Analyzing Business Domains: A Methodology and Repository System

Heeseok Lee and Jae Lee

MIS, Graduate School of Management
Korea Advanced Institute of Science and Technology, Seoul
hlee@msd.kaist.ac.kr, jlee@cais.kaist.ac.kr

Abstract

Object-oriented approaches have been widely applied in enhancing reusability for software implementation. However, identifying reusability potentials during analysis phases is not a trivial task. This paper proposes a methodology for analyzing business domains. These domains are built in the form of reusable business objects. The methodology consists of three phases: (i) domain context modeling, (ii) domain semantic modeling and (iii) domain reuse modeling. A domain repository system is developed to help analysts use the methodology for reuse supports in a systematic way. To demonstrate the usefulness of the methodology, a real-life example is illustrated.

I. Introduction

Object-oriented technology promises a natural way of building quality software by conceptualizing developers' cognition on objects and their behaviors. This conceptualization represents real human activities. These benefits are achieved by two major features of the object-oriented technology: encapsulation and inheritance [Graham 1994]. For these features, the object-oriented technology has been applied in many system development fields successfully. Reusability, in particular, plays a key role in guiding information systems into an object-oriented approach. Reusability is an important aspect of information system development. Researches on reusable codes are already abundant. However, researches on reusable analysis output are left something to be desired.

Software development is not a trivial task, that is, many development constraints, different characteristics, and special development needs of application domain must be simultaneously considered [Henninger 1995]. These complexities call for eliminating irrelevant requirements and reducing the specific problem into a manageable chunk. Therefore, it is reasonable to define the scope of a domain, an area that shares common characteristics.

Berard [1993] defines a domain as a collection of current and future application that shares a set of common characteristics. In contrast, MIS professionals are familiar with the following definition: *A manageable business area that shares the common properties.*

In order to identify and understand a domain and its commonalties, a domain analysis is needed. Domain analysis is a process of identifying, collecting, organizing, and representing the relevant information in a domain, based on the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within the domain [Kang, et al. 1990]. Domain analysis differs from other IS analysis in that it is not a life cycle activity. Domain analysis goes on in parallel with the software life cycles [Berard 1993]. Domain analysis is best separated from application analysis because development pressures make it hard to abstract from immediate concerns to flexible, complete and application-independent components

11[Graham 1995]. Therefore, domain analysis is a long-term process for defining and refining business objects to represent a business model.

As IS analysis is presented within the framework of software engineering, so is domain analysis within the framework of domain engineering. Kang et al. [1990] defines domain engineering as "an encompassing process which includes domain analysis and the subsequent construction of components, methods, and tools that address the problems of system/subsystem development through the application of domain analysis products."

Neighbors [1980] who originally introduced domain analysis, did not use an object-oriented analysis (OOA) methodology. Shlaer and Mellor [1989] introduced a brief concept of object-oriented domain analysis (OODA). The object-oriented technology offers many benefits to domain analysis. An object-oriented decomposition for a domain naturally generates reusable domain models. They may be classes, scenarios or domain-specific rules. OOA leads naturally to domain analysis and thus lends to more widely reusable components than general application analysis [Mili et al. 1995]. The object-oriented methodologies use the concept of inheritance, encapsulation, and abstraction, which supports the reusability in a more secure way. [Table1] compares the common OOA with the OODA.

[Table 1] Comparison between OOA and OODA

issue \ method	OOA	OODA
Main Task	Defining object	Defining reusable components
Project Term	Short / Middle term	Long term (on-going)
Systematic Support Need	Optional	Mandatory
Target System	Current Business System	Potential Domain System
Target Project	Individual Projects	Independent of Individual Projects (Prerequisite of OOA)

Object-oriented domain analysis deals with reusable items, such as objects, object relationships, object constraints, object responsibilities, objects in scenario, and task descriptions in specific domain. Major domain analysis methodologies to date, introduce detailed steps and input-output specifications (e.g., Kang et al. [1990], Tracz et al.[1995]). Domain analysis is a long-term process and thus requires systematic approaches. A repository system may be a good candidate for managing the domain knowledge and artifacts in a systematic manner. Systematic supports, such as database systems, and graphical models, are mandatory in reusing a domain model [Prieto-Diaz 1990].

In this paper, we (i) propose a domain analysis methodology for developing business applications and (ii) develop a domain repository system (DRS). The proposed business domain analysis (BUDA) consists of three phases, such as Domain Context Modeling, Domain Semantic Modeling, and Domain Reuse Modeling.

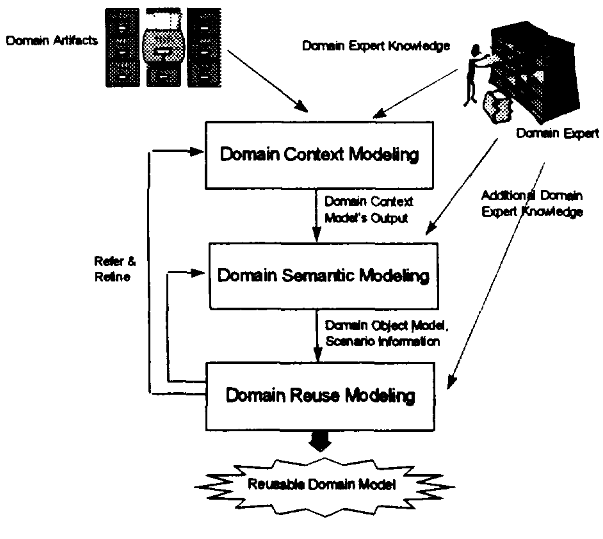
II. Business Domain Analysis Methodology

Object-oriented technologies have gained widespread acceptance in many system application areas as their chief way of building quality software in a cost-effective manner [Graham 1995]. However, objects have not been widely used to represent a business world, even though, a business may be modeled in terms of objects effectively. It has been noted that business model is represented by the use of business objects [Casanave 1995]. Business objects differ from system objects such as window user interfaces. Business objects are application independent and atomic components for representation of real world. According to Object Management Group (OMG) [BOMSIG 1995], business objects encapsulate the storage, metadata, concurrency, and business rules associated with a thing, process, or event in a business.

Business areas are more dynamic, complex to make reusable components and difficult to analyze the requirements of users than system application areas. It means that data driven approaches, like ERM, are not suitable for designing the integrated business model. ERM is originally devised for design of static data model. Many information systems are still adopted data driven approaches, dynamic aspects of business model are considered separately or simplified. Previous domain analysis methodologies that use ERM also make up for these problems by adopting Data Flow Diagrams (DFD), or State Transition Diagrams (STD). To solve these problems, mapping between business model and objects can be a good solution. From these ideas, we will develop natural system development phases, from analysis to implementation, through the medium of the business objects. So it is required that domain analysts adopt the object-oriented technology when business areas are designed.

Domain analysis on business area is the first step for building the object based enterprise infrastructure. Domain analysis is conducted before the IS analysis, and supports IS analysis by the object model. The continuous reuse of the object model can refine the object model and increase the productivity of IS. Object-oriented model may be naturally reused by using the abstraction, inheritance. However additional analysis activities increase the reusability of object model. These analysis activities are the processes of defining the reusable model components such as "contracts" [Wirfs-Brock etc. 1990] or the process of expending the object model with "roll" [Gottlob et.al. 1996].

Our methodology, BUDA consists of the three phases: *Domain Context Modeling, Domain Semantic Modeling, and Domain Reuse Modeling*. These phases are shown graphically in [Figure 1].



[Figure 1] Phases of BUDA

2.1 Domain Context Modeling

Domain context modeling consists of two subphases. The first subphase analyzes the enterprise domain contexts to cut off the domain of interest to manageable size. In business environment, a domain is a distinctive and manageable business area that shares common properties. In general, corporations have the two domain types. Vertical domain, such as marketing, administrations, and finance, studies a number of systems intended for the same class of applications. Horizontal domain, industrial division, such as hospital case, stocks company case. An analysis of enterprise business context is needed for the entire corporation. In determining the scope of a domain, both top-down and bottom-up approaches are used in combination. In general, each domain consists of business scenarios or subsystems. The domain boundary is determined by the use of the concept: "Sharing a set of common characteristics".

The next sub phase collects the related domain knowledge. The sources come from legacy systems, existing internal documents, future trends in the industry, interviews with end users, or knowledge of domain experts. These domain artifacts are categorized into (i) industry and technology, (ii) constraints and rules, and iii) legacy application information (static and dynamic). Industry and technology category includes industrial trends, technological information, and business forecasting data, which are in unspecified format. Information on constraints and rules is used for class/scenario analysis in domain semantic modeling phase. This information can be founded in a specific job/task manual. Legacy application information can be found from ERD, DFD, application screens, relational database tables, and fields. This information is used to define objects, attributes, methods, and events. (i.e., like a domain dictionary). Domain analysis team may need to append a new type of artifact. In this case, classification of the previous domain artifacts is modified. The outputs of domain context modeling are *Domain Definition, and Domain References (artifact lists).*

2.2 Domain Semantic Modeling

The domain semantic modeling phase is similar to traditional object-oriented analysis phases. In the domain semantic modeling phase, joint works with domain experts are essential. Three steps of domain semantic modeling are (i) to decompose domains by analyzing scenarios, (ii) to identify objects, attributes, and methods by the use of class cards, and (iii) to analyze collaboration between objects.

BUDA adopts a scenario-based model for analyzing domain semantics. A scenario describes each interaction that may involve several objects. Actors and scenarios are the primary components in the scenario-based model. An actor represents a role that someone or something in the domain can play in relation to the business [Jacobson et al. 1995]. Actors create events. All scenarios are triggered by the events. For example, the following is the scenario "Cash Deposit". The actor is a customer.

Customer keeps the personal information (account number, pin number, and social security number), and deposits amount on banking sheet. Wait until his/her turn. Employee calls his/her name, the customer gives a sheet to the employee. After some verification (number check, amount check), the employee gives the receipt of deposit to the customer.

Task or job analysis on the related domain is required for finding scenarios. Typically, the total set of scenarios explains the complete semantics of domain. Constraints or rules related to scenarios must be collected. In this analysis subphase, constraints or rules are analyzed and managed in an abstract way. If sets of scenarios are formulated, the next subphase is to extract verbs and nouns from each domain. Noun and verb lists in each scenario help in building the basic structure of object modeling. Naming conflict (for example, synonyms or homonyms) are resolves for keeping the unity of domain semantics. When classes are identified, actors in scenarios become the primary candidates for domain objects. Noun and verb lists are used for identifying domain objects. Nouns in noun lists are the candidates for domain objects. Verb lists can provide the responsibilities for domain objects. Therefore verb lists are the candidates for methods. Class cards are developed by the use of noun lists. They are useful in connecting *developer or IT person* with *domain expert or domain user*. A class card includes class name, attributes, methods, collaborators, generalization information, and descriptions.

The output of domain semantic modeling is the preliminary domain object model. This model represents the overall domain semantics using the object-oriented methodology.

2.3 Domain Reuse Modeling

In the domain reuse modeling phase, the domain object model is refined. Aggregation and generalization relationships are primarily considered. To find more aggregation or generalization hierarchies, it may be possible to evolve the new class from an existing one [McGregor & Sykes 1992]. Framework approaches also increase the reusability of analysis model. Frameworks are skeletal structures of programs that must be fleshed out to build a complete application [Wirfs-Brock, et al. 1990]. Many researches on the reusability are conducted using component or framework approach to increase the reuse opportunity. In fact, to extract the components, a unit of reuse, is not an easy approach, as explained. Therefore it is meaningful to apply the characteristics of framework (such as reusable type, pattern,..) to the domain model. In this paper, the unit of reuse will be explained using the relationship between scenarios and objects. In general, the relationship between scenarios and objects is described as "Many-to-Many". That is, one scenario has many objects, and one object acts in many scenarios. To explain this "Many-to-Many" relationship, this paper uses the CRUD (Create, Read, Update, and Delete), as the kind of type unit. In business application areas, the CRUD is the most frequent collaboration between business objects. These can increase the opportunity of reuse by referring scenarios and objects that is related to domain specific business environments. In object communication, the responsibility of object is classified into the two types, i) client : Object that invokes the method of other objects and ii) server : Object that executes the method of itself by requests of other objects. When application developers or other users reuse the domain information, some guidelines can be given. Outcome of a search can be one of the following types: Case of Object-related / Scenario-related / Both (object/scenario)-related / None-related. In case of "Object-related", basic object information such as class descriptions, attributes, methods, and relationships are given. Information about "class rolls in related scenarios" is also given. If the outputs of domain artifact search exist, it will be presented. When the search word is related with scenarios, the basic scenario informations (descriptions, related domains) are given. Informations about "class collaborations in the scenario" are also given. Class collaborations consist of class names (client, server), method, method arguments, and CRUD type. "Both-related" implies that analysts get information related with both object and scenario.

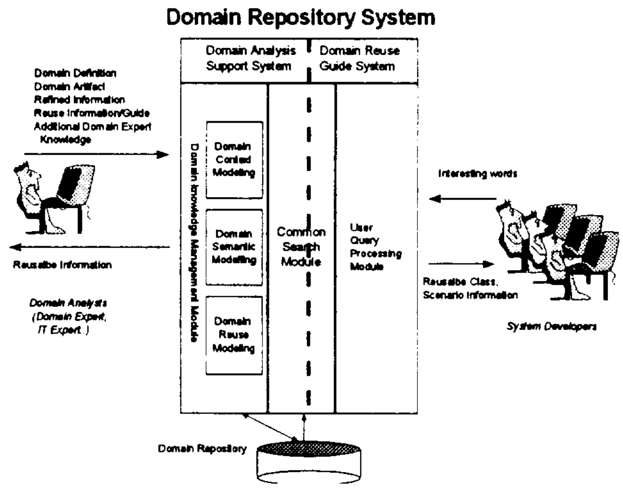
[Table 2] compares other domain analysis methodologies with BUDA. This table shows that generic, common steps for domain analysis exist and the object-oriented technology is applied to domain analysis. Reuse methods are not only the reusable domain model, but also reuse guidance, or reuse opportunities. In this table, it is clear that the emphasis of BUDA is on the reusability using a systematic repository and reusable class role information on business applications.

[Table 2] Comparison of Domain Analysis Methodologies

	FODA	DSSA	JODA	DA InDE by ARC	BUDA
Author(s)	Kang et al. [1990]	Tracz	Hollibaugh [1993]	Stropky et al. [1995]	Lee et al. [1997]
Target Domain	IS Software	IS Software	IS Software	Business	Business
Base Model	ER	ER+OO	OO	OO	OO
Analysis Steps	1. Context Analysis 2. Domain Modeling 3. Architecture Modeling	1. Define Scope of the domain 2. Define/Refine Domain Specific concept 3. Define/Refine Domain Specific Design Implement. 4. Develop Domain Architecture 5. Produce reusable Workproduct	1. Prepare Domain 2. Define Domain 3. Model Domain	1. Identify Information Source 2. Gather Domain Information 3. Define Domain 4. Describe Domain 5. Establish Domain Knowledge 6. Determine Composition 7. Define Structure 8. Identify Commonality	1. Domain Context Modeling 2. Domain Semantic Modeling 3. Domain Reuse Modeling
Main Products	1. Domain Model 2. Feature Model	1. Domain Model 2. Domain Specific Software Archi. 3. Domain Reusable Component	1. DomainDefinition 2. Domain Model	1. Domain Architecture Model	1.Domain Object Model
Reuse Methods	1. Feature Model 2. Reuse Library	1. Reusable Components	1. Domain Model 2. Active Repository	1. Reuse Opportunity	1 Reuable Class Role 2. Systematic Reuse Guidance
Repository Support	N/A	DADSE(DICAM Appication Development Support Environment)	Active Repository	DKDB(Domain Knowledge Database)	DRS (Domain Repository System)

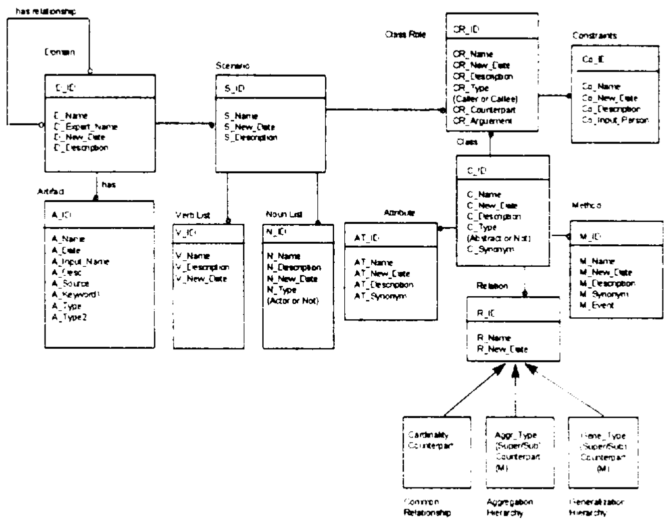
III. Domain Repository System

A Domain Repository System (DRS) consists of the following two sub systems: (i) Domain Analysis Support system (DASS) and (ii) Domain Reuse Guide System (DREGS). DASS helps domain analysis team in following the BUDA methodologies. The primary purpose of DASS is to manage and utilize domain information. DASS provides the subsystems for an each phase of BUDA. We can use the Common Search Module (CSM) to find useful domain information for BUDA. CSM helps DASS, DREGS in finding target domain information for reuse the domain information. Current version of CSM utilizes the search technique from SQL (structured query language) statements. DREGS is a guiding-system for system developers and other related users. DREGS also enables the domain analysis team to reuse of domain information for high quality and secure future systems. The overall system architecture of DRS is shown in [Figure 3].


Domain Repository System
Domain Analysis Support System
Domain Reuse Guide System
Domain Definition
Domain Artifact
Refined Information
Reuse Information/Guide
Additional Domain Expert Knowledge
Reusalbe Information
Domain Analysts (Domain Expert, IT Expert.)
Domain knowledge Managem ent Module
Domain Context Modeling
Domain Semantic Modelling
Domain Reuse Modeling
Common Search Module
User Query Processing Module
Interesting words
Reusalbe Class, Scenario Information
System Developers
Domain Repository

[Figure 3] System Architecture of DRS

In accordance with each BUDA phase, domain information is produced and saved in the repository. To deliver a fast, correct search result and to store the domain information effectively, the following meta-data schema is used. As current version of DRS uses the relational database, an ER data model will be presented. This meta-data schema (see [Figure 4]) has focused on the three components: Domain, Scenario, and Class. A domain can have the many scenarios. Classes have relationships with scenarios, but do not have the direct relationships with domains.

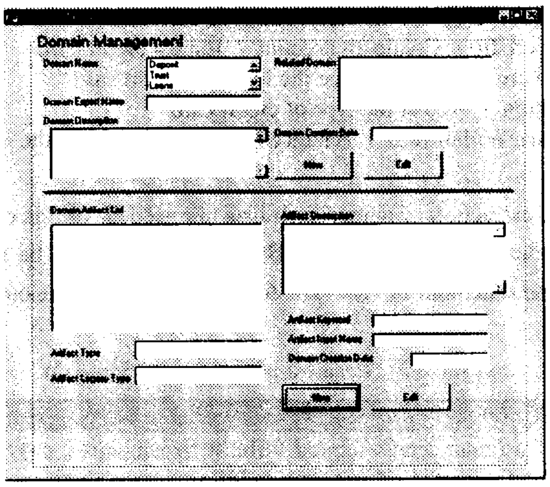


[Figure 4] Metadata Schema of Domain Repository

3.1 DASS (Domain Analysis Support System)

DASS is a support system for the primary phases of BUDA. If the DSSA button is pushed, users can use the

DSSA with a pull-down menu. Pull-down menu consists of Domain Semantic Modeling, Domain Semantic Modeling, and Domain Reuse Modeling. At first, domain context modeling is conducted. The following figure shows the domain context modeling support module. In this module, we conduct the two main activities: (i) Domain Management (Definitions and Descriptions) and (ii) Domain Artifact Management.

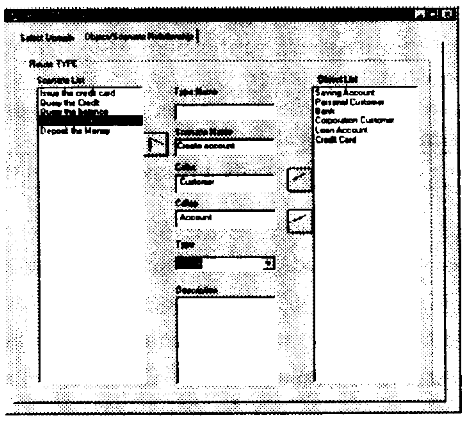


[Figure 5] Domain Context Modeling Support Module

To analyze domain semantic model, class card and scenario management modules are given. If the analyst chooses a "domain reuse modeling" menu, the domain reuse modeling support module will be activated. [Figure 6] is a supporting module for "class collaboration analysis". In this screen, the analyst selects a scenario and classes (client, server) to describe the collaboration name. Scenarios in this screen are in the domain that selected a previous screen. If the analyst finds the more generalization and aggregation hierarchies, analysts can append these information by using class card.

4.2 DREGS (Domain Reuse Guide System)

DREGS provides the useful information to domain users who develop information systems. When the domain user requests a search, DREGS provides the output of search such as the related information in the lists of objects and scenarios.



[Figure 6] A Screen of a Domain Reuse Modeling Module

IV. Case Study

This chapter introduces a small case of banking applications. Target bank is F. bank (FB) in Seoul, Korea. FB has a plan to analyze a customer-focused domain. Customer domain is defined as "Areas that deal with customer related information or has a contact with customer" Collected domain artifacts are (i) legacy information system documents (ERD, System Screens...), (ii) interviews with domain experts, and (iii) other banks related materials. To analyze this domain, get the following scenario lists:

Query the customer information / Request the credit status of customer
Adjust the degree of customer credit / Create an account / Issue a credit card
Cancel an account / Cancel a credit card / Notice the loss of a credit card
Change the customer information / Adjust the type of credit card
Reissue a credit card / Deposit /Withdrawal / Balance inquiry / Calculate the interest...

To analyze the scenario, the extraction of noun / verb lists are needed. In this example, scenarios of "Create an account", "Change the customer information", and "Balance inquiry" will be analyzed. Instead of verb, verb phrases are used for keeping the semantics of verb. The following tables are the outputs of the scenario analysis.

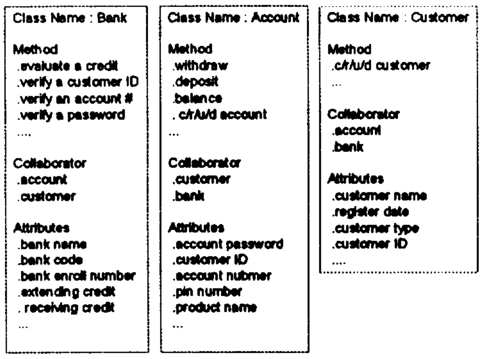
[Table 3] "Create a new account" Scenario

Scenario	Description / List
"Create an account" Scenario	Customer send a request form, with ssn, name, password, type of account, and personal information (address, phone number, age, office name, and office phone number). After some verification (ssn check, password rule check, and credit of personnel), if problems exist, then return a request form and request the modification of personal information to customer. Else create the account number and request an approval to bank. After the approval, return the new-bank book to customer.
Name List	customer, request form, ssn, name, password, type of account, address, phone number, age, office name, office phone number, verification, rule, credit of personnel, problem, account number, approval, banking manager, bank-book
Verb List	send a request form, verify a request form, request the modification, create the account number, request an approval to banking manager, return the new-bank book

[Table 4] "Balance inquiry" Scenario

Scenario	Description / List
"Balance inquiry" Scenario	Customer sends an inquiry form, with account number, password to bank desk. After password verification, After the approval, return the balance information to customer.
Name List	Customer, inquiry form, account number, password, bank desk, address, verification, approval, balance information
Verb List	send a inquiry form, verify a password, return the balance information

After the extraction of noun / verb lists is done, analysts conduct the naming conflict resolution for noun / verb lists. Naming conflicts can occur in a scenario or between scenarios. For example, in the "Change the customer information" scenario, "change the information" and "update the customer information" have the same semantics. The following lists are some resolutions of naming conflict: Customer ->Personal Customer / Name ->Personal Customer Name / Approval -> Password Approval. Now we obtain the candidate objects such as Customer, Account, and Bank. To test the responsibilities of objects, the simple class cards are made like these:



Class Name : Bank

Method
.evaluate a credit
.verify a customer ID
.verify an account #
.verify a password
....

Collaborator
.account
.customer

Attributes
.bank name
.bank code
.bank enroll number
.extending credit
. receiving credit
...

Class Name : Account

Method
.withdraw
.deposit
.balance
. c/r/u/d account
...

Collaborator
.customer
.bank

Attributes
.account password
.customer ID
.account nubmer
.pin number
.product name
...

Class Name : Customer

Method
.c/r/u/d customer
...

Collaborator
.account
.bank

Attributes
.customer name
.register date
.customer type
.customer ID
....

[Figure 7] Class Cards for Classes

To model the reuse characteristics, we must analyze the generalization and aggregation. Customer class evolves into personal customer and corporation customer class. Account class also evolves into saving account and loan account class. Finally, the analysis of the class collaborations is needed. The following tables are the outputs of this analysis.

Role Name	Request the creation of account	Scenario Name	Create a new account
Client Name	Personal Customer	Server Name	Saving Account
Method	SavingAccount.Create()	Constraints	SSN Check
Argument	Customer Name, Password	CRUD	Create

[Figure 10] "Request the creation of account" Type

Role Name	Request the verification of new customer	Scenario Name	Create a new account
Client Name	Saving Account	Server Name	Bank
Method	Bank.VerifyNewCustomer()	Constraints	Password Check
Argument	Customer SSN	CRUD	Read

[Figure 11] "Request the verification of new customer" Type

V. Conclusions

Domain analysis is preliminary for system analysis. Domains are best analyzed in an on-going fashion. For long-term projects, the support of managers is important. Managers of domain analysis must be experts who have professional knowledge in the domain of business and technology.

In this paper, a methodology is developed to enforce business analysis outputs with reusability and quality by the use of object-oriented technologies. This methodology differs from previous ones in applying the domain analysis to business application area. It proposes the type of reuse by the use of class-scenario relationship. Furthermore, it supports the complete cycle of domain analysis and reuse activities by using a domain repository system.

The work in this paper needs further enhancements and research. First, the methodology must be extended to cover the complete phases of domain engineering. Second, the repository system needs more versatile features like graphical diagram supports or intelligent agents. Third, the methodology may be validated though the application to a real-life projects in a larger scale.

References

[Berard 1993] Edward V. Berard, "Essays on Object-Oriented Software Engineering: Volume I", Prentice-Hall, 1993

[BOMSIG 1995] Business Object Management Special Interest Group, "OMG Business Application Architecture (White Paper)", WWW URL :http://www.omg.org, 1995

[Booch 1990] Grady Booch, "Object Oriented Design with Applications", Benjamin/Cummings Publishing, 1990

[Casanave 1995] Cory Casanave, "Business Object Architecture and Standards", OOPSLA 95, Workshop report: Business object design and implementation, 1995

[France et al. 1995] France, Robert B. and Horton, Thomas B., "Applying Domain Analysis and Modeling: An Inderstrial Experience", Proceedings of the ACM SIGSOFT, SSR'95, April 1995, pp206- 214

[Gottlob et.al. 1996] Georg Gottlob, Michael Schrefl and Bgrigitte Rock, "Extending Object-Oriented Systems with Roles", ACM Transactions on Information Systems, Vol.14,No.3, July 1996, pp268 - 296

[Graham 1994] Ian Graham, "Object Oriented Methods", Addison-Wesley, 1994

[Graham 1995] Ian Graham, "Migrating to Object Technology", Addison-Wesley, 1995

[Henninger 1995] Scott R. Henninger, "Developing Knowledge Through the Reuse of Project Experiences", Proceedings of the ACM SIGSOFT, Symposium on Software Reusability(SSR'95) , 1995

[Hollibaugh 1993] Robert Holibaugh, "Joint Integrated Avionics Working Group (JIAWG) Object-Oriented Domain Analysis Method (JODA)", Special Report CMU/SEI -92-SR-3, November, 1993

[Jacobson 1992] Ivar Jacobson, "Object Oriented Software Engineering: AUse Case Driven Approach", Addison-wesley, 1992

[Jacobson et al. 1995] Ivar Jacobson, M.Ericsson, and A. Jacobson, "The Object Advantage: Business Process Reengineering with Object Technology", Addison Wesley, 1995

[Kang et al. 1990] Kyo C. Kang, Sholom G. Cohen, James A. Novak, William E. Hess, and A.Spencer Peterson, "Feature Oriented Domain Analysis (FODA) Feasibility Study", CMU/SEI-90-TR-21, Software Engineering Institute, Pittsburgh, PA, Novemeber, 1990

[Lorenz 1993] M. Lorenz, "Object-oriented Software Development: A Practical Guide", Prentice Hall, 1993.

[McGregor & Sykes 1992] John D. McGregor, and David A.Sykes, "Object-Oriented Software Development : Engineering Softearc for Reuse", Van Nostrand Reinhold, 1992

[Mili et al. 1995] Hafedh Mili, Fatma Mili, and Ali Mili, "Reusing Software: Issues and Research Directions", IEEE Transaction on Software Engineering, Vol.21, No.6, 1995

[Neighbor 1980] J.M. Neighbors, "Software Construction Using Components", Technical Report 160, Department of Information and Computer Sciences, University of California, Irvine, 1980

[Prieto-Diaz 1990] Ruben Prieto-Diaz, "Domain Analysis: An Introduction" , ACM SIGSOFT, Software Engineering Notes vol15, no2, April 1990, page 47-54

[Shlaer and Mellor 1989] Sally Shlaer, and Stephen J.Mellor, "An Object-Oriented Approach to Domain Analysis", ACM SIGSOFT, Software Engineering Notes Vol.14, No.5, 1989

[Stropky and Laforme 1995] Maria E. Stropky, and Deborah Laforme, "An Automated Mechanism for effectively applying Domain Engineering in Reuse Activities", The Army Reuse Center News, Dec. 1995 Vol.4, No.4

[Tracz 1995] Will Tracz, "DSSA Pedagogical Example", ADAGE-LOR-94-13, April 1995

[Tracz and Werkman 1995] Will Tracz, and Keith Werkman, "What is DSSA?", WWW DSSA Home Page (http://www.sei.cmu.edu/arpa/dssa/dssa-adage/what-is-dssa.html), Oct 1995

[Wirfs-Brock et al. 1990] R. Wirfs-Brock, B.Wilerson, and L.Wiener, "Designing Object-Oriented Software", Prentice Hall, 1990