Design of Object-Oriented Methodology for Client-Server Information Systems Domain

Yim, Hongsoon (KAIST Center for Advanced Information System)

Kim, Jongwoo (Chungnam National University, Department of Statistics)

Park, Sungjoo (KAIST Graduate School of Management)

Abstract

Current object-oriented methodologies are inappropriate for development of client-server information system not because they are wrong methods, but because they are too general to apply to every different problem domain and span of development lifecycle. In this paper, we propose a tailored object-oriented methodology that is applicable to system development for specific domain, client-server information system, in practice. The proposed methodology consists of modeling language and development process that reflect characteristics of client-server information system such as business modeling, client-server application partitioning, modeling of collaboration among computing components.

1. Introduction

While the popularity of client-server system is increasing in trends, practical development support of methodology has fallen behind the trend of increasing popularity. With the growth of visual programming tools, form-centered design methods are used widely for the development of client-server information system, but they bring up new problems such as lack of modularity, extensibility, and maintainability [10]. Due to the fact that object-oriented methodologies are known to deliver solutions for those weaknesses, they are expected to be widely and generally accepted in developing client-server information system [6,10,13].

Although dozens of object-oriented methodologies have been published, their actual utilization rate seems to be considerably low [13]. The main contribution of this failure lies in that current object-oriented methodologies are too general to apply to every different problem domain and span of development lifecycle. For the effectiveness and the productivity of client-server information development reach their maximum, existing general-purpose object-oriented methodology should be tailored in domain-specific way

In this paper, we propose a tailored object-oriented methodology for the development of client-server information system and introduce some issues on the tailoring methodology. The proposed methodology includes modeling language and development process. Based on Unified Modeling Language (UML) [2], modeling language is designed to be applicable for business applications, client-server applications, and visual programming tools in object-oriented way. Through the extension mechanism of UML such as stereotypes, tagged values, and constraints, modeling language is tailored without loss of original semantics in UML. The development process is derived from ISO 12207 [5] and extended to include business analysis phase and physical design phase. Business analysis is extended requirement analysis such that the environment of business is analyzed, evaluated, and designed. Physical design is a phase for activities that describe a physical solution along implementation environment. In particular,

through pre-defined detailed outputs are used as a milestone of process, uncertainties of development process such as progress, increment, iteration of process could be minimized. Also, the development process includes rich guidelines and issues that are related to tasks.

2. Design of domain-specific methodology

2.1 Background and scope

Most existing object-oriented methodologies are designed with general-purpose to be applicable every different domains, so they have vast modeling artifacts that are elaborate and accurate. In result, complete compliance of them does not only require excessive effort, but also cause a drop of productivity. Actual strategy is a tailoring of existing methodologies in design of proper object-oriented methodology for own environment, because a creation of new methodology does not require great cost, but also not guarantee successful result.

The scope of this paper is about methodological components such as modeling language and development process rather than managerial components such as labor, cost, and time.

2.2 Analysis of development environment components

Analysis of development environment is for identifying and reflecting characteristics of domain in object-oriented way to decrease the complexity of applying a general-purpose methodology and to represent target domain effectively. The tailored methodology that reflects characteristics of domain makes productivity be increased by intensifying power of domain representation and improving comprehension of development participants. Analysis of development environment includes analysis of implementation environment such as implementation tools. Although an object-oriented methodology is applied for analysis and

design, object-oriented implementation tools are not always used in every case of implementation. Actually, visual programming tools are widely used for the development of client-server information system but supporting methods and activities for visual programming tools are not enough. The reflecting implementation environment improves executability and understandability of design model.

The target domain of proposed methodology is a business application under client-server environment. Also, the target implementation environment is in case that application is developed with visual programming tool and relational database. In this domain, components to reflect characteristics are as follows.

- Methods and activities for business modeling
- Integration between business model and analysis/design model
- Abstract mapping computing components of client-server system to objects
- Effective representation of collaboration among computing components
- Level of abstraction for GUI objects
- Application partitioning, allocation, and architecture in client-server system
- Mapping rules for conversion from design objects to implementation objects

Based on above characteristics, a modeling language and a development process are described in next chapter sequentially.

3. Design of the modeling language

3.1 Design and structure of modeling language

A modeling language is defined as a language for specifying, visualizing, constructing, and

documenting the artifacts of software system. It includes notations, semantics, and diagrams as components of the language. Based on UML, the proposed modeling language is designed in modification and extension manners rather than creation as new modeling language.

The core of object-oriented development is identifying and specifying objects that are valid and stable. In fact, although there are various models such as class model, interaction model, and use case model, information from those models is integrated to class description. Therefore, without loss of semantics in original model, modification and extension of existing modeling languages should be done to support class description. Based on this principle, the proposed modeling language is designed in three dimensions as follows

- Support object identification
- Support object design
- Support object verification

To support object identification, various stereotypes, which are meta-classifications of modeling components in UML, are pre-defined for each models and development phases. Stereotypes are effective for supporting object identification because they make analysts concentrate on narrow scope, intensify comprehension of domain, and fit level of abstraction. For business model, stereotypes include actor, subsystem, entity, etc. For analysis and design model, there are domain objects, control objects, form objects, library objects, utility objects, etc.

To support object design, diagrams are modified and extended without change of notations. In UML, there are eight kinds of diagrams: use case diagram, class diagram, statechart diagram activity diagram, sequence diagram, collaboration diagram, component diagram, and deployment diagram. In the proposed modeling language, all of those diagrams could be used but usage of some diagrams is limited. For example, limited usage of statechart diagram is

suggested not because it is useless model, but because timing control is not important in domain information systems. Excluding activity diagram and interaction diagram, other models are slightly changed such that represent stereotypes.

For business model, activity and use case diagrams are modified. In activity diagram, activities are arranged in time-order, represented with actor, and abstracted to use case after business model. Also, relationships between activities have tagged values such as delivery time, document, and throughput time. Interaction model is designed to make in class-level then it keeps consistency with class diagram. Also, In interaction model, messages are extended such that a request and the response message are represented together in one request message. In collaboration among objects, when client object requests some services to server object, client object should wait for the response of server object to process the next work. By those extensions in interaction model, design of operations in server object and trace of message are supported systematically.

Run-time view is added to verify designed classes. In class model, run time view is to verify static structure among classes with instances and links. Also, in interaction model, it is to verify consistency between scenario and collaboration among classes. With instances and concrete messages, run-time view covers the difficulty that collaboration among instances from the same class is described.

3.2 Issues on design of modeling language

(1) Tailoring existing methodologies

For design of own modeling language, the development of new modeling language does not only require great effort and cost, but also does not guarantee successful design. So, based on existing methodologies, tailoring them minimizes risks from design of own methodology.

Methodologies that recently are published in the community of object-oriented methodology such as Mainstream Objects [13], UML, etc., are integrated methodologies with past methodologies and have enough rich notations and semantics to adapt for own environment.

The proposed methodology is successfully designed without loss of the semantics in modeling components of UML through extension mechanisms of UML such as stereotypes, tagged values, and constraints.

(2) Utilization of object-oriented business model

The development of information system requires extended requirement analysis such that the business environment is analyzed, evaluated, and designed from business model. Most existing business models have utilized methods from business reengineering or information engineering. Although they are effective for business analysis, they have some problems that are difficult to be integrated with system analysis/design in object-oriented way.

Using the same paradigm, utilization of object-oriented model for business model supports to integrate business analysis and system analysis/design. Recently, there are many researches for object-oriented business model but they just focus on object-oriented representation [3, 11]. For business analysis, researches for methods and processes that analyze, evaluate, and resolve business problems are necessary. In the proposed methodology, through extensions of activity and use case diagram in UML, business model is designed. Also, business model includes measures such as delivery time, number of stage, and throughput time. Measures are developed as tagged values of relationships among activities.

4. Design of the development process

4.1 Design and construction of the development process

Development process includes activities, regulations, and guidelines that occur during the system development life cycle. Even though existing object-oriented methodologies provides their own developing processes, they are too abstracted to be applied to real world. Especially, regulations and guidelines that needed for the problem area or development environment in practice are insufficiently provided.

The proposed development process is derived from ISO 1227 [5], which is a standard process software development, as its base. The development process consists of 6 phases: business analysis, systems analysis, logical system design, physical system design, implementation, and test. The business analysis as an extended requirement analysis for business application development is designed to fulfill business process reengineering which can express as-is business model and should-be business model. While system analysis is a stage in modeling the problem of target domain, system design is a stage in modeling a solution of the problem. Difference of logical design and physical design is a matter of independence with the implementation environment. For example, although the same logical design model is used, physical design models are different according to the selection of implementation tools. According to the divisions of design, additional design activities enhance executability of design model by keeping consistency of structure in design model and implementation model.

The development process of proposed methodology consists of the activities of corresponding processes and it denotes the task that has to be performed by these activities. Each task provides its performing subjects, products, guidelines, and rules according to its role. Especially, it has been designed to minimize uncertainty of tasks by regarding the products to be a major concern. The development process includes not only the development activities such

as analysis and design but also the supporting activities such as documentation, quality assurance, and exception handling activities.

4.2 Issues on design of development process

(1) Seamless transition between business analysis and system analysis/design phases

Business analysis is performed as task analysis in order to analyze requirement of information system. Using object-oriented business model makes business analysis phase naturally combine with analysis/design phases. In this methodology, models of analysis/design are resulted from models that are made in the business analysis. Analysis and design objects are derived from stereotype classes such as actors, subsystems, entities used in business analysis phase. Use cases that are identified and modeled from activity model in business analysis are utilized as a unit of analysis and design. In particular, including stereotype classes, use case models are utilized as a seamless connector between business analysis and system analysis/design phases, because business model is extended from activity model, use case model is derived business model, and analysis/design model are derived from use case model.

(2) Seamless transition between system analysis/design and implementation phases

Even though an object-oriented development methodology is used for systems analysis and design, there are many cases that complete object oriented implementation tools are not used because of restrictions like time, resource, and capacity of implementation technology for development environment. In this case, additional methods and processes are needed for the seamless transition from design model to implementation model. For example, the task to change design model to fit implementation tool is needed if non object-oriented tools such as

visual programming tool and relational database are used as implementation tools. In this methodology, physical design phase is added as transform process to adapt the structure of implementation environment. In physical design phase, rules for mapping from design object to implementation object are documented and the architecturing of implementation object is performed. The productivity of modeling work is enhanced by minimizing excess modeling work, which might happen in the detailed design phase of GUI widgets such as buttons or check boxes, by introducing the form class which is one of stereotype classes as a aggregated class of GUI widget objects.

(3) Roles of development products

Incremental and iterative development process support to improve the quality and to verify the validity of software system. In order to fulfill the above benefits, measures that monitor the progress status of processes should be developed. The process management can be performed through milestone telling the progress point to a next phase or the repetition point of a process. In this methodology, pre-defined products described in the corresponding process are utilized as a milestone. For example, even though analysis and design use the same class diagrams, using stereotype classes, classes included in class diagrams of analysis phase and design phase are defined in advance and used. Also, by clarifying the difference of products occur incrementally along process, incremental and iterative process are supported.

5. Concluding Remarks

While the development of client-server information system is growing in trends, practical development support of methodology has fallen behind the trend of increasing development. Based on prototyping, although form-centered design methods have benefits such as speed and

easy of the development, they include problems such as lack of modularity, extensibility, and maintainability. Current object-oriented methodologies are too abstracted to apply to actual system development of specific domain. For the effective utilization of object-oriented methodologies, domain characteristics should be reflected to a development methodology.

In this paper, we propose a tailored object-oriented methodology for the development of client-server information system and introduce relate issues on the tailoring methodology. The UML (Unified Modeling Language) is selected as a basis modeling language, is tailored in object-oriented manner. By extension mechanisms such as stereotypes, tagged values, and constraints, the modeling language is designed without loss of original semantics in UML. For the development process, based on ISO 12207, business analysis as an extended requirement analysis is added and is integrated with system analysis/design phases by using object-oriented business model. Design phase is divided into logical design and physical design phases. Physical design phase includes activities to support seamless transition to implementation under non object-oriented tools such as visual programming tool and relational database.

In the proposed methodology, improved comprehension of domain, simplified modeling work, and intensified executablity of design model support to enhance the development productivity. In particular, the pre-defined outputs including the pre-defined stereotypes supports to identify object systematically, set level of abstraction concretely, and minimize uncertainty of tasks.

References

[1] Berson, A, *Client/Server Architecture*, McGraw-Hill, 1992.

[2] Booch, G., J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language*, Version 1.1, Rational Inc., Sep., 1997.

[3] Jacobson, I., Object Advantage: Business Process Reengineering with Object Technology, ACM Press, 1994.

[4] Jacobson, I., M. Christerson, P. Jonsson, G. Overgaard, *Object-Oriented Software Engineering. A Use Case Driven Approach*, Addison-Wesley, 1992.

[5] ISO/IEC 12207, Information Technology – Software Life Cycle Processes, Feb., 1995.

[6] Pancake, C. M, "The Promise and the Cost of Object Technology: A Five-Year Forecast," Communication of the ACM, Vol. 38, No. 10, Oct., 1995, pp. 33 - 49.

[7] Park, S.J. and J.W. Kim, "Intelligent Campus: Integration of Digital Library and Campus Information Systems," *Proceedings of International Symposium on Digital Libraries 1995*, August 22-25, 1995, Japan.

[8] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, *Object-Oriented Modeling & Design*, Prentice-Hall, 1991.

[9] Rumbaugh, J., "Modeling models and viewing views: A look at the model-view-controller framework," *Journal of Object-Oriented Programming*, May, 1994, pp. 14-20, 29.

[10] Sutherland, J., *Distributed Object Architecture for IS Applications, Distributed Object Computing*, SIGs Publication, 1994.

[11] Taylor, D.A., Business Engineering with Object Technology, Wiley, 1995.

[12] Yim, H.S., J.W. Kim, and S.J. Park, "An Object-Oriented Design Techniques for Client-Server Information System," Research on Korea Management Information System, Dec., 1996.

[13] Yourdon, E., K. Whitehead, J. Thomann, K. Oppel, and P. Nevermann, *Mainstream Objects: An Analysis and Design Approach for Business,* Yourdon Press, 1995.