Proposal of Information System Implementation
Method using Conceptual Data Modeling and Responsibility-Driven Design

Yu Okada† Shigeo Kaneda†
† Graduate School of Engineering, Doshisha University
E-mail: † yokada@ishss10.doshisha.ac.jp

Abstract The concept of object-orientation is a powerful tool for design information systems. However, it is difficult for inexperienced software engineers to maintain data consistency in applications, to find appropriate relationships among stakeholders, and to reflect the demands of application domain experts to the design. To resolve these problems, this paper proposes a new method that combines Conceptual Data Modeling and Responsibility-Driven Design into a consistent and systematic development method. The output of object-oriented analysis is directly reflected in the final implementation. The proposed method was experimentally applied in the design process of a content management system. Experimental evaluation shows that the proposed method clarifies the essential business targets and the appropriate object responsibilities.

Keyword Object-oriented Approach, Responsibility-Driven Design, Conceptual Data Modeling

1. Introduction

Object-orientation is one of the most powerful tools for design information systems. Many methodologies, which are based on the object-oriented concept, have been proposed and are widely used in development. However, it is difficult for inexperienced software engineers to appropriately develop software systems using object-oriented concepts. When they misunderstand or fail to sufficiently understand the concept of object, they often design information systems whose maintenance will be difficult.

To solve this problem, Responsibility-Driven Design (RDD), advocated by Wirfs-Brock, has received much attention because it is suitable for various kinds of system development projects. RDD is a modeling method that not only covers all use cases but also considers the responsibility of the thread that runs through every activity: this is our focus on software responsibilities [1]. Specifically, our method clarifies problems using CRC cards and role stereotypes and defines and identifies the objects that fulfill their responsibilities.

But since RDD’s main purpose is software design, it doesn’t have enough power for object-oriented analysis. RDD focuses on the design process using CRC cards and design stories and looks at and refers to system responsibility. Thus for the process of object-oriented analysis, it may not be suitable to find appropriate relationships among stakeholders, to understand its departments, and to refer to the possibility of business reform.

To resolve the problem, this paper proposes a new method that combines Conceptual Data Modeling (CDM), which was proposed by the Manufacturing Architecture for Series Product (MASP) [2], and RDD. If a designer focuses on CDM’s important Monos and Kotos, he/she can find appropriate relationships and reliably and flexibly establish an information system. CDM was successfully applied to the development project of JFE Steel and KDDI Corporations [5][6]. Using CDM in object-oriented analysis can locate market demands, and the results cover each Mono and Koto in RDD. The proposed method has been experimentally applied to the design process of a WEB content management system.

In Section 2, we briefly define CDM and RDD. Based on the explanations of CDM and RDD mentioned in Section 2, Section 3 presents our models and a methodology that builds upon and uses notations used in CDM and RDD. Then we give an overview of related works in Section 4 and consider our model’s effectiveness in Section 5. A conclusion with a summary and future works is provided in Section 6.

1 The word “Mono” means something valuable that has a existence though it needs not be a material existence.
2 The word “Koto” means a state desired to be in the system.
2. Brief introduction to RDD and CDM

RDD is a practical approach to object-oriented design based on concrete principles. RDD moves the design viewpoint from mathematical or algorithmic viewpoints to autonomous collaborative objects like human beings who have responsibility for certain actions. RDD is a software design methodology from the viewpoint of roles, responsibilities, and collaborations, CRC cards, and role stereotypes [1]. They are discussed as follows.

Since this RDD concept and method is mainly for software design, it is not adequate for object-oriented analysis or suitable for the process of object-oriented analysis to find appropriate relationships among stakeholders, to understand departments, and to find possible business reforms. Furthermore, an approach that simultaneously finds both domain and system objects in RDD may complicate object-orientation. Searching for candidate objects in design stories only focuses on Monos as well as previous methods.

2.1. Roles, Responsibilities and Collaborations
Responsibility means the obligation to perform a task or to know information that the object has to satisfy. Role denotes a set of related responsibilities. A collaboration means an interaction of objects or roles [1]. In RDD, an application system is designed by following a step-wise procedure:

1. An application needs to carry out the main responsibility.
2. We derive objects from the main responsibilities and assign to each object relatively minor responsibilities that are gotten by dividing the main responsibilities.
3. Then we assign to each object a new responsibility that we cannot find in Step 2.
4. Then we set the role as a set of responsibilities of each object and reconsider the essentials of each object.
5. Roles collaborate with other objects to carry out their responsibilities. Due to the evolution of a development model, Step 3 will be conducted again.
6. These objects work together to fulfill the wider responsibilities of the application.

Figure 1 illustrates the basic concept of RDD.

2.1.2. Design story
The design story is explained as follows [1]. It describes the rough outline of the critical aspects of the application and the design ideas and provides a framework for locating potential candidates of the objects. After identifying the candidate objects, they will fall into place and support various aspects, which include use cases, requirements, architecture, users, or sponsors. The candidate objects pull all these aspects together and delineate the original description.

2.1.3. Role Stereotypes
The Role stereotypes are explained as follows [1]. A well-defined object supports a clearly defined role. Using purposeful oversimplification or a role stereotype increases the focus on each object’s responsibilities. Our goal is to build consistent and easy-to-use objects. Attaching a stereotype to each object is suitable in the design process. This is partly because we can focus on the behavior of objects by ignoring their details. The following six stereotypes are used in RDD:

1. Information Holder: knows and provides information for collaboration
2. Structure: maintains relationships between objects and information about those relationships
3. Service Provider: performs tasks and generally offers computing services
4. Coordinator: delegates events to other objects
5. Controller: makes decisions and directs the actions of others
6. Interfacer: transforms information and requests between different parts of the system

Figure 1: a basic concept of RDD
2.1.4. CRC card
CRC stands for “Candidates, Responsibilities, and Collaborators” [1]. An CRC card is a physical paper item, so it is easy to handle, but it is still a low-tech tool for exploring initial design ideas. If CRC cards are used in the design process, people unfamiliar with UML can design because the system designer doesn’t have to expressly write multiplicities and associations as classes. Thus application domain experts can participate in software design, and the design result is intuitively easy to understand.
Simulating use cases with CRC cards improves the system designers’ understanding of the system and effectively accomplishes the design idea. The final result of the arrangement of CRC cards is used for designing the class structure.

2.2. CDM
Conceptual Data Modeling (CDM), proposed by the Manufacturing Architecture for Series Product (MASP) [2], focuses on important Monos and Kotos [3][4]. CDM uses the following diagrams for designing information systems:

1. Static Model diagram: represents the entity relationship relation of the Monos
2. Dynamic Model diagram: describes the Kotos of the Mono
3. Organization Model diagram: shows relations between organizations
4. Function Model diagram: shows the detailed functions excluding the above models

Among the above diagrams, the static model, the dynamic model, and the organization model diagrams play essential roles for business process modeling. We used them to analyze a business model in our paper.

When we analyze a business, designing the appropriate granularity of the Mono is one of the most essential problems. If two Monos have identical state changes in the dynamic model diagram, they will be treated as identical. A slight difference of state can be distinguished by the value of the attributes of identifiers. Moreover, the range of the Koto decides the granularity of the Mono. Concerning Koto, we focus on the specific Koto that will rewrite the attribute values of Mono.

After drawing these diagrams, we focus on the organization that uses the data and has responsibility and find an appropriate workflow that represents not “how it is” but “how it should be” [3]. As mentioned above, CDM is just a business analysis model among stakeholders. The application system implementation process is not represented in CDM. Thus, for designing information systems, establishing the process model is necessary using CDM.

3. Proposed Method
Finding and defining objects are established by writing a design story and CRC modeling in RDD. The design story describes the essential aspects of the application, but it is difficult to derive them from the analytical phase.
Furthermore, the approach for simultaneously finding both domain responsibilities and system responsibilities by CRC modeling is slightly complex.
Thus, we derive the essential business demand by CDM that focuses on both Mono and Koto and apply this result to RDD and to object-oriented analysis by writing the design story. We show the process of the proposed method using CDM and RDD and its detail steps in Fig. 2.
[Step 1]
First, we carry out CDM by setting appropriate stakeholders from data derived from job analysis after personal interviews with experts and decide the domain and the appointed work of the business enterprise. We create the Static Model diagram from Monos and the Dynamic Model diagram from Kotos in the real world. Then we derive the Organization Model diagram from Monos and Kotos. After modeling, by comparison with the real world and the Organization Model diagram, we modify their diagrams to clarify the responsibilities of the domain objects in the application.

[Step 2]
Based on a Dynamic Model diagram and an Organization Model diagram, we carry out RDD. Specifically, we write use cases and a design story. When writing a design story, we can compensate for the scope of the business and then concentrate on creating a design idea as a connection for understanding the workflow by referring to the Static Model diagram, which provides useful hints to build the idea.

After writing the design story, the designer can identify important themes including the system logic to provide time to think about a concept derived from CDM and how to connect it to the system. We find candidates of objects that represent functions performed by the software, things our software affects, information, control, decisions to make, and structuring methods. Moreover, since the Monos derived from CDM are domain objects with responsibility, we make them candidates as often as possible.

[Step 3]
We transform the object candidates derived in Step 2 into design objects. First, we use the CRC cards to record the knowing and doing responsibilities and connect the candidates to objects performed by the software. When describing the CRC cards, we should give an appropriate name and characterize it as fitting one or more stereotypes.

Second, we carry out CRC modeling, and note the following important point. If we use the Mono as an object candidate, we assign it to the Information holder, the Structure, and the Service Provider because the Controller and the Coordinator are concerned with system control and the Interfacer deals with information and requests between distinct parts of our system; it is not derived from CDM. In particular, if a Mono only knows information, it is an Information holder. If a Mono manages its relations or organizes large numbers of them, it is a Structure. If a Mono comes to realize some Kotos, it is a Service Provider. When writing the responsibilities, the Mono attribution and identifier describe the responsibilities.

As above, we simulate the details of the functions and define the preliminary classes and sequence diagrams. If objects are assigned appropriate roles and the collaboration model are clear and simple, the phase moves to Step 4.

[Step 4]
We conduct design refinement so that the results are more predictable, consistent, flexible, and understandable. We identify control style and decide static and dynamic visibility relationships between objects. Finally, we define the class definitions and diagrams.

4. Prototype Experiment
To verify the validity of our proposed method, we applied it to the design of a web application system for a local government. In our system development, we used Ruby on Rails, which is widely used for rapid development [7]. As a feature of this system, the contents were complicated because they interacted with each other. We had to lay an expandable information system base, so we needed to precisely conduct object-orientation.

4.1. Evaluation of only RDD
At first, we didn’t use CDM or analysis through interviews about the application domain uses. We set following contents.

1. Era travel: cross-references for each period with the modern era using historical maps
2. Local government map: expressing sightseeing resources using current maps
3. Promenade: proposal recommended by local government experts

From the analysis results, we wrote use cases, carried out an RDD, and finally made a class model (Fig. 3).
As shown in Fig. 3, there were 10 classes, 29 attributes, and 23 methods. But since this result was only from analysis through interviews, it lacked validity.

4.2. Evaluation of proposed method

To review the essential business model, we analyzed this system again using CDM. First we analyzed it from the domain and the appointed work of a business enterprise. After each diagram had been made, we finally made the following Static Model diagram (Fig. 4).

From this diagram, the concepts were established to focus on the business. For example, the stakeholder covers the inhabitants and clarifies the flow of information between the present and the past. We derived Monos with designed attributions, like place to see (present day) in information (present day). Furthermore, we clarified a viewpoint only with the web system, like “Promenade was based on the role as a bridge with the present and the past.”

We applied the above approach to make a design story and derived object candidates. We set the role stereotype to be an Information holder or a Structure if the candidate was an important Mono; if it was a Mono about important Kotos, it was a “Service Provider”. We wrote them on CRC cards, took their attribution, identification, and association as responsibilities for the system, and included them in the cards. We characterized them using role stereotype, distributed application control, and revised the model. We show the detailed class diagram in Fig. 5.

As shown in Fig. 5, there were 18 classes, 64 attributes, and 36 methods. The number of attributions and methods is larger than in only RDD. Eight classes were newly derived. As a result of the CDM of Fig. 3, deriving the domain objects was finished, and thus we were able to complete the object-oriented design simply by adding such system objects as image classes and the location of maps.

5. Discussion

The experiment of the previous chapter produced very different results between only RDD and the proposed method. This confirmed that we can use CDM in object-oriented analysis. In analysis by only RDD, the objects called “sightseeing” weaves history with the present. But we cannot derive objects concerning inhabitants, which are important targets in government A, because we cannot clarify the business targets. All contents were independent by only RDD.

But with the proposed method, we clarified the relationship of the contents to derive and to divide the objects. This is because we were conscious of the Koto by the Dynamic Model diagram in CDM. Moreover, we derived responsibilities and defined objects with relationships between organizations from the separation process. We also realized that the contents had to interact with each other and became conscious of the flow of information. Research candidates for this time were relatively modest CMS, but we could determine “how the web system should be” by using CDM. In the design of the domain layer, the CDM role focuses on important Monos and Kotos for business. RDD plays a role in which we add the design viewpoint and branches and leaves as responsibilities in the system.

In summary, we realized an RDD that included business aspects and set new classes, attributions, and methods with business logic that were not only derived by RDD. Moreover, CDM and RDD maintained each role. We also
proposed a method to connect the analysis and design process compatibilities.

6. Conclusion

Object-orientation is one of the most powerful tools in design information systems. Many methodologies based on the object-oriented concept have been proposed that are widely used in development. However, it is difficult for inexperienced software engineers to appropriately develop software systems using the object-oriented concept.

To solve this problem, Responsibility-Driven Design (RDD) advocated by Wirfs-Brock has received much attention and is suitable for various kinds of system development projects. But RDD focuses on the design process, so it may not be suitable for the process of object-oriented analysis for the possibility of business reform.

To resolve the problem, this paper proposed a new method that combined Conceptual Data Modeling (CDM), which was proposed by the Manufacturing Architecture for Series Product (MASP), and RDD. Moreover, we apply our proposed method to a system requested by a local government of Kyoto. We found an appropriate workflow that represented web systems and carried out RDD with both important Monos and Kotos. In conclusion, we realized an RDD that included the business aspects. Our proposed method is an efficient approach output of object-oriented analysis that is directly reflected in the final implementation.

In this paper, our CMS is relatively modest, but CDM is found in a certain grain size of the target business, and thus we must consider what remains to be seen when it is implemented in large-scale systems.

References