VNU Journal of Science, Natural Sciences and Technology 26 (2010) 89-94

Specifying Object-Oriented Design Patterns using OWL

Vu Dieu Huong[1,*], Nguyen Van Vy[1], Le Viet Ha[2]

[1]*College of Technology, VNU, 144 Xuan Thuy, Hanoi, Vietnam*

[2]*Information Technology Institute, VNU, 144 Xuan Thuy, Hanoi, Vietnam*

Received 20 July 2009

Abstract. Design patterns provide good solutions for problems occurred in the design stage. Using design patterns in the software development processes helps improve productivity and quality of software products. Pattern Oriented Analysis and Design Process has four steps related to patterns, namely the acquaintance with design patterns in the pattern library, the retrieval of the pattern candidate, the selection of suitable patterns and the application of selected patterns. It is necessary to have a method to specify patterns in a machine understandable form to automate the above four steps. Some works have tried to specify the structural aspects of design patterns by ontology. We add the specification of the behavior aspects into design pattern ontology so that this ontology can be used to automate steps in the Pattern Oriented Analysis and Design Process.

Keywords: design pattern, OWL, Ontology.

1. Introduction

Pattern Oriented Analysis and Design Process (POAD) is a systematic process that promotes pattern-based development [1]. POAD consists of four important steps: acquaint with design patterns in the reusable asset library, retrieve of the pattern candidates, select suitable patterns and use the selected patterns.

Design patterns are usually described in the unstructured documents. A pattern document includes many sections such as name of pattern, intent, motivation, applicability (applicability context of patterns), structure, implementation and consequence [2].

Design patterns are also stored in some databases or expressed by some means. In recent years, many design pattern libraries have been built [1].

The acquaintance activity in POAD includes browsing catalogs of patterns that are stored in libraries for the purpose of understanding existing patterns. In this step, we focus on intent sections and applicability sections of the pattern documentations.

The retrieval activity in POAD is defined as selecting patterns from the library. The selected patterns are those produce solutions for application requirements. Input of this step is the set of application requirements. The outcome of this step is a set of pattern candidates. This set of pattern candidates is used as input of the selection activity to select the suitable patterns to pass to the next step.

* Corresponding author. Tel.: 84-4-37549016.
E-mail: huongvd@vnu.edu.vn

In the traditional way, to carry out the acquaintance and the retrieval activities, all patterns need to be read and the intent section and the applicability section in document are focused. Identifying and retrieving set of pattern candidates are performed manually. These activities take much time and effort. So, it is necessary to automate these activities. If the retrieval step is automated then the acquaintance step isn't necessary in POAD.

To retrieve set of pattern candidates for a software system automatically, we need a technique that allows comparing automatically the requirement specification of the software system with the context in which patterns should be applied (the applicability context of patterns). However, the comparison depends on the specification method for the software requirements and the applicability context of patterns.

There were some specification methods for Design Patterns (Patterns). Gamma et al described twenty three patterns using text and UML [2]. Maplesden introduced a set of graph notations to model patterns [3]. These specifications assist us in understanding patterns.

Many researches specified patterns using different formal specification languages such as LePUS, Slam-Sl [4-6]. Using the formal specification languages, we can specify different aspects of patterns. However, it costs much time to study formal methods and it is very difficult to understand formal specifications of patterns for new users.

Dietrich specified design patterns using OWL [4]. OWL is quite similar to Object Oriented modeling languages. Therefore, it is easier for us to use OWL to specify patterns and understand the OWL specification of patterns as well. However, Dietrich just specifies the structural aspect of patterns.

In this paper, we propose a method to specify both the structural aspects and the behavior aspects of patterns using OWL. Our idea is specifying the applicability context of design patterns in the form of object model to compare the application requirements with the applicability section of design patterns automatically. This technique can therefore assist in retrieving pattern candidates.

The rest of this paper is organized as follows. Section 2 introduces an overview of Ontology and OWL. Section 3 and section 4 represent our contribution, a method to specify patterns and retrieve the pattern candidates from ontology. We also illustrate in these sections our approach's implementation and an example of retrieving the pattern candidates from our ontology to be used in refining Customer-Account management system. The last section gives some perspectives and concludes the paper.

2. Ontology and OWL

Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them [7].

We develop ontology for purposes:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit

OWL (*Ontology Web Language*) is language for representing ontology [5]. It is a powerful language to represent knowledge in a machine understandable form based on a simple data model using linked resources.

Most of the elements of an OWL ontology concern *classes, properties, instances* of classes (*individuals*), and *relationships* between these instances.

The data described by an OWL ontology is interpreted as a set of "individuals" and a set of "property assertions" which relate individuals to one another. An OWL ontology consists of a set of axioms which place constraints on sets of individuals (called "classes") and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided.

OWL is the semantic web mark-up language. A big advantage of it is openness. Therefore, we can share knowledge in the OWL ontology via internet and new knowledge can also be added easily.

3. The Design Pattern Ontology

We develop the design pattern ontology to share design patterns and to share experience in using design patterns.

We can use some of languages to develop ontology such as LOOM, LISP, XML, SHOE, OIL, DAM+OIL, RDF, RDFS, OWL. In this research, we use OWL to build the design pattern Ontology.

3.1.Developing the design patterns ontology using OWL

The design pattern ontology is defined with classes, namely *DesignPattern, Catalog, Participant, Operation and ApplicationClass.* The class Catalog classifies a pattern according to different categories. The class Participant specifies information about participants in patterns. Methods of these participants are specified by the class Operation. The class Participant and the class Operation represent the collaboration of participants and therefore represent the behavior aspect of patterns. The class ApplicationClass describes the context where patterns should be applied. Attributes of classes in ontology are shown in the table 1.

Table 1. Properties of classes in DP Ontology

Classes	Attributes	Type of Value	Range
Design Pattern	Intent	DataTypeProperty	Text
	InCatalog	ObjectProperty	The class Catalog
Catalog	Decription	DataTypeProperty	Text
Paticipant	OfDesign Pattern	ObjectProperty	The class DesignPattern
	isAbstract	DataTypeProperty	Boolean
Operation	OfPaticipant	ObjectProperty	The class Paticipant
	isAbstract	DataTypeProperty	Boolean
	OfAppClass	ObjectProperty	The class ObjectProperty
ApplicationClass	OfDesignPattern	ObjectProperty	The class DesignPattern

In the pattern document, the context is described by text. However, in our approach, applicability contexts of design patterns are specified visually by a set of objects and collaborations among them. The applicability context of design patterns is specified by objects which are instances of the class Class and the collaboration of these objects. For example, the specification of the applicability context of Composite pattern is presented visually by the class diagram in the figure 1. There are whole class (e.g. PICTURES) and partial classes (e.g. LINES and CIRCLES). The relationship among these classes is composite

relationship. Concretely, PICTURES is a composition of LINES and CIRCLES.

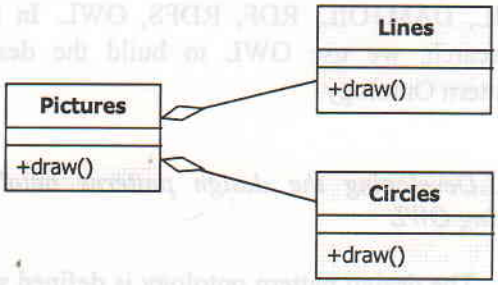

Fig. 1. Class diagram represents the applicability context of Composite pattern.

3.2. Retrieving design patterns from Ontology

Artifacts of the analysis phase in the object oriented development process include simple class diagrams and simply interactive diagrams. These diagrams are results of modeling application requirements. We call these diagrams as the initial diagrams of the design phase in the development process and they are inputs of retrieval step in POAD. The designers need to retrieve design patterns which produce solutions to refine these initial diagrams.

The applicability context of a design pattern is specified by class diagram in which classes are composed of attributes and operations. So, we can select the pattern candidates automatically by comparing classes in the initial diagrams with classes in applicability contexts of design patterns to find the commonity. We can obtain that by comparing such kinds of relationships as dependence, composite, and inheritance, etc. In addition, we can compare operations like creation (constructors) and/or deletion (destructors). We can also consider types of operation (abstract, concrete), the execution order of operations or parameters and return values, etc.

4. Implementation

We can use some tools to build, edit and update the ontology, such as OntoEdit, OilED, WebODE, Chimera DAG_Edit and Protégé.

In this research, we used Protégé 3.3.1 to develop Design Pattern Ontology. Protégé is a free, open source ontology editor. The Protégé platform supports two main ways of modeling ontology via the Protégé-Frames and Protégé-OWL editors. Protégé ontology can be exported into a variety of formats including RDF(S), OWL, and XML Schema.

We specified object oriented design patterns of Gamma et al [2]. A design view of Design Pattern Ontology is shown in the figure 2.

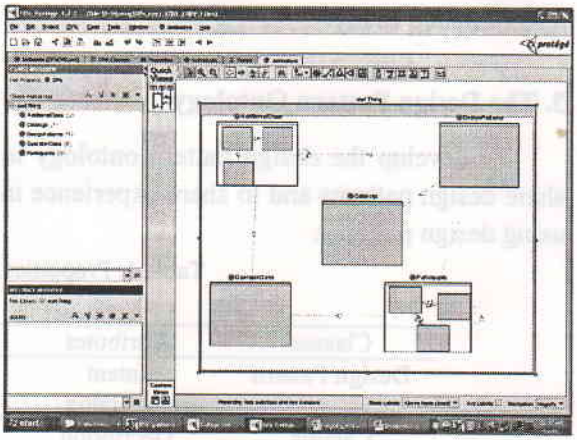

Fig. 2. A view of the Design Pattern Ontology.

In order to illustrate our approach about how to retrieve patterns from our ontology, we execute the retrieval activity on an example. It aims to refine a design of Customer-Account management system in a bank. The initial design class diagram of the system is represented in the figure 3:

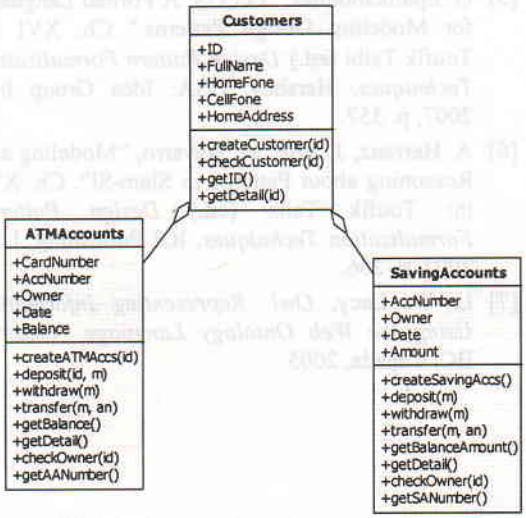

Fig 3. A class diagram of Customer-Account management system in a bank.

We need execute all queries which find out the common properties of elements in the figure 3 and elements in the specification of the applicability context of each pattern in DP Ontology.

In this example, the query of Composite pattern returns these match elements: the Customs class is markable with the Pictures class. The ATMAccounts class and SavingAccounts are markable with the Line class and Circles class. It means that the Customers class is the whole class, the ATMAccounts class and the SavingAccounts classes are partial classes.

Therefore, relationships of classes in the initial class diagram (presented in figure 3) and relationships of classes in the specification of Composite pattern have in common. This means that we detected the Composite pattern is a pattern candidate because it have the same structural properties correspond at the initial class diagram as illustrated in the figure 3.

Some patterns such as Iterator pattern, Abstract Factory pattern and From Abstract Classes to Interfaces can't be in the set of pattern candidates for this initial class diagram.

5. Conclusion and Future works

We proposed a specification method for design patterns using OWL. In this approach, we specified both the structural aspects and the behavior aspects of patterns and specified the applicability context of patterns by a set of objects and the collaboration among them visually.

This method allows us to share design patterns and to share experience in using these patterns. This also assist in retrieving set of pattern candidates which respond to a given software requirement.

We also developed a Design Pattern Ontology using OWL.

The difficult of our method is in specifying the applicability context of patterns. We need understand patterns to represent the applicability context of patterns visually. However, it is very easy for who develop patterns or who have experiment in using patterns.

Ontology query languages such as Ontology Web Language – Query Language (OWL-QL) and ontology query tools such as OWQL Query Service and Racer Manager Software have been developing. We are going to study on integrating Design Pattern Ontology with an existing ontology query tool, study on developing a new tool which support for selecting design patterns automatically in Pattern-Oriented Analysis and Design Process and then evaluate the productivity of retrieval.

Acknowledgements

This work is partly supported by the research project No. QCT 08.02 granted by Vietnam National University, Hanoi.

References

[1] S. M. Yacoub, H. H. Ammar, *Pattern-Oriented Analysis and Design: Composing Patterns to Design Software Systems*, Addison Wesley, U.S, August 28, 2003.

[2] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns – Elements of reusable object-oriented software*, Addison-Wesley, U.S, 1995.

[3] D. Maplesden, J. Hosking, J. Grundy, "A Visual Language for DP Modeling and Instantiation", *HCC'01, IEEE*, 2001, p. 338.

[4] J. Dietrich, C. Elgar. "A formal description of Patterns using OWL", *ASWEC'05*, IEEE, 2005, p. 243.

[5] G. Epameinondas. "LePUS: A Formal Language for Modeling Design Patterns." Ch. XVI in: Toufik Taibi (ed.) *Design Pattern Formalization Techniques*. Hershey, USA: Idea Group Inc, 2007, p. 357.

[6] A. Herranz, J. Jose, M. Navarro, "Modeling and Reasoning about Patterns in Slam-Sl". Ch. XVI in: Toufik Taibi (ed.) *Design Patterns Formalization Techniques*, IGI Publishing, U.S, 2007, p. 206.

[7] L. W. Lacy, *Owl: Representing Information Using the Web Ontology Language*. Victoria, BC, Canada, 2005.

Đặc tả các mẫu thiết kế hướng đối tượng sử dụng OWL

Vũ Diệu Hương[1], Nguyễn Văn Vy[1], Lê Việt Ha[2]

[1]*Trường Đại học Công nghệ, ĐHQGHN, 144 Xuân Thủy, Hà Nội, Việt Nam*

[2]*Viện Công nghệ Thông tin, ĐHQGHN, 144 Xuân Thủy, Hà Nội, Việt Nam*

Mẫu thiết kế cung cấp các giải pháp tốt cho các vấn đề nảy sinh trong giai đoạn thiết kế hệ thống. Tiến trình phân tích, thiết kế hướng mẫu (POAD) là tiến trình phần mềm hướng đến mục tiêu tăng khả năng sử dụng mẫu thiết kế. POAD có bốn bước liên quan đến mẫu: *làm quen* với các mẫu trong thư viện mẫu, *lấy ra* các mẫu ứng viên phù hợp với hệ thống hiện tại, *lựa chọn* mẫu phù hợp nhất trong danh sách mẫu ứng viên, *sử dụng* các mẫu đã chọn để thiết kế hệ thống. Hiện tại, chúng ta vẫn thực hiện bốn bước này bằng tay. Để tự động hoá các bước này, chúng ta cần có một phương pháp đặc tả các mẫu theo cách mà máy tính có thể hiểu được để nó có thể trợ giúp chúng ta thực hiện các hoạt động với các mẫu. Trong nghiên cứu này, chúng tôi sử dụng OWL – ngôn ngữ sử dụng để xây dựng ontology trên web, để đặc tả cả khía cạnh cấu trúc và hành vi của các mẫu để chúng ta có thể tự động hoá các bước trong POAD.