AN INTERNET ACCOUNTING SYSTEM
A large scale software system development using model driven architecture

Kenji Ohmori
Faculty of Computer and Information Sciences, Hosei University, 3-7-2 Kajino-cho, Koganei-shi, Tokyo, Japan

Keywords: Accounting System, Model Driven Architecture, Unified Modelling Process

Abstract: Software development should be changed from a handcraft industry to industrialization like manufacturing to obtain high productivity. In knowledge creating industry of software development, engineers have to concentrate on core works. Peripheral works should be avoided as much as possible. Model driven architecture helps programmers work mainly in analysis and design without considering much about implementation. Internet Accounting System, which is a standard model of enterprise systems have been developed with model driven architecture with high productivity.

1 INTRODUCTION

A big innovation is being brought about by model driven architecture (MDA) in development of a software system. Internet Accounting System described in this paper has also been developed using MDA (Mellor 2002, Raistrick 2004, Kleppe 2003, Frankel 2003). Although many accounting systems have already been developed, most of them are concentrated systems that are used by a limited number of accountants. As companies are exposed to severe competitions, it is required to sustain accurate accounting information updated by each transaction. In order to reply to such a request, accounting information is necessary stored to the accounting system where and when a transaction occurs. Since the conventional concentrated system scarcely responds to these requests, it is obliged to develop a new accounting system utilizing Internet, which includes difficult problems inherent in a distributed system such as system configuration, a user interface, and security are problems to be solved.

Internet Accounting System is uncertain in development. We can hardly figure out if its development leads to a success before and even during its development. In recent years, a rational unified process (RUP) attracts attention to development of such software. A RUP (Jacobson 1999, Kruchten 2000) shares the same concepts with knowledge management (Nonaka 1995) in business administration. Unlike the water flow model developing a system in top-down from requirements to design, implementation and evaluation, A RUP divides a system into portions. RUP begins by developing a portion with the biggest risk and it progresses gradually to development of other portions with less risks. In development of each portion, it will move from requirement specifications to design, implementation and evaluation. Although requirement specifications and design are intellectually creative activities, implementation and evaluation are labour-intensive work. If these phases are realized by automatic code generation, software development will be challenged as more intellectually creative work, which is accomplished by a small number of designers and programmers. Internet Accounting System described in this paper does not attain the level of a commercial system in error handling. However, all the required functions as an accounting system have been incorporated. The system has been developed in seven months by only single person. The successful development in the short period and by a small number of people is owed to automatic code generation in the implementation phase using MDA. We could concentrate our energies on the requirements and design phases.

2 SYSTEM OUTLINE

On-line entry at the spot is the most important requirement. It has been realized in a web based

Ohmori K. (2005).
AN INTERNET ACCOUNTING SYSTEM - A large scale software system development using model driven architecture.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 407-410
DOI: 10.5220/0002553504070410
Copyright © SciTePress

system, where each user is equipped with a personal computer connected to a web server through Internet. In this environment, a user can call a web server through browser, get an input form, and send a filled form to the server whenever a financial transaction arises at his place.

This accounting system naturally includes functions of financial accounting with a general ledger and financial statements. It also has administrative accounting with the management of sector, segment and period spending and sales.

Internet Accounting System must be enough reliable to preserve transactions safely. An application server that provides rollback functions is necessary for realizing Internet Accounting System. Moreover, a database server is also necessary to store financial and administrative accounting data and transaction records. The hardware structure is composed of three tiers of web, application and database servers.

A software subsystem has been arranged for each server. A servlet container has been provided for the web server to support sophisticated interfaces. It is supposed that many homepages be required to implement the application. We also equipped the container with a web application framework to help a programmer implement a complicated system in high productivity. An enterprise application framework has been provided for the application server. An enterprise application framework fulfilling J2EE technology is divided into two parts: web and database sides. In the web side, business logic of the application is realized by receiving requests from the servlet container and sending results to it. The database side has interface to the database server using an object-relational mapping. The web side is realized by a set of session beans and the database server is carried out by a set of entity beans. These beans are kinds of classes. Each entity bean corresponds to a table on the database server. The database server is equipped with database software.

The servlet container in this system is Tomcat and the web application framework is Struts. Struts needs three kinds of classes to implement a web application. One is a JSP, which sends a homepage to user. Another is an action program, which controls the transition of JSPs. The other is an action form to store data entered by a user. If a user pushes a submit button on the current homepage, the request

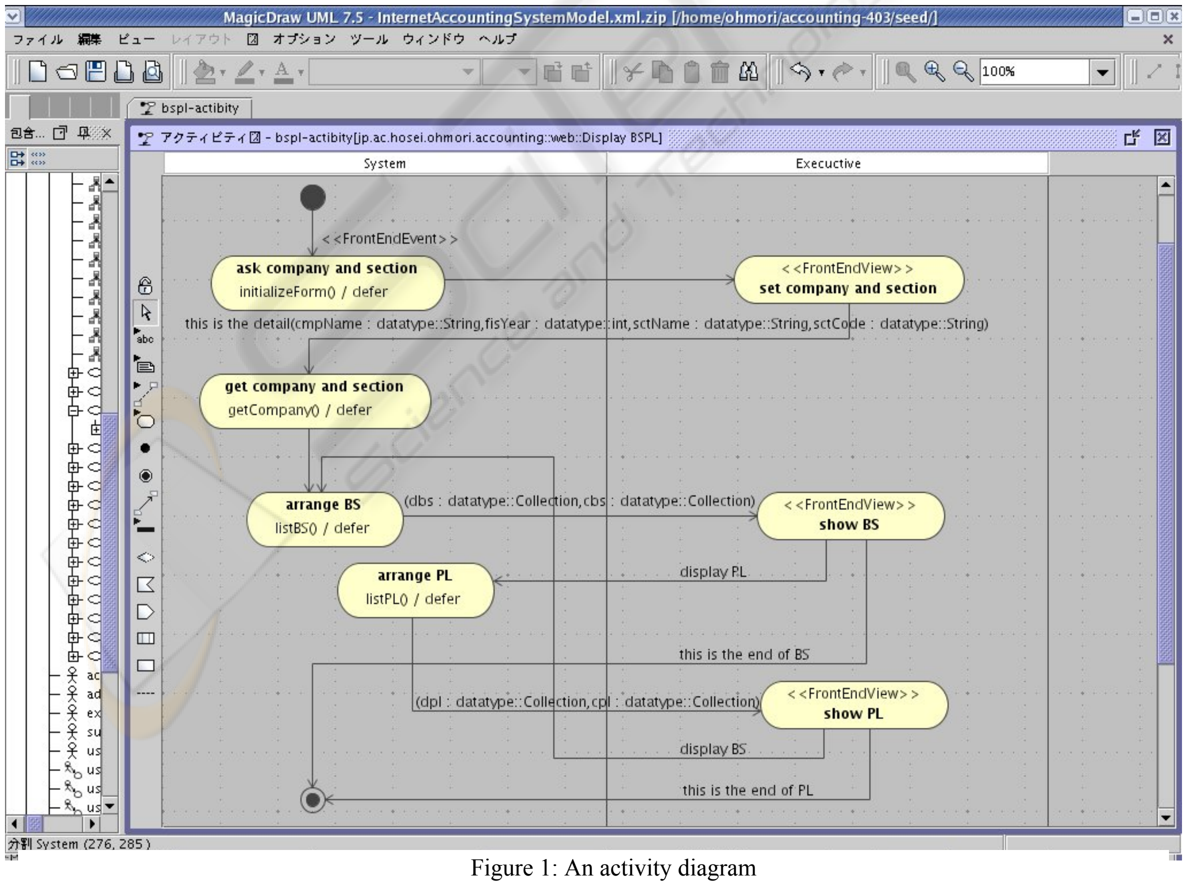
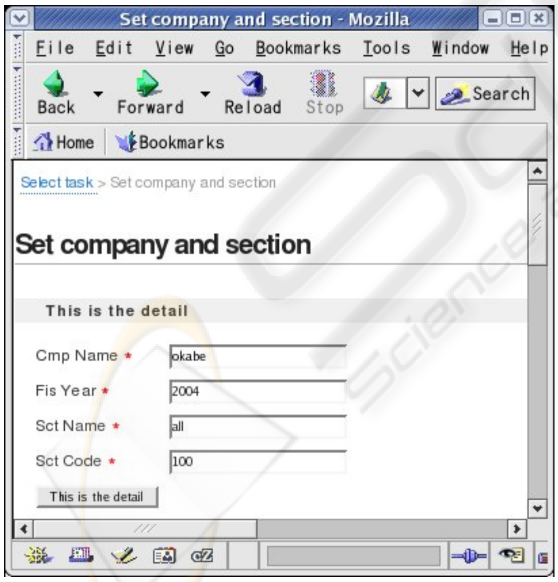

MagicDraw UML 7.5 - InternetAccountingSystemModel.xml.zip [/home/ohmori/accounting-403/seed/]
ファイル 編集 ビュー レイアウト オプション ツール ウィンドウ ヘルプ
100%
bspl-actibity
アクティビティ - bspl-actibity[jp.ac.hosei.ohmori.accounting::web::Display BSPL]
System
Executive
<<FrontEndEvent>>
ask company and section
initializeForm() / defer
<<FrontEndView>>
set company and section
this is the detail(cmpName : datatype::String,fisYear : datatype::int,sctName : datatype::String,sctCode : datatype::String)
get company and section
getCompany() / defer
arrange BS
listBS() / defer
(dbs : datatype::Collection,cbs : datatype::Collection)
<<FrontEndView>>
show BS
arrange PL
listPL() / defer
display PL
this is the end of BS
(dpl : datatype::Collection,cpl : datatype::Collection)
<<FrontEndView>>
show PL
display BS
this is the end of PL
分割 System (276, 285)

Figure 1: An activity diagram

is sent to the corresponding action program. At the same time, data entered by the user are stored in the corresponding action form. The action program gets the data from the action form, processes them and sends the result to the next JSP. It sends a new homepage to a user and the homepage waits for new entries form a user. The same process is repeated.

Most programs embedded in the web and application frameworks are not coded by programmers but generated automatically from UML diagrams. According to requirement specifications and design results, we have described three kinds of diagrams using UML. Use cases define how a user uses the application. Activity diagrams define in which order activities of a use case is realized. Class diagrams define what classes are used to realize the application. Translation software generates program codes from these diagrams. The first homepage is created from a use case. It allows to select other use cases. Each activity diagram generates program codes of JSPs, action programs and action forms. Most program codes realizing the web application are automatically generated except business login included in action programs. From class diagrams, most program codes are generated for the application server except business logic of session beans.

We used AndroMDA to generate program codes



Select task > Set company and section

Set company and section

This is the detail

Cmp Name *	okabe
Fis Year *	2004
Sct Name *	all
Sct Code *	100

This is the detail

Figure 2. A homepage.

from UML. AndroMDA generates intermediate diagrams for translation from these diagrams. XDoclet scripts are generated from intermediate diagrams. Then, XDoclet generates program codes for the web and application servers including system configuration files as well as database configuration files. VSL files in AndroMDA defines how an intermediate diagrams are translated into XDoclet scripts for each type of namespaces. The namespaces correspond to types of program codes such as control, service and value object classes. As VSL files allow customization, we added necessary functions and modified some functions.

3 IMPLEMENTATION

The programming language used in developing the system is Java so that the system can be installed on almost any kind of processors and operating systems. It can be installed on Linux, Microsoft Windows or Mac. JBoss is used as an application server to give persistent functions to an application program. It can be connected to a web server and a database server. Many database servers are connectable to JBoss through the JDBC API. Web servers are also connectable to JBoss though RMI (Java Remote Method Invocation). As the system is experimental, we are using Struts and Hypersonic SQL as a web server and a database server, which are in-house subsystems installed in JBoss. If the system becomes commercial, Apache will be used as a web server and PostgreSQL, MySQL or Oracle will be used as a database server. JBosss has two selectable subsystems to provide for an object-relational mapping between J2EE technology and a relational database. CMP (Content-Managed Persistence) is more reliable and Hibernate is more flexible. We have used Hibernate in this system, considering with future development.

An example of an activity diagram is shown in Figure 1. It is related to financial statements. The figure consists of two swim lanes. JSPs (Java serve pages) are automatically created from the right-hand side lane, action programs from the left-hand. Action forms are automatically generated from the parameters defined at arcs connecting a right lane node and a left one. The transition of nodes is automatically defined by a Struts configuration file.

The figure defines the initial node and the final one as the entrance and exit of this activity, respectively. The next node (which is named as "ask company and section") connected to the initial node sets up initial conditions. Then, the process moves from this node to the right-hand side node (which is named as "set company and section"). A JSP is automatically generated from this node. It sends a homepage as shown in Figure 2 to a user display.

Table 1: Results

		Total	Automated	Half-automated	Hand writing
WEB	JSPs	348	348		
	Action Programs	171	151	20	
	Action Forms	242	242		
	Configuration	5	5		
EJB	Session Beans	78	45	34	
	Entity Beans	42	42		
	Schema	6	6		
	Web Service	13	13		
Common	DTO	27	27		
	Exception	18	18		
Class Diagram	WEB	45			45
	EJB	48			48
	DTO	27			27
Activity Diagram		23			23
Use Case Diagram		23			23
Total		1116	897	54	166

4 EVALUATION

Table 1 shows ratios of automatically generated classes. The number of classes in the class diagrams becomes 120 in total. We have 23 activity diagrams. As each use case diagram corresponds to one activity diagram, we also have 23 use case diagrams. From these diagrams, the number of web related classes are 761. Among these, 741 classes are fully automatically generated. The number of EJB related classes except schema is 133 in total. Among these, 99 classes are fully automated. Moreover, we have 45 common programs. These are fully automated. As consequence, the automation ratio becomes 94.2% (885/939).

5 CONCLUSIONS

Internet Accounting System has been developed using new technologies including MDA and RUP. The development started at the beginning of May, when new milestone version of AndroMDA was announced. As AndroMDA is still on the course of development, we used nightly build versions. We encountered to develop our application on an unsteady and changeable platform. It results in good experience of verifying MDA platform independence. This experience shows that MDA is helpful in a developing environment.

This milestone uses activity diagrams for the first time to generate web related classes fully. As implementation of web related control programs is very complicated and tiresome, automatic code generation is very helpful. It shows a new way to use MDA.

It becomes clear that programmers can concentrate only on business logic since other parts are automatically generated. It leads for programmers to help develop reliable software in a short time.

REFERENCES

Philippe Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley, 2000

Ivar Jacobson, Grady Booch, James Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999

Ikujiro Nonaka and Hirotaka Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford University Press, 1995

Stephan Mellor, Marc Balcer, and Marc J. Balcer, Executable UML: A Foundation for Model Driven Architecture, Addison-Wesley, 2002

Chris Raistrick, Paul Francis, and John Wright, Model Driven Architecture with Executable UML, Cambridge University Press, 2004

Anneke Kleppe, Jos Warmer, and Wim Bast, MDA Explained: The Model Driven Architecture--Practice and Promise, Addison-Wesley, 2003

David S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, Wiley, 2003