Systematic Object Identification Process for Object-Oriented Software Development

Young-Sik Lee*, Sung-Joo Park**, and Moon-Ho Kim***

*, *** CALS&CIM DIV., LG-EDS Systems Inc., ** Graduate School of Management, KAIST

Abstract

This paper is written for suggesting systematic object identification process which helps the analyst find out quality objects especially in the software having GUI. The process includes the procedure and the related heuristics both. The procedure is the sequence of the activities required for object identification, and the heuristics are the useful techniques for each activity. The critical milestone is employing the concept of responsibility as a system decomposition criteria. Responsibility is a set of inherent data, functions, and relationships. The basic process is the decomposition based on the responsibility and sequential application of the related heuristics. The heuristics include object typing like macrostereotype, and microstereotype and their pattern, the generic system scenario which is made up of macrostereotypes, the idea on rule modeling, and the OO metrics.

1. Introduction

Passing through SA/SD[1], object technology has been expected to provide breakthrough which would jump over the limitation of structured method, more and more complexity from large scale and problem domain intricacy. However, the developers are less benefited from object technology. This situation comes from most because they misunderstand the real philosophy of object technology, are accustomed to structured method, and use object technology in structured way. This approach is called "object-based approach". The misuse of OO causes another complexity and low quality which is far from our expectations.

This paper tackles the essence of object philosophy. To be real object-oriented approach, we first should get quality object set in the target system. That is about identification of object which is the most critical step to real quality OO software. This is not for efficiency of OO software but for effectiveness.

2. Related Works

2.1 Classification Theory from Other Discipline

Classical Categorization

Classical categorization says that all the entities that have a given property or collection of properties in common form a category.

Conceptual Clustering

In conceptual clustering, classes (clusters of entities) are generated by first formulating conceptual description of these classes and then classifying the entities according to the description.

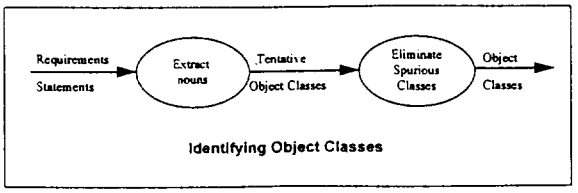
Prototype Theory

The most recent approach to classification is prototype theory. Prototype theory is for some abstractions that have neither clearly bounded properties nor concepts. One example is about classifying chairs. We understand beanbag chairs, barber chairs, and contour chairs as being chairs, not because they share some fixed set of defining properties with the prototype, but rather because they bear a sufficient family resemblance to the prototype [Lakoff & Johnson 80].

2.2 Informal English Description with Some Heuristics

Rumbaugh[2]

The approach begins by listing candidate object classes found in the written description of the problem. Classes often correspond to nouns.



After extracting nouns from problem description, he suggest to discard unnecessary and incorrect classes according to removing criteria.

[1] Structured Analysis/Structured Design

[2] [Rumbaugh et. al. 91]

Booch[3]

Booch also suggested similar method with Rumbaugh's. He emphasized the need of participation of domain experts for elicitation of candidate nouns and use of related metrics for evaluating object quality.

2.3 Object Typing

Jacobson[4]

Jacobson has classified object types in his use-case[5] analysis. He suggests that there are three types of object i.e. interface objects, control objects, and entity objects exist in each use-case. This could give us insights about what type each objects candidates match and thus helps object identification.

Budd[6]

Budd has also suggested object types. He proposed four object types - Data Managers, Data, or State Classes, Data Sinks or Data Sources, View or Observer Classes, and Facilitator or Helper Classes.

2.4 Other Object Identification Analysis

As other object identification methods, there are [Shlaer & Mellor 88], [Coad and Yourdon 90], [Wirfs-Brock & et al. 90], CRC[7], and Domain analysis of [Moore & Bailin 88].

2.5 Weakness of the Existing Object Identification

They all, in their own way, have contributed and helped analysts to identify the objects. However they are not yet complete. They are not practical both in identifying objects and assigning responsibilities because they still maintain upper cognitive point of view and do not consider the problem of responsibility assignment and real current software development environment, event-driven GUI system.

3 Systematic Object Identification

Basically the systematic object identification is the trial to identify object by its generic and inherent responsibility. Fundamentally responsibility relies on our common sense. That is what for object to do.

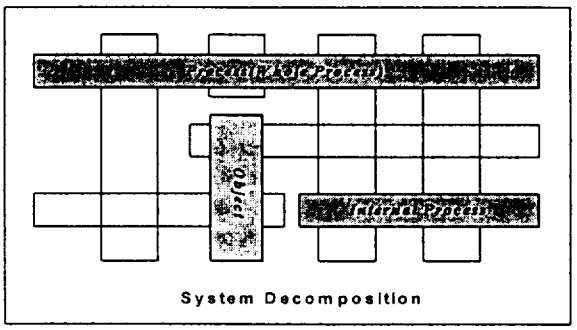
3.1 Responsibility

In classical meaning of responsibility in CRC or Wirfs-Brock, responsibility means simply behavior of object. But in more modern and object-oriented sense, responsibility not only means behavior but also data and linkage to other objects.

Basically responsibility should be "generic and inherent" to an entity. The object which have generic and inherent responsibility is "single-minded and self-managed" object. "Single-minded" means that the object have only one goal as possible as it could, that is, the hammer has the responsibility of nailing only. And "self-managed" means that the object is independent on the other objects as much as possible. Object identification based on its generic and inherent responsibility would produce these single-minded and self-managed objects.

3.2 Software Systems Decomposition by Responsibility

This paper propose the decomposition by responsibility. The paper have chosen the whole process as the first candidate for decomposition because process is the basic unit of collaboration. The process could be the whole process penetrating all the related objects or internal process connecting a few objects in collaboration unit. The internal process is almost close to functional cohesion which provides service to whole process. However the whole process directly contacts with user.



System Decomposition

3.3 Business Rule Modeling: Object as a Data Manager

Business rule is a specialization of control. For example, consider the following rule.

> "The good client could get 10 processing orders simultaneously"

In rule modeling, the heuristics in control modeling could equally be applied. That is if a rule involve several objects, we should first distribute rule and next create additional rule object. Another really helpful points to consider is the following. Almost all case, a rule accompanies critical data, for example, "10" is that data in the above rule. A object, as a data manager, should have the rule if it owns that critical data. Therefore, the above rule should belong to the client. The client object don't need to be passive any more.

---

[3] [Booch 94]

[4] [Jacobson 92]

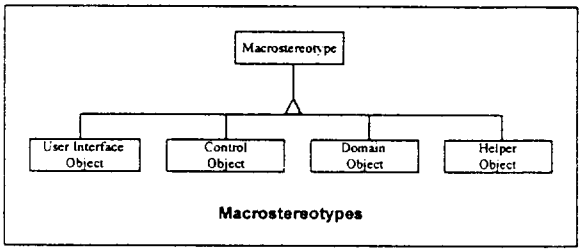
[5] a particular form or pattern or exemplar of usage, a scenario that begins with some user of the system initiating some transaction or sequence of interrelated events

[6] [Budd 91]

[7] Class, Responsibility and Collaboration. [Beck & Cunningham 89]

3.4 Macrostereotypes

Macrostereotype literally means more general categorization of objects. In general client/server application having GUI is composed of these responsibilities; user interfacing responsibility, control responsibility, data managing responsibility, and system helping responsibility.



Macrostereotypes

User Interface Objects
The responsibility of user interface objects is to allow the user to access the system and is the entrance point to the system.

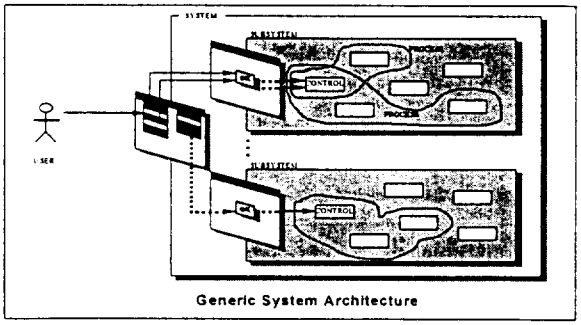
Control Object
Control object is the objet which sequence and coordinate the other objects and is the object which first takes the message from the user interface objects.

Domain Object
Domain objects are real world objects. Domain objects have persistent data and these data are stored in database.

Helper Object
Generally helper objects means system utilities, common libraries, or API objects.

3.5 Generic Scenario of GUI Software System[8]

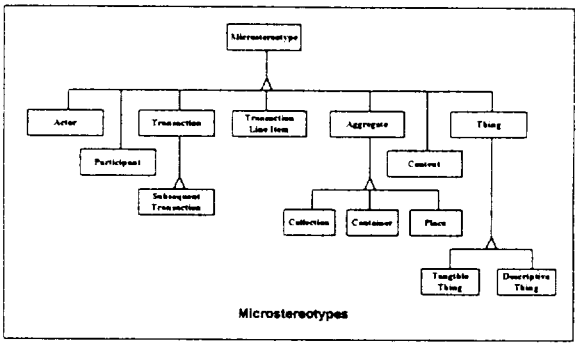
By assembling macrostereotypes properly, we could construct general system scenario.


Generic System Architecture

This generic scenario would be greatly helpful to the developer as a basic guidelines when designing new application. He can add specific attributes and methods to this skeleton

3.6 Microstereotypes

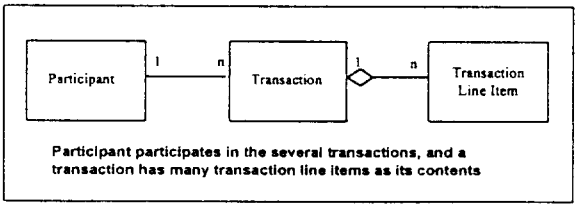
Microstereotype is the categorization of domain objects, which is one kind of macrostereotype, based on their generic responsibilities. Following figure shows these microstereotypes hierarchy.


Microstereotypes

Actors
Actor represents entities that act meaningful role in the system.

Participants
In many cases, actor participates in some transactions as a participant, see following figure.

Transactions and Transaction Line Items
Transaction is a event that must be remembered for a long time. For example, consider agreement, assignment, payment, purchase, reservation, sale, shipment, withdrawal, session, and etc.. Almost all transaction has transaction line items, see following figure.



Participant participates in the several transactions, and a transaction has many transaction line items as its contents

Aggregate and Contents
Aggregate is aggregation of something, contents.

Tangible Things and Descriptive Things
Tangible things are all the things that are visible, and have shapes, so we can touch them.

Descriptive things are all the things without shape and are invisible and untouchable, but they certainly exist.

3. 7 Metrics for Object Identification

Metrics implies that these metrics must be managed and we must design the system and identify the objects considering the concepts of metric at the very first time. At the same time, the metrics are used to measure some design features and feedback the design criteria. The

[8] GUI software system means software system having Graphic User Interface.

useful metrics are as follows.

Metrics for Object Modeling Stage
. Number of Data Attributes
. Number of Public Methods
. Number of Relationships
. Semantic Cohesion
. Understandability

Metrics for Collaboration Diagramming Stage
. Number of Messages in a Collaboration Unit
. Number of Objects in a Collaboration Unit
. Average Number of Messages (= Number of Messages in a Collaboration Unit / Number of Objects in a Collaboration Unit)
. Number of Parameters & Amount of Returned Data
. Number of Public Methods
. Method Size by Lines of Code

3.8 Object Identification Process

Object-Oriented Heuristics
As a summarizing and reminding comments, the following basics are worth to be kept in mind.

- Consider the reason for being of object rationally.
- Consider what the events are really going on in real world. Object as possible as we can, must exist in the real world and the collaboration scenarios are also.
- The issues about design must be postponed.
- But the reusability must be considered.
- One system, one control and one service.
- Grant soul and intelligence to nonliving things. Object as data manager.
- Remember the rule of rule distribution.
- There are macrostereotypes and generic scenario in software system having GUI.
- Use microstereotypes and their pattern when deal with domain objects.

The Object Identification Process
- Identify the objectives of the system with about 3 sentences.
- Identify all system features.
- Group the discovered processes based on similar functions.
- Prototype user interface form for each whole process.
- Microstereotype domain objects.
- Build domain object model.
- Write out the first cut process i.e. scenario to accomplish the features.
- Add the scenario with the user interface objects, other helper objects based on the generic scenario of GUI system. And consider rule distribution.
- If applicable, evaluate the process and objects with proper metrics.
- Return to the first step to refine the models.

4 EXAMPLE

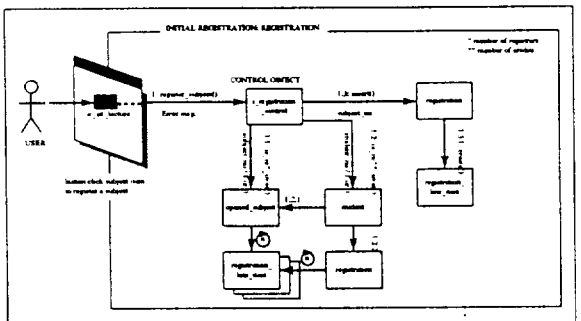
Exemplifying case is registration process in scholar information system which is a subsystem of CAIS[9]. Target system is already constructed and working. The paper have examined this existing system first and redesigned it with suggested object identification process.

Collaboration Diagram for Registration of a Subject

The student initiate the registration process by double clicking one item of opened subjects to insert that subject in the pool of the registered subjects. This is the most important event in registration subsystem.

First we apply the macrostereotype, microstereotype, and pattern according to the suggested process, we easily obtain object model. Next applying generic architecture and related heuristics like rule modeling, we obtain the reengineered version, see following figure.

For example, in the scenario, two rules are involved that are rule of maximum number of total registrars and the maximum of total credits. If these two rules are satisfied, the student can register a subject. In present system, the control object checks all rules, and the domain object only behaviors like data repository. By applying the rule-checking heuristics, we could distribute these rules among the related objects.



Reengineered Collaboration Diagram for Scenario of Registering Subject

From this revision, the concentration of responsibility on the control object was reduced. Consequently, the LOC of control object was greatly reduced. The amount of the transferred parameters and data was reduced also. The number of objects in the scenario is not changed. The qualitative metrics like understandability and semantic cohesion was raised. We can more easily under-

[9] Campus Advanced Information System which is the campus-wide information system in KAIST.

stand the scenario and the location of the responsibilities.

5 Conclusions

The most important and difficult thing in everyday real life is finding out the happy medium in trade-offing problems. Object identification is another kind of trade-offing problem. Where to locate the responsibility is this kind of problem. The paper have tried to solve this problem i.e. proper responsibility distribution in object-oriented software system having GUI. And the result is "systematic object identification process". The process is the mix of procedure and related heuristics.

Basically the procedure is the systems engineering process, that is, recursive decomposition based on responsibility. The related heuristics are helpful to the problem of what kinds of responsibility to what kind of object. The specific results are summarized in the following:

- . systematic process adapted from systems engineering
- . heuristics on the concepts of generic and inherent responsibility, in other words the concepts of single-minded and self-managed objects
- . heuristics on macrostereotypes and their generic responsibilities
- . generic architecture of GUI system, this is where to start heuristics
- . heuristics on microstereotypes and their inherent responsibilities
- . heuristics on rule distribution
- . metrics which could be used for feedback to find out quality objects

The very contribution of this paper is the trial to systemize the object identification problem which was performed based on the human cognition and common sense before.

Further Research Directions

If the object type and responsibility type could be formalized in particular domain, we may expect pattern set and frameworks in that domain. In frameworks basic responsibility are given and additionally the expert system can help the developer decide where to locate the responsibility based on more concrete heuristics. This result goes beyond existing CASE tools with additional intelligence and more automatons and less decisions by developer. Developing more intelligent CASE tools will greatly contribute and will be real opportunity in object-oriented software engineering area.

References

[Beck & Cunningham 89], , Beck, K. and Cunningham, W., A Laboratory for Teaching Object-Oriented Thinking. *SIGPLAN Notices* vol. 24(10), October 1989

[Booch & Rumbaugh 95], , Grady Booch and James Rumbaugh, *Unified Method version 0.8*, Rational Software, 1995

[Booch 94], , Grady Booch, *Object-Oriented Design* 2nd Ed., Benjamin-Cummings, 1991

[Budd 91], , Timothy Budd, *An Introduction to Object-Oriented Programming*, Addison-Wesley, 1991

[CAMIS 95], , CAMIS (Center for Advanced Management Information System), *Registration System: Development Models*, CAMIS, 1995

[Coad & Yourdon 90], , Coad, P and Yourdon, E. 1990. Object-Oriented Analysis, Prentice-Hall, 1990

[Coad & et. al. 95], , Peter Coad & et. al., *Object Models - Strategies, Patterns, and Applications*, Prentice-Hall, 1995

[Coleman et. al. 94], , Derek Coleman et. al., *Object-Oriented Development: The Fusion Method*, Prentice-Hall, 1994

[Hendersen-Sellers 96], , Brian Henderson-Sellers, *Object-Oriented Metrics - Measures of Complexity*, Prentice-Hall, 1996

[Jacobson 92], , Ivar Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992

[Jacobson 95], , Ivar Jacobson, *The Object Advantage*, Addison-Wesley, 1995

[Lakoff & Johnson 80], , Lakeoff, G. and Johnson, M., *Metaphors We Live By*, The University of Chicago Press, 1980

[Lorenz & Kidd 94], , Mark Lorenz and Jeff Kidd, *Object-Oriented Software Metrics - A Practical Guide*, Prentice-Hall, 1994

[Moore & Bailin 88], , J. Moore and S. Bailin, *Position Paper on Domain Analysis*, CTA, 1988

[Rumbaugh et. al. 91], , James Rumbaugh et. al., *Object-Oriented Modeling and Design*, Prentice Hall, 1991

[Shlaer & Mellor 88], , Shlaer, S. and Mellor, S., *Object-Oriented Systems Analysis: Modeling the World in Data*, Yourdon Press, 1988

[Tegarden et. al. 95], , David P. Tegarden, Steven D. Sheetz, and David E. Monarchi, "A Software Complexity Model of Object-Oriented Systems", *Decision Support Systems* 13, 1995

[Wilkinson 95], , Nancy M. Wilkinson, *Using CRC Cards*, SIGS Books, 1995

[Wirfs-Brock & et al. 90], , Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener, *Designing Object-Oriented Software*, Prentice-Hall, 1990