CBEADS©: A Framework to Support Meta-design Paradigm

Athula Ginige and Buddhima De Silva

University of Western Sydney, Locked Bag 1797, Penrith South DC, 1719, NSW, Australia
a.ginige@uws.edu.au, bdesilva@scm.uws.edu.au

Abstract. We have developed a meta-model for Business applications. To generate applications using this meta-model we created a **C**omponent **B**ased **E**Business **A**pplication **D**evelopment and **D**eployment **S**hell; CBEADS©. The meta-model we created was based on three abstraction levels: Shell, Applications and Functions. The Shell provides the functionality common to any Web-based Business Application, and a set of configurable components and tools to create functions that are specific to an application. By using CBEADS© we can rapidly develop Web-based Business Applications by creating instances of the meta-model based on the Meta Design Paradigm. The key aspect that underpinned this research work was the viewpoint that "software is a medium to capture knowledge rather than a product". The developer's knowledge is embedded into the shell and the tools. The End-user's knowledge is used to populate instances of the meta-model from which applications are generated within CBEADS©.

1 Introduction

The AeIMS research group at the University of Western Sydney has been working with Small to Medium Enterprises (SMEs) in the Western Sydney region to investigate how Information and Communication Technologies (ICT) can be used to enhance their business processes to become competitive in a global economy [1, 2]. In this research the challenge was to find a way to develop web-based business applications that can evolve with changing business needs[3]. This is to support the co-evolution of the web application as identified by Costabile et. al. [4]. Also, we had to develop these applications rapidly and in a cost-effective manner [5]. The development approach should also reduce the gap between what the users actually want and what is being implemented in terms of functionality [6].

To address the above-mentioned issues it is very important to empower end users to become involved in the original application design during design time and be able to modify the application as a result of evolving requirements during the use time [7]. If we are to empower end users to actively participate during design time and be able to modify the application during use time rather than developing a specific application, we need to provide a set of tools and a framework that they can use to develop and change the application in response to changing needs. Rather than

C. Stephanidis (Ed.): Universal Access in HCI, Part I, HCII 2007, LNCS 4554, pp. 107–116, 2007.

developing "the application" we need to develop a Meta Model of the Application and a set of tools within which the End Users can create the applications that they want.

We define creating an application based on a Meta Model as the Meta-Design Paradigm. One can therefore view the application that was built as an instance of the Meta-Model. The amount of computer domain knowledge required by the people developing the application can be greatly minimised by developing appropriate meta-models to conceptualise the applications users want, and by providing tools to support the creation of the applications as an instance of the meta-model. This can eventually lead to end users being able to develop complete applications.

In this paper we present the meta-model of web application that we have developed for web-based business applications. We have also developed a component-based architecture to implement this meta-model. Based on this architecture we created the **C**omponent **B**ased **E**Business **A**pplication **D**evelopment and Deployment **S**hell, CBEADS©, to support the meta-design paradigm [3, 5, 8].

2 Related Work

The Meta-Design paradigm in the context of End User Development was conceptualised by Fisher G.[7, 9] to address the need to accommodate evolution in Information Systems. They have proposed the following definition [7]: "meta-design characterizes objectives, techniques, and processes for creating new media and environments allowing 'owners of problems' (that is, end-users) to act as designers. A fundamental objective of meta-design is to create socio-technical environments that empower users to engage actively in the continuous development of systems rather than being restricted to the use of existing systems". The *(SER)* process model, the *seeding, evolutionary growth*, and *reseeding,* support meta-design. The SER model encourages designers to conceptualize their activity as meta-design, thereby supporting users as designers rather than restricting them to passive consumers [7, 9].

The Software Shaping Workshop (SSW)[10, 11] methodology is a meta-design approach to develop interactive systems. SSW exploits the metaphor of the artisan workshop, where an artisan finds all and only the tools necessary to carry out her/his activities. The development team including designers and end-user representatives are supported in their reasoning on software design and development by software environments tailored to their needs, notations and experience. They can also exchange among them and test the results of these activities to converge to a common design.

We reviewed the existing component-based development approaches in relation to the support these provide for meta-design paradigm. The component-based approaches such as Catalysts [12] and UML components [13] focus on specifying components required in an application, extending UML to design the components. Web Composition Modeling Language (WCML)[14] is another attempt to specify the individual components based on patterns by extending OOHDM. Li [15] has proposed a content component model within which a content component is regarded as an independent process unit performing necessary content organizing, processing and presenting functions. CoOWA[16] is another component-based approach to develop web applications. CoOWA has two different types of components, called

client and server. The server components handle their own functionality and communicate with other components via interfaces. These component-based approaches have been developed with the objective to reduce development time through reuse, and thereby reduce the cost. They are not specifically designed to support the evolutionary growth or changes at runtime.

Won, M. et al.[17] have introduced a customizable approach with two-dimensional and three-dimensional customizable environments based on a component model called Flexibean, which can be used effectively by end users. Integrity checks are built into the customizable environment, and customizable artifacts can be exchanged through a shared repository. This approach can help end-users to use the meta-design based on a meta-model effectively and efficiently.

3 Meta Model for Web-Based Business 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.

To manage the complexity we developed the meta-model at three levels of hierarchical abstraction called Shell, Application and Function as shown in figure 1. The Shell provides the common functionality required for any application. The Application provides a set of functions required to perform a business process. Functions provide views or user interfaces required to perform actions in a business process.

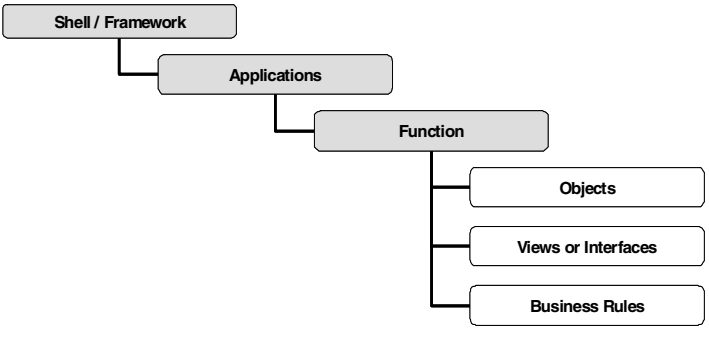


Fig. 1. Hierarchy of Abstraction Levels in Business Application Meta-Model

3.1 Shell Level

We analysed many business applications to identify features that are common to any application. In every application there are users who need to be given the authority to perform various actions. For example in a “Leave Processing Application” we can

have a function for an employee to submit a leave form. The manager needs a function to approve this and forward it to a staff member in the Human Resource division. To meet this requirement we require an authentication and access control mechanism. We also require a mechanism to generate "personalised menus" for each user when they log into the system to show what actions they are authorised to perform and a way to execute the corresponding function to perform the action. As access control and personalised menu generation are common to all applications we decided to provide these modules at the Shell level.

We also need databases to store the business objects. We debated whether to keep databases as part of an application or keep them at the Shell level so that data can be shared across many applications. As we saw the importance of being able to share the data across many applications we decided to keep the databases that are used to store Business objects such as 'Employee', 'Products' etc. that are common to many applications at the Shell level and any database that is used to store Business Objects that are specific to an Application at the Application level.

3.2 Application Level

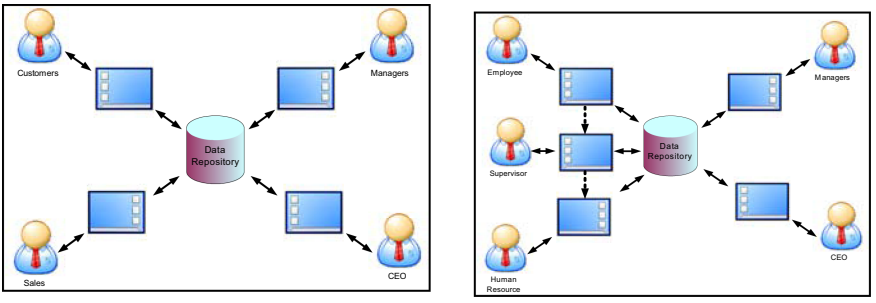
As mentioned earlier, an application consists of many functions. In an application, some of the functions can be performed by a specific user at any time. For example, an employee can apply for leave at any time, but the function for the Manager to approve the leave application can be executed only when there is a pending application. Thus we need a Menu to provide access to functions that can only be executed depending on the state of a process. We call these functions state-dependent functions. As an application can have many state-dependent functions, we have modeled and implemented this at the Application Level. In our implementation we call this state-dependent menu "My Tasks".

If necessary, applications can have their own presentation style rather than the default style provided by the Shell. In addition, an Application may have its own databases.

3.3 Function Level

A simple business application such as maintaining a product catalogue can be modeled as a set of functions that provide different views, such as an interface to view the products, update or add new products. Different users can be given access to different functions, e.g. the product manager of the company can access functions that will allow the product manager to create and update product information. A public user can be assigned the function that will display the product catalogue. These functions are linked to the Business Object "product catalogue" stored in a data repository. Such a model of a business application is shown in figure 2a.

In addition, more complex applications will require sequencing of different views. For this we need business rules to govern what happens after an action is performed. In a leave processing application, when an employee submits a leave form, a link to the function to approve the application should become visible to the Manager. Thus we need a way to define a flow. A model of such an application is shown in figure 2b.



(a) **(b)**

Fig. 2. (a) Application model without sequence-ing of Interfaces. **(b)** Application model with sequencing of Interfaces.

There could also be rules to specify who can access what instances of a Business object. For example, if the Organisation has a Sales Division, Production Division, and Accounts Division then it may be necessary to specify that the leave application from an employee in a particular division needs to be approved by the Manger of that division.

Another category of business rules that needs to be specified is how new information can be derived based on existing information. For example, an organisation might give discounts based on quantity purchased; such as 5%, 10% and 15% discounts for 10, 100 and 1000 items respectively. When deriving the total cost we need to base this on the base price, quantity purchased and applicable discount.

The elements of the three levels of abstractions of the meta-model are shown in Table 1 below.

Table 1. Three Levels of Abstractions of the Meta Model of Web Application

Abstraction	Description
Shell (Framework)	Has Applications Has Databases Has an Authentication Mechanism Has an Access Control Mechanism Has Session Management Mechanism Has a Menu to provide access to 'functions for authorised users Has User and Access Control Management Application Has a Function to create New Functions and Applications Has a Directory Structure for application related physical Artifacts Has a Default Presentation Style
Application	Has Functions May have many Presentation Styles Has a Menu to provide access to state depended functions May have Databases
Function	Has Objects Has interfaces (Views) to perform actions Has Business Rules

4 CBEADS© Architecture

The high-level architecture of CBEADS© is shown in figure 3. The Shell itself is made of components, which can be grouped into two major sub-systems. The first sub-system is CORE CBEADS that provides the overall framework within which different business applications can be developed and deployed. All the aspects of the meta-model corresponding to the Shell level are implemented in the CORE CBEADS sub-system. It consists of a security module, system components, a system database and a workflow component. The second sub-system consists of various applications deployed within the shell. The detailed application architecture was created based on the MVC architecture pattern introduced in 1979 by Reenskaug.

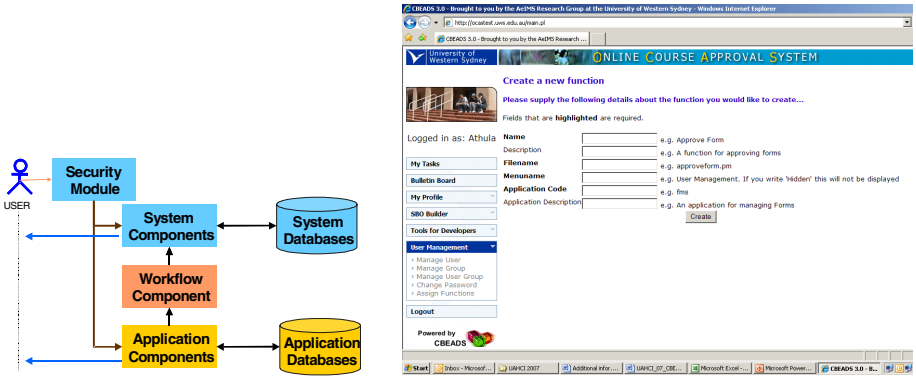


Fig. 3. Overall CBEADS Architecture and User Interface

All user interactions coming from Web browsers on users' computers are passed through a security module that performs authentication and checks what applications and functions the user is permitted to access. Firstly, a user needs to login to the system. After verifying the user, the user is given a personalised home page. This page displays the applications and the functions that the user is permitted to access. The user can activate these functions by clicking on the links. As an added security measure, before the requested function is activated the security module again checks whether the user is permitted to use the requested function.

Within the system components there is the User and Access Control Management Application. This application has functions to create user groups, allocate different functions within other applications to user groups, and create users and allocate them to user groups. This user and access control Management Application can be considered as a configurable component to suit the needs of the Business applications.

There is a special function among CBEADS© system components that can be used to create new functions, enabling CBEADS© to grow. Created functions can be grouped to form new applications. This function on the CBEADS© user interface is shown in figure 3.

Using the function to create new functions in CORE CBEADS, we have developed set of tools to facilitate the development of applications. These tools range from simple code editors and tools to create databases, to SMART tools for generating Smart Business Objects (SBO) [18], Workflow Engines (WFE) and State Dependent

Access control (VTMAC). Using an English-like language, users can specify the object model using the Smart Business Object generation tool, which then creates the business objects required for the application. We have embedded some of the computer domain-specific knowledge into these SMART tools.

5 Developing an Application Using CBEADS©

In this section we show how a leave processing application would be modeled based on the meta-model and implemented within CBEADS©. The Use Case Diagram for the leave processing application is shown in figure 4. Based on the use case diagram we can identify four functions required for the Leave Processing System which are given in Table 2.

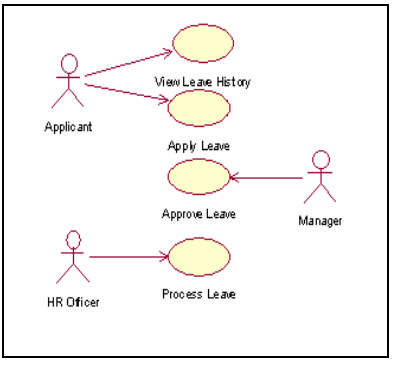


Fig. 4. Use Case Diagram for Leave Processing System .Example

Table 2. Functions in Leave Application System

Function	Type of Function	Actor
Apply Leave	Static	Employee
View Leave History	Static	Employee
Approve Leave	State depended	Division Manager
Process Leave	State depended	HR Manager

Using the SBO builder tool in CBEADS© we then specify the data objects required for the Leave Application using the SBOML language [18] as shown in figure 5. The SBO builder then generates the Business Objects for the application.

In leavesystem, leave has date, applicant, from (date), to (date), type (which could be sick or annual or no pay), status (which could be approve or reject), many approval (has date, approved by, comment)

Fig. 5. SBOML Specification of the Leave Object

Next, we develop the interfaces required for the four functions using the UI Generator shown in figure 6.

Fig. 6. Generating UIs for leave processing application using SBO UI Generator

Next we need to sequence the views. We specify the sequence of views or the workflow in a spreadsheet, upload to CBEADS© as a tab delimited text file and generate the required workflow definition file using a parser available within CBEADS©. For this application we modeled the workflow as shown in Table 3.

Table 3. State Table for Leave processing workflow

Current State	Actor	Function	Buttons	Do Action	Next State
1	Employee	Apply_Leave	Submit	Email Division_Manager	2
2	Division_Manager	Approve_Leave	Approve	Email HR_Manager, Employee	3
			Reject	Email Employee	4
3	HR_Manager	Process_Leave	Processed	Email Employee, Division_Manager	4
4	HR_Manager	View_Applications	-		4

Once we have defined the workflow, we can specify the instance level access rules by defining Departments and their Managers. Finally, reports such as the Leave History, which is based on existing data, can be generated using the UI Generator. Using the tools provided within CBEADS©, this whole application can be generated within an hour without writing any code or SQL queries.

Over time, roles can change in such an application, e.g. a new manager being employed. Through the "Access Control and User Management Application" in the Shell the new person can be assigned as a Manager. Changes to the Business Object may also occur, e.g. there may be a need to add a new leave type called "Paternity Leave". This can easily be done through SBO Builder. You can easily add an attribute to a Business Object; deleting attributes however is a non-trivial task. Another type of

change could occur at the business process level, e.g. a new policy specifying that all leave greater than five days requires the CEO's approval. In the Excel spreadsheet that we use to define the process flow, a new state needs to be added to include the CEO's approval and a new workflow definition file needs to be generated using the tools provided.

Thus Business Applications developed using CBEADS© can evolve with changing Business needs.

6 Conclusion and Future Work

In this paper we have presented a framework to support the meta-design paradigm within which users can develop and deploy evolutionary web-based applications. We adapted the philosophy that "software is a medium to capture knowledge than a product". To implement this philosophy we developed a meta-model to represent Business Applications to capture the knowledge of computer domain experts. This knowledge was then embedded into a component-based shell, as well as tools that can be used to generate web-based business applications. Users can create a Business Application by instantiating an instance in the meta-model. The knowledge captured from the users during this process is used to configure the relevant components in the shell or to generate new components using the tools. To a limited extent we have experimented with the meta-design paradigm using CBEADS© framework with SMEs and obtained successful initial results. Based on the results of these experiments we have plans to refine and extend the tools.

Acknowledgements. Since 1998 when CBEASD© was first created it has evolved considerably. From time to time many research students in the AeIMS research group have made valuable contributions to the evolution of CBEADS©. The authors would like to acknowledge the support of these researchers, particularly; Ioakim (Makis) Marmaridis who designed the session management module, security subsystem and implemented the workflow engine, Anupama Ginige who developed the workflow modeling method for end users and Xufeng (Danny) Liang who developed the Smart Business Objects.

References

1. Ginige, A.: Collaborating to Win - Creating an Effective Virtual Organisation. In International Workshop on Business and Information, Taipei, Taiwan: Shih Chien University and National Taipei University (2004)
2. Ginige, A.: From eTransformation to eCollaboration: Issues and Solutions. In: 2nd International Conference on Information Management and Business, (IMB, 2006), Sydney, Australia, pp. 15–23 (2006)
3. Ginige, A.: New Paradigm for Developing Software for E-Business. In: IEEE Symposia on Human-Centric Computing Languages and Environments, Stresa, Italy, pp. 243–246. IEEE, New York (2001)

4. Costabile, M.F., Fogli, D., Marcante, A.: Supporting Interaction and Co-evolution of Users and Systems. In: Advanced Visual Interfaces, Venice, Italy, pp. 143–150. ACM, New York (2006)
5. Ginige, A.: Re Engineering Software Development Process for eBusiness Application Development. In: Fifteenth International Conference on Software Engineering and Knowledge Engineering, San Francisco Bay, USA, pp. 1–8 (2003)
6. Epner, M.: Poor Project Management Number-One Problem of Outsourced E-Projects, in Research Briefs, Cutter Consortium, Retrieved (February 10, 2007) (2000), from http://www.cutter.com/research/2000/crb001107.html
7. Fischer, G., et al.: Meta Design: A Manifesto for End -User Development. Communications of the ACM 47(9), 33–37 (2004)
8. Ginige, A.: New Paradigm for Developing Evolutionary Software to Support E-Business. In: Chang, S.K. (ed.) Handbook of Software Engineering and Knowledge Engineering, pp. 711–725. World Scientific, Singapore (2002)
9. Fischer, G., Giaccardi, E.: Meta Design: A framework for the future of end user development. In: Lieberman, H., Paterno, F., Wulf, V. (eds.) End User Development: Empowering People to flexibly Employ Advanced Information and Communication Technology, pp. 427–457. Springer, Heidelberg (2006)
10. Costabile, M.F, et al.: Building Environments for End User Development and Tailoring. In: IEEE Symposia on Human Centric Computing Languages and Environmnets,VL/HCC03, Auckland, pp. 31–38. IEEE, New York (2003)
11. Costabile, M.F., et al.: A meta-design approach to End-User Development. In: IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC'05, Dallas, Texas, USA, pp. 308–310, IEEE, New York (2005)
12. D'Souza, Francis, D.: Objects, components, and frameworks with UML: the CATALYSIS approach. Addison-Wesley, London, UK (1999)
13. Cheeseman, J.: UML components: a simple process for specifying component-based software. Addison-Wesley, Reading (2001)
14. Gellersen, H.W., et al.: Patterns and Components: Capturing the Lasting admidst the Changes. In Active Web Conference, UK, UK retrieved (February 10, 2007) (1999), from http://www.visualize.uk.com/conf/activeweb/proceed/pap20/
15. Li, Q., Chen, J., Chen, P.: Developing an E-Commerce Application by Using Content Component Model. In: 36th International Conference on Technology of Object-Oriented Languages and Systems TOOLS-Asia'00) Xi'an, China, p. 275, IEEE, New York (2000)
16. Zhao, W., Chen, J.: CoOWA: A Component Oriented Web Application Model. In: 31st International Conference on Technology of Object-Oriented Language and Systems, Nanjing, China, pp. 191–199, IEEE, New York (1999)
17. Won, M., Stiemerling, O., Wulf, V.: Component based Approach to Tailorable Systems. In: Lieberman, H., Paterno, F., Wulf, V. (eds.) End-user Development, pp. 115–141. Springer, Heidelberg (2006)
18. Liang, X., Ginige, A.: Smart Business Objects: A new Approach to Model Business Objects for Web Applications. In: 1st International Conference on Software and Data Technologies, Setubal, Portugal, pp. 30–39 (2006)