SOLVING DESIGN ISSUES IN WEB META-MODEL APPROACH TO SUPPORT END-USER DEVELOPMENT

Buddhima De Silva and Athula Ginige
University of Western Sydney, Locked Bag 1797, Penrith South DC, 1719, NSW, Australia

Keywords: Meta-model, end-user development, web application.

Abstract: End-user development is proposed as a solution to the issues business organisations face when developing web applications to support their business processes. We are proposing a meta-model based development approach to support End-User Development. End-users can actively participate in web application development using tools to populate and instantiate the meta-model. The meta-model has three abstraction levels: Shell, Application and Function. At Shell Level, we model aspects common to all business web applications such as navigation and access control. At Application Level, we model aspects common to specific web applications such as workflows. At Function Level, we model requirements specific to the identified use cases. In this paper we discuss how we have solved the issues in application development for business end-users such as need for central repository of data, common log in, optimizing user model, application portability and balance between "Do it Yourself" (DIY) and professional developers in hierarchical meta-model approach. These solutions are being incorporated into Component based E-Application Development and Deployment Shell (CBEADS©) version 4, supporting meta-model implementation. We believe that these solutions will help end-users to efficiently and effectively develop web applications using meta-model based development approach.

1 INTRODUCTION

The characteristics of the web such as ubiquity and simplicity make it a suitable platform to disseminate information and automate business processes. AeIMS research group at University of Western Sydney has been working with businesses in Western Sydney region to investigate how Information and Communication Technologies (ICT) can be used to enhance their business processes (Arunatileka and Ginige, 2004), (Ginige, 2006), (Ginige, 2005). In this work, we have identified many issues Business users have to overcome when trying to implement web applications (Ginige, 2005). These issues vary from not being able to get web applications developed to meet needs of the business in a timely manner to development projects running over budget. The development approach should also reduce the gap between what the users actually wanted and what is being implemented in terms of functionality (Epner, 2000). Researchers propose to empower end-users in web application development as a solution to these issues (Ginige, 2005), (Ginige and De Slva, 2007), (Costabile et al., 2005), (Fischer and Giaccardi, 2005), (Fischer et al., 2004).

There are different approaches suggested to empower end-users. One approach is to provide end-users with tools to develop any kind of web applications such as informational, search directory and directory look up, workflow and collaboration, e-commerce and web portal, etc. However, such tool will become complex because it has to cover a variety of features (Ginige, 2005). The other approach is to develop different tools to support different types of web applications. Our approach falls in to the second category. We have analysed the requirements for many business web applications. We identify that users need to store, process and reporting information. Sometimes the information may flow from one person to another in a business process such as a leave form being forwarded to the manager for approval. Thus business web applications have the "store-process-report information" pattern. Therefore meta-models for business web applications at conceptual level are form being routed based on rules, store information entered through forms, and produce reports.

De Silva B. and Ginige A. (2007).
SOLVING DESIGN ISSUES IN WEB META-MODEL APPROACH TO SUPPORT END-USER DEVELOPMENT.
In *Proceedings of the Second International Conference on Software and Data Technologies*, pages 298-304
DOI: 10.5220/0001348202980304

Example instance of that meta-model is a leave processing system where employees can apply for leave. Meta-model elements are high level abstract concepts such as user, role, form User Interface and business object.

We have encountered the design issues for business end-users such as need for a corporate information repository, common log in facility for all the applications with in the organization, sharing user attributes between different applications, separate the development tasks at granular level of aspects and application portability. In this paper we discuss how we address these design issues in the hierarchical meta-model to support the development of business web application. In section 2 we discuss the solutions to the design issues and section 3 presents the hierarchical meta-model. In section 4, we discuss the logical architecture of Component based CBEADS© framework that supporting the meta-model. Section 5 reviews the related work and section 6 concludes the paper.

2 SOLVING DESIGN ISSUES IN META-MODEL

A business organization has systems in place for proper operation of the organization. We develop web based information systems to support these systems to get the competitive advantages. Similar to organization holding the systems we develop a shell to hold the applications that are supporting the system. In other words the shell becomes the container for the applications. We view an application as a collection of functions. For example the leave processing application can have functions such as apply leave, approve leave, view leave history, etc. Functions deliver the functionality the users expect from an application. In the following sub sections we discuss how we solved the above mentioned design issues.

2.1 Corporate Information Repository

In a business organization we need to maintain a common data repository. This will avoid duplication of data which is a problem in having stand alone applications. It is required to manage the integrity of data for efficient use of information. We also need to facilitate sharing the corporate data between different applications to achieve efficient use of information. That means applications with in the shell need a common repository of objects.

During the development time we create/modify the business object definitions and relationships with in the application. But these business objects are stored at the shell level. Therefore all the applications with in a shell can use the same business object thus avoiding any duplication. The application that created the business object becomes the owner of that. Other applications can use the business object. They can extend the business object by adding more attributes. But they can't delete/change existing attributes. Shell/Application becomes the name space for the business objects. The central repository also helps to manage the data easily. Therefore the data management operations such as back up, recovery are available at the shell level.

2.2 Common Log in to All Applications

Once a user log in to a system he/she should be authenticated to all the functions which he is authorized to access. This helps to use their time efficiently. Having all the applications with in a shell, it facilitates to maintain a common login. The basic user object has user name, password name and e-mail address at this level. Name and e-mail address are used by the shell to communicate with user. For example, if user request for password retrieval it needs to be happened at the shell level and the shell uses the e-mail address to communicate with the user. User object is a special type of object with predefined template consisting of basic user attributes such as user name, password, first name, last name and e-mail.

2.3 Optimised User Model

Sometimes applications may want to record other attributes for the users. For example, in the leave application we may need to record the department of the employee to route the leave application correctly to his/her head of the department. That means we have to record head of the department for employee users. These user attributes are derived from the relationship "user plays the role of employee". We model the roles as groups. Similar to different applications sharing objects we may want to share the attributes at the group level. For example user group employee and head of the department both may need to record the attribute "department". Same time if a user becomes a member of both groups we want to use the same attribute value to minimize data entry and to avoid inconsistencies.

To solve these design issues we maintain a list of common group attributes at shell level. This facilitates sharing the group attributes between groups. For example, if end-user creates the employee object with attribute department, then it will be added to the group attribute list. Next time user create a group called head of department he can reuse the attribute definition "department". Then for each user we maintain the list of attributes in a user attribute object. This allows us to reuse the attribute values for same user.

2.4 Balance between DIY and Professional Developer

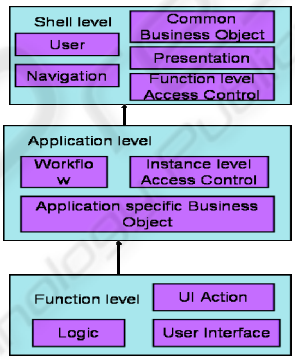
The research on end-user mental model shows that end-users have only little or no attention to low level critical concerns such as session management, database connection, etc (Rode et al., 2004). Therefore in the meta-model approach we model the aspects of web applications using high level abstractions such as user, object, process, hypertext and presentation. End-users may be able to model some of these aspects. In meta-model we provide the default models of these aspects to assist the end-user developers. For example, if end-user wants he/she can develop a web application with the default presentation style provided with in the meta-model. That means application can inherit the attributes defined at the shell level in the hierarchical meta-model. On the other hand we can override the shell level attributes at the application level. For example, if end-user wants a custom look and feel he may get a professional developer to do that for him. Inheritance and overriding at different levels help end-users to trade off between DIY and use of professional developer.

2.5 Application Portability

While solving all the above mentioned design issues we may also need to port the applications from one server to another. Since we create objects, group etc. with in an application, we store all these models at the application level. This will improve the portability of applications.

3 HIERARCHICAL META MODEL OF BUSINESS WEB APPLICATIONS

As mentioned previously we view a Web based business application as an instance of the meta-model. Theoretically by creating appropriate meta-model and developing tools to populate the instance values we can generate business applications. In practice creating this meta-model to support end-user development is not easy because of the complexities of business applications. To manage the complexity we developed the meta-model at 3 levels of hierarchical abstraction called Shell, Application and Function as shown in figure 1.



- Shell Level: Aspects common to many web applications such as user, navigation are modeled at shell level.
- Application Level: Aspects common to a web application such as workflow is modeled at this level.
- Function level: The function specific aspects required to implement the functionality are modeled at function level. Examples are user interface model UI and action model.

Figure 1: Hierarchy of Abstraction Levels of Meta-Model.

Shell Level

We analysed many business applications to identify common functionality required in most of the web applications. These common functionalities are executed at the shell level. User model, Access Control model, Navigation Model and Business Object Model are four models stored at this level. Shell level of meta-model is shown in figure 2. Each user has a profile. Profile can have many properties such as name, address, e-mail address, etc. User can play many roles. Role can have role attributes. A role can have one or more users. Each role has functions in an application assigned to it. Application has presentation properties. Each function associate to a menu link. Business Objects have many attributes and associations.

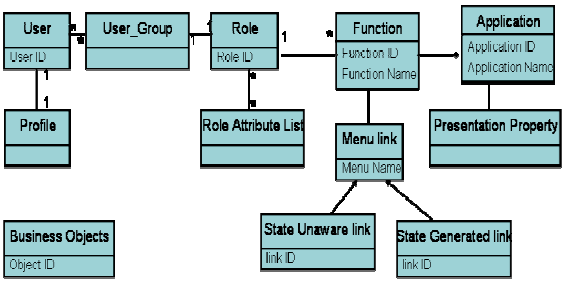


Figure 2: Shell level of meta-model.

User Model

There are users/participants who play different roles in any web application. Role has role attributes. User Model defines the user's profile and the roles users assigned to in a business web information system. The user model is used to authenticate the users. In user profile at least we maintain the properties user name, password, first name, last name and e-mail.

Function Level Access Control Model

Every business application has persistent functions which users want to use. Users are authorized to perform the functions based on the function level access control model. For example in a "Leave Processing Application" we can have a function for an employee to submit a leave application. We use function level access control model to authorise the authenticated users to access functions. Function level Access model defines the function of the applications and the roles that can access the functions.

Navigation Model

Navigation model is the mechanism that authenticated users can use to access the authorized functions. We have identified two types of functions: State independent functions and State dependent functions. In the leave processing system the functions like "apply for leave", "view leave history, which are always available to authorized users, are examples of state independent functions. These functions are independent from the state of any processes. Once a user log in, user will be provided with a menu to access state independent functions based on the navigation model. On the other hand, the "approve leave application" and "process leave application" are examples of state dependant functions. These functions are available to users only if a leave form is waiting for approval or processing for that particular user at that time. We have a menu "My Tasks" with links to access such functions in all applications at one place. Navigation model defines the menu link and navigation type for functions.

Common Business Object Model

We need to share data between applications. Common Business Object model defines the shared data in the web business system. For example, business objects such as 'employee', 'Products', etc. which are used in many applications are kept at Shell level. In other words the corporate data repository is managed in the shell. Business objects define attributes and relationships between the objects.

Application Level

As mentioned earlier an application consists of many functions. Therefore, application consists of models which support many functions. The Application level of the meta-model is shown in figure 3. Application inherits the function level access control, common business objects and navigation models from the shell level. It consists of workflow model, instance level access model and application specific object models.

Workflow Model

In a web application, the business rules govern what happens next when a function is performed. In the leave application example when an employee submits a leave form a link to access the function to approve the leave should become visible to the Manager. Conditional rules can also be associated to

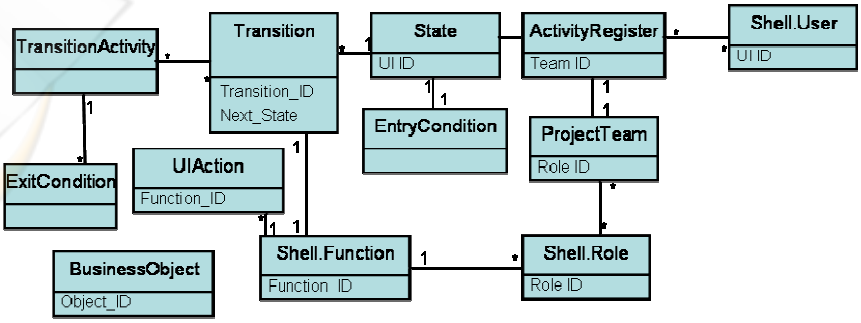


Figure 3: Application Level of the meta-model.

the flow. For example, we have a rule to say leave applications for a period of more than 3 months should be approved by Director of the company. Once it is approved by head of the department it should get directed to the Director instead of human resource division. We define the flow at the application level in the workflow model. Workflow model defines the state, entry condition, exit condition, transition, transition activity, exit condition. Transition also associated with a function and UI action.

Instance Level Access Model
When the business objects are accessed through the functions there could be rules specifying who can access what instances of a Business object. For example, if the organisation has a Sales division, Production division, Accounting Division, etc., then it may be necessary to specify that the leave form from an employee in a particular division needs to be approved by the Manger of that division. By applying the instance level access rules we can identify the 'project team' that participates in actions in functions in the workflow. Instance level access control model defines the project team, the members of the team and the activities waiting for their action at a given time.

Application Specific Business Objects
The Application may have business objects specific to the application. Examples of application specific business objects are reference data objects used in an application such as leave types. That means these data may use only within the application namespace. However, if the user wants he can store them at shell level.

Function Level
Functions are the way of performing the actions in an application. The UI is the mechanism to perform the functions. UIs, Business rules associate with UI elements in input mode and UI Action models are defined at the function level. The function level meta-model is shown in figure 4.

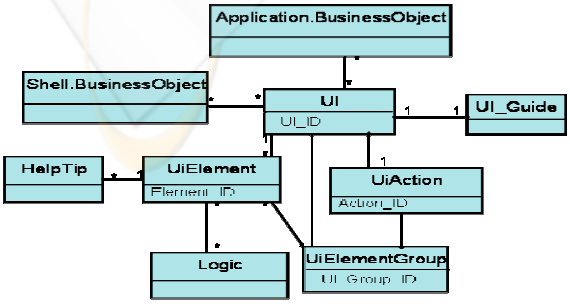


Figure 4: Function level of the meta-model.

User Interface Model
The user interface model defines the interfaces used in functions of the application. User Interface Model consists of UI guide, `UIElementGroup`, and or `UIElements`, and UI Actions. UI elements can be in input mode or output mode. In a form we have UI elements in Input mode. In a report we have UI elements in output mode. `UIGuide` provide the guidelines to use the particular interface. For example, in a form interface, the guidelines can help the users to understand the purpose of the form UI. In a report interface, the guidelines can help the users to interpret the report properly. UI is associated with a business object which binds with the interface. For a UI element in input mode, we can also have associated help tip. Help tips help users to fill the values of the form UI element correctly. Sometimes, it is required to logically group `UIElements`. For example a product order may include more than one product. The data required for each product order can consist of quantity of product and price of product. Thus, product details can be in a `UIElement` group of the order UI. UI Actions such as add product, amend product actions can be associated with that `UIElement` group. Order UI Model can have UI Action to process the order.

Logic Model
At function level we model two types of business rules. One type is the business rules used to derive new object attribute values based on existing object attribute values. For example an organisation might give discounts based on quantity purchased; such as 5%, 10% and 15% for 10, 100 and 1000 items purchased. Thus we can derive new object attribute total cost based on the base price, quantity purchased and applicable discount. The other category is the validation rules applied over values of form field in a user interface. Example for a validation rule is quantity must be a number.

UI Action Model
The user actions in UI model also modeled at function level. Example activities that can happen in an action model are state updates in the workflow or updates to a business object.

4 COMPONENT BASED E-APPLICATION DEVELOPMENT AND DEPLOYMENT SHELL

The logical architecture of a CBEADS© Application is created based on MVC Architecture pattern introduced in 1979 by Reenskaug (Reenskaug, 1979). It is shown in figure 5. The information flow between controller and application is described below:

- Controller receives the user actions.
- Controller checks the access rights with the Access Control and authorise access.
- Controller invokes the Application Function.
- Application function then retrieves or modifies the Data.
- Application sends the view information to the View Generator.
- View Generator applies the presentation information to the view and sends the web UI to the user's browser.

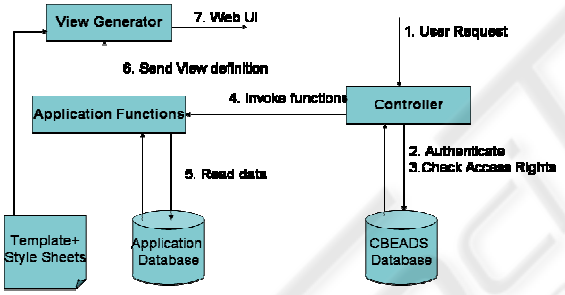


Figure 5: Logical Architecture of a CBEADS© Application.

5 RELATED WORK

Several different meta-models exist for web applications based on Meta Object facility and UML profiles. These meta-models want to achieve interoperability and standardized the web models. One such model is UWE meta model (Kraus and Koch, 2003). It is designed as an extension based on Meta Object Facility (MOF 1.4). The objective of UWE meta-model is to provide a common meta-model for the web application domain, which will support all web design methodologies. W2000 (Luciano et al., 2005), a successor of HDM (Garzotto et al., 1993), focus on model semantics and transformation rules to achieve consistency between models. Muller et al. (Muller et al., 2005) present a model-driven design and development approach with the Netsilon tool. The tool is based on a meta-model specified with MOF 1.4 and the Xion action language. Recently another two meta models (Schauerhuber, 2006), (Nathalie et al., 2006) based on MOF and UML 2 profiles are presented for WebML design methodology with the objective of interoperability.

All these meta-models of web applications are towards the precise definition of the semantics of existing web models. They are committed to be MDE compliant and achieve interoperability. However, they are more focused towards generating web applications from models. Our work is complementary to existing web meta-models, in that we propose a meta-model with new semantics which helps to effectively involve end users in development.

6 CONCLUSION

In this paper we have presented a hierarchical meta-model enabling business end-user to develop web applications. Most importantly we have addressed issues when using meta-model based approach to web application development for a business organization. This will help end-users to develop their web applications in an efficient and effective way.

REFERENCES

Arunatileka, S. and A. Ginige. *Applying Seven E's in eTransformation to Manufacturing Sector*. in *eChallenges*. 2004.

Ginige, A. *From eTransformation to eCollaboration: Issues and Solutions*. in *2nd International Conference on Information Management and Business (IMB 2006)*. 2006. Sydney, Australia.

Ginige, J., A., B. De Silva, and A. Ginige. *Towards End User Development of Web Applications for SMEs Using a Component Based Approach*. in *International Conference on Web Engineering ICWE05*. 2005. Australia.

Epner, M., *Poor Project Management Number-One Problem of Outsourced E-Projects*, in *Research Briefs, Cutter Consortium,* 2000.

Ginige, A. and B. De Silva. *CBEADS: A framework to support Meta-Design Paradigm*. in *3rd International Conference on Universal Access in Human-Computer Interaction (UAHCI07)*. 2007. China.

Costabile, M., F., et al. *A meta-design approach to End-User Development*. in *VL/HCC05*. 2005.

Fischer, G. and E. Giaccardi, *A framework for the future of end user development*, in *End User Development: Empowering People to flexibly Employ Advanced Information and Communication Technology*, H. Lieberman, F. Paterno, and V. Wulf, Editors. 2005, Kluwer Academic Publishers.

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

Rode, J., M.B. Rosson, and M.A. Perez-Quinones. *End-Users' Mental Models of Concepts Critical to Web Application Development*. in *2004 IEEE Symposium on Visual Languages and Human Centric Computing (VLHCC'04)*. 2004. Roma, Italy: IEEE Computer Society.

Reenskaug, T. (1979) *Model-View-Controllers*. Technical Report. Vol.2. scanned copy at http://heim.ifi.uio.no/ ~trygver/mvc/index.html.

Kraus, A. and N. Koch, *A Metamodel for UWE*. 2003, Ludwig-Maximilians-Universität München.

Luciano, B., C. Sebastiano, and M. Luca, *First experiences on constraining consistency and adaptivity of W2000 models*, in *Proceedings of the 2005 ACM symposium on Applied computing*. 2005, ACM Press: Santa Fe, New Mexico.

Garzotto, F., P. Paolini, and D. Schwabe, *HDM — A Model-Based Approach to Hypertext Application Design.* ACM Transactions on Information Systems (TOIS), 1993. 11(1): p. 1-26.

Muller, P. and et.al., *Platform independent Web application modeling and development with Netsilon.* Software & System Modeling, 2005. 4(4): p. 424-442.

Schauerhuber, A. *Bridging existing Web Modeling Languages to Model-Driven Engineering: A Metamodel for WebML*. in *ICWE 2006*. 2006.

Nathalie, M., F. Piero, and V. Antonio, A UML 2.0 profile for WebML modeling, in Workshop proceedings of the sixth international conference on Web engineering. 2006, ACM Press: Palo Alto, California.