent user feedback helps to detect gaps or miscommunication of requirements early and correct these accordingly.

Conventional software development approaches consist of requirement analysis, design, implementation and testing (Boehm, Egyed et al. 1998). Application requirements need to be specified at the start of a development process. Once implemented, making changes to the implemented application become very expensive and time consuming. Thus trying to rectify any missing requirements is not cost effective with these development approaches. Also for the same reason these approaches are not suitable to develop evolving web based applications.

Agile methods (Highsmith 2002) also need requirements of the application to be specified at the beginning of a development project. However with Agile methods it is much easier to make changes to the implemented software; yet it requires fair amount of time and effort.

Model Driven Architecture (Soley 2000) and Model Driven Development (MDD) are developed by the Object Management Group (OMG) to manage the changes in technology and the proliferation of different kinds of middleware. Model Driven Web Engineering (MDWE) (Moreno, Romero et al. 2008) evolved from Model Driven Development to provide a hybrid approach that uses models to capture requirements and using model transformation techniques provide a way to rapidly develop the application.

Escalona and Aragon (Escalona and Arago´n 2008) have analysed set of empirical studies where various modelling techniques were used to capture user requirements and the process of translating these into analysis models required by designers to implement the system. They concluded that it is very difficult to translate users' necessities into Web analysis models as requirements are frequently defined using nonformal models.

This highlights the need for a comprehensive model to capture the essential aspects of an enterprise web application.

3 META-DESIGN PARADIGM

Meta-Design paradigm builds on some of the concepts in Model Driven Web Engineering and End-User development. The meta-design paradigm was initially proposed in the context of end-user development by Fisher (Fischer, Ye et al. 2004; Fischer and Giaccardi 2006). It was to accommodate evolution in Information Systems.

We have applied this concept to development of enterprise web applications. Instead of developing a specific enterprise web application, we developed a meta-model to store the individual application models and generate the application from the meta-model instanced values. The attributes of the meta-model will corresponds to different aspects of the physical application. When requirements change, the end-users can change the values in the corrosponding attributes of the meta-model and regenerate the application thus providing an easy way to evolve the web application. The relationship between a web application and the proposed Meta-Model is shown in figure 1.

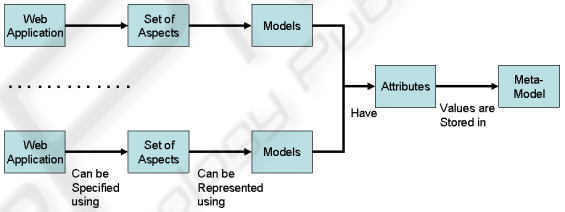

Figure 1: Relationship between a web application and the Meta-Model.

The figure 2 shows the conceptual model of two very common enterprise web applications; a leave processing system and a purchase requisition system.

The "Leave Processing System" can be viewed as an employee (actor) submitting a leave form which then gets routed based on a set of business rules to different actors for approval and processing. From time to time managers (another actor) can view leave records of the employees. Thus we can conceptualise this application as a form being routed based on a set of business rules and acted by different actors.

Similarly a purchase requisition is raised by an employee (actor) and acted by different people (other actors) based on a set of business rules. Thus one can see that both these applications are very similar at a conceptual or the meta-level. But in the detailed level, the leave object and the purchase requisition object will have different attributes, different business rules, different actors and different reporting requirements. If these specific details can be captured as instance values of a suitable meta-model then from these values we can generate both

the applications as well as many more applications that fit this meta-model.

As shown in figure 2 to auto generate a enterprise web application we need an object builder, workflow engine, access control system and an interface renderer to interpret the meta-model instance values and generate the applications.

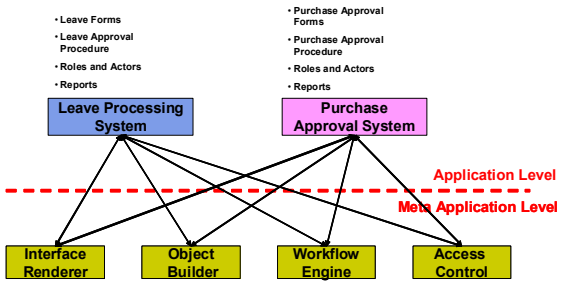

Figure 2: Example of a meta-design paradigm.

4 META-MODEL FOR ENTERPRISE WEB APPLICATIONS

As mentioned previously, we view web-based business applications as an instance of a Meta-Model. In theory, by creating a meta-model and developing tools to populate the instance values we can generate business applications. In practice however, creating a meta-model is not easy due to the complexities of business applications.

Many researchers have proposed different ways to model web applications (Schwabe, Rossi et al. 1996; Fratenali and Paolini 1998; Schewe, Thalheim et al. 2004). A comparison of these methods can be found in (Kappel, Pröll et al. 2006). Our task was to select appropriate models to represent different aspects of a web application and create an overall meta-model to store attributes required to specify these aspects.

The meta-model that we developed for Enterprise web applications is shown in figure 3. It is important to note that Meta-Model attributes are high-level compositions of many basic attributes. For example the "Rules for Sequencing Use Cases" is represented as a state table. Thus this high level attribute is a model in its own right and consists of set of low level attributes to capture the details required to represent a state table.

Below we explain how specific aspects of a web application can be modelled as attribute values of the Meta-Model shown in figure 3 using an example of a Customer Relationship Management (CRM) System. Figure 4 shows a screen shot of the generated application using instance values stored in the Meta-Model.

As can be seen from the expanded CRM menu in the figure 4 this application has many use cases and a link is provided from the Menu to perform the activities in these use cases. Some of these use cases are Company Details, Contact Details, Deals Management, Task Management, ….. and Case Management.

Most of the Use Cases are to manage some aspect of an information object. These use cases belongs to the Business Process category in the meta-model and have specific tasks such as create a new instance of the information object, edit, delete view and approve. Typically approval process consists of few use cases where different actors need to perform set of tasks in a pre defined order. These rules need to capture in the "Rules for Sequencing Use Cases" attribute of the meta-model.

In our previous research we have developed the concept of Smart Business Objects (SBO) (Liang and Ginige 2006); objects that are aware how to render, validate and store itself according to data type of the attribute. These SBOs recognise high-level data types such as email, address, photo, description, document in addition to the low level data types such as binary, varchar etc.

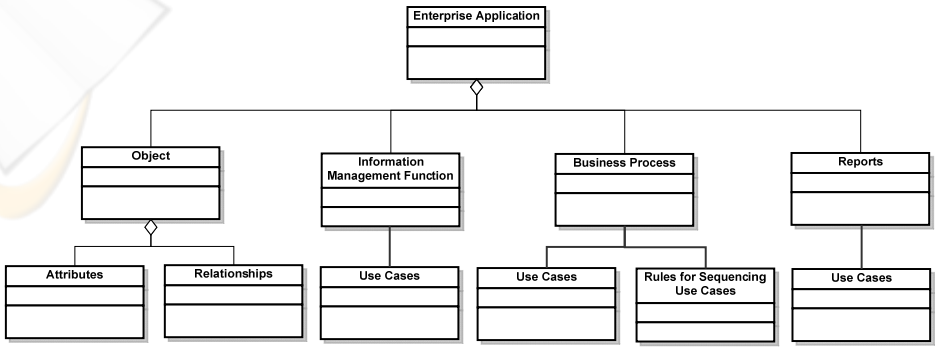

Figure 3: High-level Meta-Model of an Enterprise Web Application.

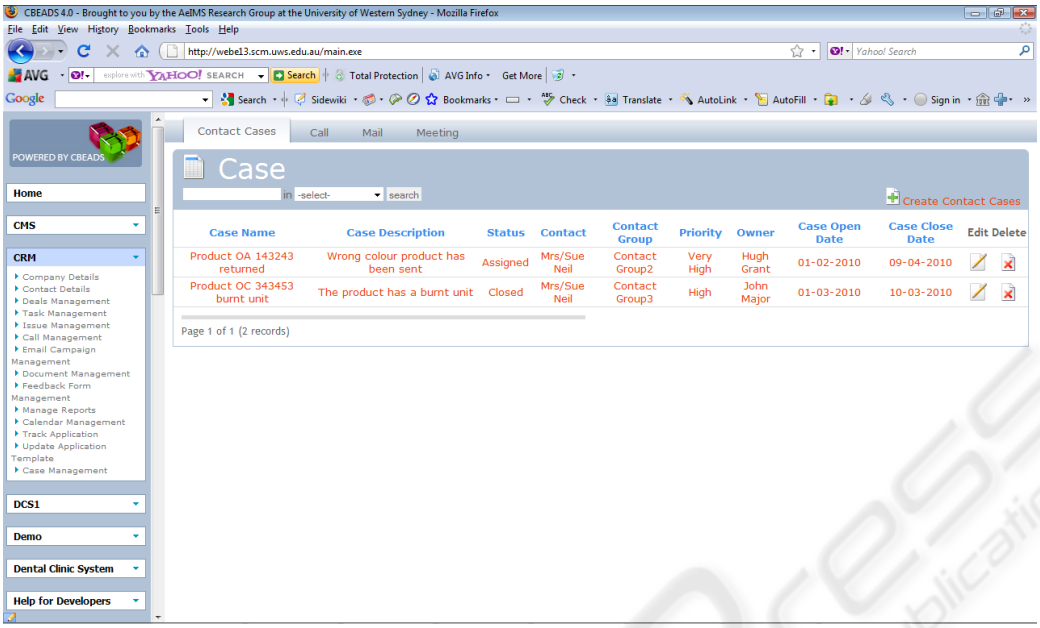

Figure 4: Screen Shot of the CRM application generated using meta-deign paradigm.

We used SBOML "Smart Business Object Modelling Language" (Liang and Ginige 2008; Liang 2009) to capture details of various businesses objects used in Enterprise Web applications. The expressive power of SBOML greatly simplified the number of attributes required in the Meta-Model to fully capture all the details of the enterprise web applications.

Below we show using SBOML how we can specify a "contact cases" object and a related "owner" object in our CRM application.

in crm, contact_cases has case_name, case_description, status(which could be Unassigned or Assigned or Closed or Re-Opened), contact(id), contact_group(id), priority(High or Medium or Low), owner(has employee_no, name, email, home_phone, mobile, address), case_open_date(date), case_close_date(date)

This SBOML expression is entered through a web form and stored in the object table of the Meta-Model for enterprise web applications as an attribute value. When generating the "Customer Relationship Management" application, a new database with the name (namespace) "crm" is created. In this database two tables or classes; "contact cases" and "owner", are created with the attributes specified above using SBOML. At any time we can edit the Business object specification in the Meta-Model object table and regenerate the business objects.

The Meat-Model also has database table where we store the properties of the attributes. The main properties are the data types such as varchar, integer, date etc, how to validate the input values using a regular expression, how an attribute when rendered as an input on a web form should appear such as text box, radio buttons, check boxes, text box with a WYSIWYG JavaScript editor, upload button or a date picker if the attribute is a date etc. Similarly how an attribute when rendered as an output on a web browser should appear such as if the attribute is an email with an "mailto" link, if the attribute is an address then as a location on Google maps, etc also get stored. This now allows enterprise web application objects to be specified using high-level application domain concepts such as documents, photos, email, birthday, address etc. When developing multiple applications, this ability to reuse high-level application domain knowledge becomes very useful.

SBOML uses following syntax to represent relationships among objects.

crm contact_cases has crm contact

crm contact_cases has many crm contact_group

These text strings are stored in the relationship table as attribute values and use during application generation time to physically create the relationships among objects in the enterprise web application.

Use cases support CRUD (Create, Read, Update and Delete) operations on a namespace (set of classes) class (table) or an instance value in a class (record). How the use case specifications are stored

in the use case table in the meta-model is best described by first showing an interface that get generated when we create the required enterprise application and then working backwards.

Figures 4 and 5 show two generated user interfaces for the "Case Management" use case in a CRM application that we developed using this system.

The interface shown in figure 4 allows user to perform CRUD operations on instance values in 4 classes; Contact Cases, Call, Mail and Meeting. User can navigate from class to class using "Tab" menu at the top. Within each tab selected attributes of the instance values in that class are displayed in a table form. Also there are navigation links to "Create Contact Cases" as well as edit or delete existing "Contact Cases". Clicking on "Create Contact Cases" or edit button will navigate the user to the form shown in figure 5.

The specifications as to what classes to be displayed in the "Tab" menu, what attributes of instances in each class to be shown as table within each tab and when navigated to a create or an edit form what attributes to be shown are stored in the Use case table of the Meta-Model. This string expression is based on JSON data structure and for the above interfaces this string expression is given below.

```
{crm::contact_cases->render_as_menu(
        item_order => [
         'contact_cases',
         'call',
         'mail',
         'meeting',
        ],
         'create' => '1',
         'search' => '1',
         'edit' => '1',
         'ajax' => '1',
         'view' => '1',
         'delete' => '1',
        );
}
```

Rules for sequencing of use cases are captured as a state table. An example of a captured state table is shown table 1.

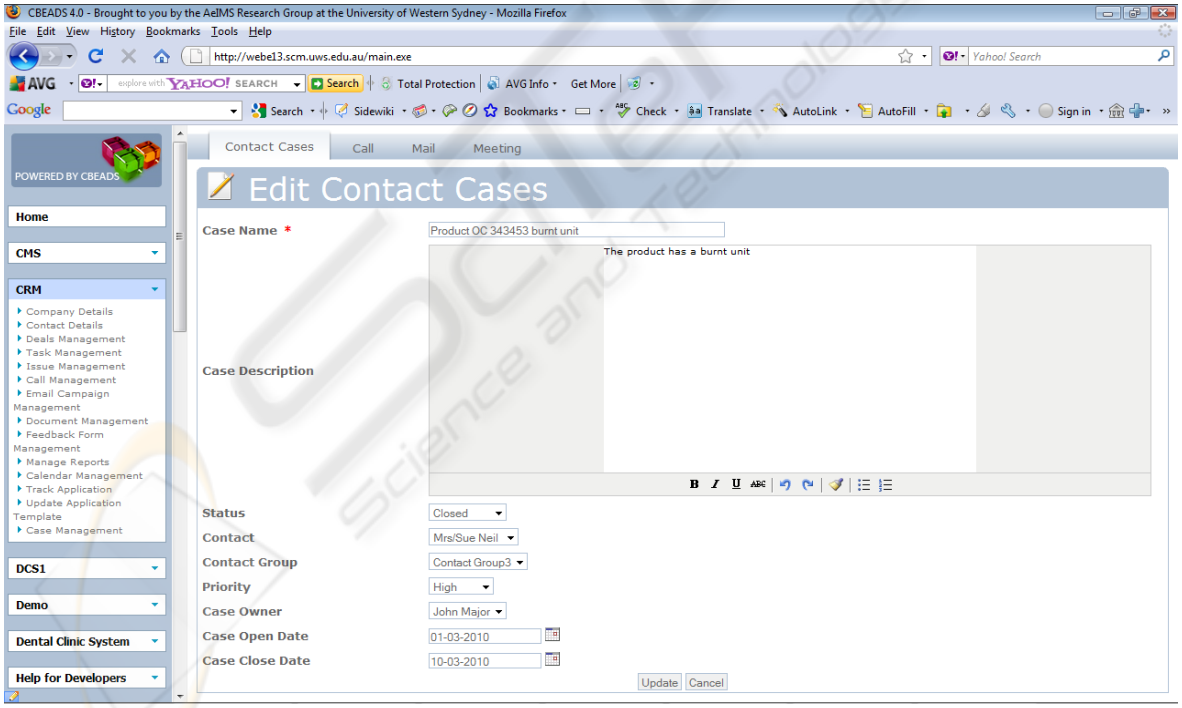

Figure 5: Form interface to edit a Contact Case.

Table 1: Showing Rules for sequencing of use cases as a state table.

Current State	Actor	Function	Buttons	Do Action	Next State
1	Contact	Complaint	Submit	Create Issue	2
2	Manager	Assign issue Owner Employee	Assign	Assign Owner, Email Owner	3
3	Owner	Process Issue	Process	Set Deadline	4
			Reject	Email Contact, Manager	5
4	Owner	Resolve Issue	Resolved	Email Manager, Contact	6
			Unresolved	Email Manager, Contact	
6	Manager	View Issues	-		6

5 CONCLUSIONS

In this paper we have presented a meta-design paradigm based approach to rapidly develop enterprise web applications. Using the framework that we developed, we are now able to develop enterprise web applications within days which could have taken weeks or months if developed in conventional ways. One of the major trials we did with the new approach was to develop a Customer Relationship management system for couple of Small to Medium size enterprises that we work with. As these business owners were able to see the discussed functionality and the interfaces that got generated in few hours after each meeting we had with them, they were able to interact closely with the team that developed the application and get things modified to suite their specific requirements.

This showed that we can get some basic specifications from the users, start creating the system using this approach and system iteratively get refined. It also supports evolution of user requirements which happened many times during this trial. Often when the business owners see a generated interface they wanted a new attribute added to a business object or change the way a particular attribute is displayed. With this system most of these changes can be done straight away and show to the business users to verify whether the requested change has been now implemented to the satisfaction of the user.

In these trails we have also identified few areas that require further research. At present use cases are limited to small set of CRUD operations that can be performed using set of pre defined interfaces. The use case model needs to be further refined to facilitate such wider range of operations.

This work has clearly demonstrated that the Meta-Design paradigm is suitable way to get Enterprise Web application that meet user requirements developed on time and within the agreed budget.

REFERENCES

Boehm, B., A. Egyed, et al. (1998). Using the Win Win Spiral Model: Case Study. IEEE Computer. July 1998: 33-44.

Escalona, M. a. J. and G. Arago´n (2008). "NDT. A Model-Driven Approach for Web Requirements." IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 34(3).

Escalona, M. J. and N. Koch (2007). Metamodeling the Requirements of Web Systems. Web Information Systems and Technologies, Springer Berlin Heidelberg. 1: 267-280.

Fischer, G. and E. Giaccardi (2006). Meta Design:A framework for the future of end user development. End User Development: Empowering People to flexibly Employ Advanced Information and Communication Technology. H. Lieberman, F. Paterno and V. Wulf, Springer. 9: 427-457.

Fischer, G., E. G. Y. Ye, et al. (2004). "Meta Design: A Manifesto for End -User Development." Communications of the ACM 47(9): 33-37.

Fratenali, P. and P. Paolini (1998). A conceptual model and a tool environment for developing more scalable and dynamic Web applications. EDBT 98, Valencia, Spain.

Highsmith, J. (2002). Agile Software Development Ecosystems, Addison Wesley.

Kappel, G., B. Pröll, et al. (2006). Web Engineering - Systematic Development of Web Applications, Wiley.

Krigsman, M. (2008). Study: 68 percent of IT projects fail. ZDNet.

Liang, X. and A. Ginige (2006). Smart Business Objects: A new Approach to Model Business Objects for Web Applications. 1st International Conference on Software and Data Technologies, Setubal, Portugal

Liang, X. D. (2009). Smart Business Object. School of Computing and Mathematics. Parramatta, University of Western Sydney.

Liang, X. D. and A. Ginige (2008). Smart Business Objects for Web Applications: A New Approach to Model Business Objects. Software and Data Technologies J. Filipe, B. Shishkov and M. Helfert. Berlin, Springer Berlin Heidelberg. Volume 10: 307-322.

Moreno, N., J. R. Romero, et al. (2008). An Overview of Model-Driven Web Engineering and the MDA. Web Engineering: Modelling and Implementing Web Applications. G. Rossi, O. Pastor, D. Schwabe and L. Olsina, Springer London: 353-382.

Sauer, C., A. Gemino, et al. (2007). "The impact of size and volatility on IT project performance." Communications of the ACM Vol. 50(No. 11): 79 - 84.

Schewe, K.-D., B. Thalheim, et al. (2004). Modelling and Stories in Web Information System. Information Systems Technology and its Applications (ISTA), Salt Lake Ciy, Utah, USA.

Schwabe, D., G. Rossi, et al. (1996). Systematic hypermedia application design with OOHDM. seventh ACM conference on Hypertext, Bethesda, Maryland, United States, ACM Press.

Soley, R. (2000). Model Driven Architecture, Object Management Group.

Standish_Group (1995). Chaos. THE STANDISH GROUP REPORT, The Standish Group