The Reflection pattern in the immune system

Junichi Suzuki
Department of Computer Science,
Graduate School of Science and Technology,
Keio University
Yokohama, 223-8522, Japan
+81-45-563-3925
suzuki@yy.cs.keio.ac.jp

Yoshikazu Yamamoto
Department of Computer Science,
Graduate School of Science and Technology,
Keio University
Yokohama, 223-8522, Japan
+81-45-563-3925
yama@cs.keio.ac.jp

Abstract

Reflection is a design principle that allows a system to have a representation of itself in the manner that makes it easy to adapt the system to a changing environment. It has now matured to the point where it is used to address real-world problems in various areas. This principle has been also identified as a pattern of software architecture (PoSA) [1]. This paper presents a non-software examples of the Reflection pattern. We present a simple and household example and an example in the immune system. We believe our work helps understand the intent, structure and consequences of the Reflection pattern.

1. Introduction

Reflection is a design principle that allows a system to have an explicit representation of itself in the manner that makes it easy to adapt the system to a changing environment. It was originally introduced by 3-Lisp [2] and has been studied within various programming languages. Recently, it has been applied to more generic system designs such as databases, concurrent/parallel computing, operating systems and distributed computing. As this principle has now matured to the point where it is being used to address real-world problems, it is identified as a pattern of software architecture (PoSA) [1]. This paper presents two non-software examples of the Reflection pattern: a simple and household one and one in the immune system.

The remainder of this paper is organized as follows: Section 2 introduces the essential concepts and terms related to the reflection pattern. Section 3 presents examples of the pattern.

2. The essence of Reflection

The base unit of computation in object-oriented systems is *object* (or *baseobject*). Through the interaction among objects, a system computes the certain task. The reflection pattern generally introduces the notion of *object/metaobject separation*. A *metaobject* (or *metalevel object*) is an object that contains information about the internal structure and/or behavior of one or more baseobject. In other words, metaobjects can track and control certain aspects (i.e. structure and/or behavior) of baseobjects. A set of metaobjects is called a *metalevel* or *metaspace*, and a set of baseobjects is called a *baselevel*.

Reflection is the ability of a program to manipulate as data something that represents the state of the program [3], and to adjust to changing requirements. The goal of reflection is to allow a baseobject to reflect on its own execution state and eventually alter it to change its meaning. In contrast to reflection, *reification* [4] is the process to make something accessible that is normally unavailable in the baselevel or is hidden from the programmer. For the execution of a baseobject to be supervised, it must first be reified into the corresponding metalevel. A set of interfaces with which a baseobject interacts with its metalevel is called *Metaobject Protocols* or *MOPs*. The relationship among the constructs described above is illustrated in Figure 1.

There are various approaches to achieve the reflection, and they can be classified with some perspectives: compile-time and runtime reflection [5], structural and behavioral reflection [6] and introspection and intercession [7]. The essential principle of the reflection, however, is to reify the information of a component in a system and use the reified information from the component or another component.

3. Examples of the Reflection pattern

3.1 An example in using a mirror

The first example of the reflection pattern is a simple and household one in using a mirror. We use a mirror to shave or cosmetize. The mirrored face represents the state of the real one. We shave or cosmetize seeing it, because we cannot see the real face directly.

In terms of the reflection pattern, a mirror is a metaobject and a person in front of the mirror is a baseobject. A baseobject can determine its behavior by referring its status

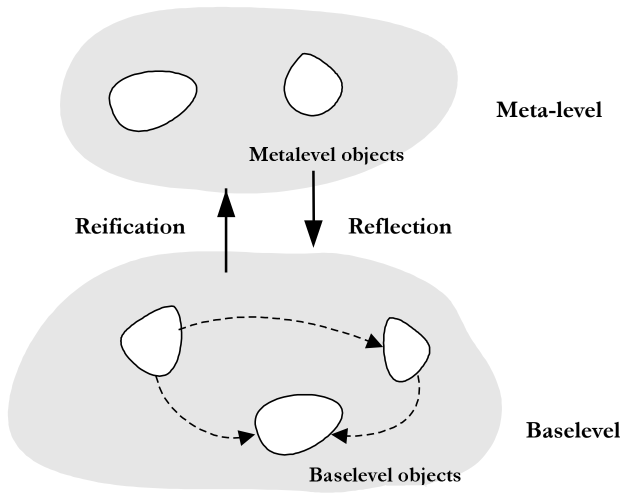

Figure 1: Typical reflective architecture.

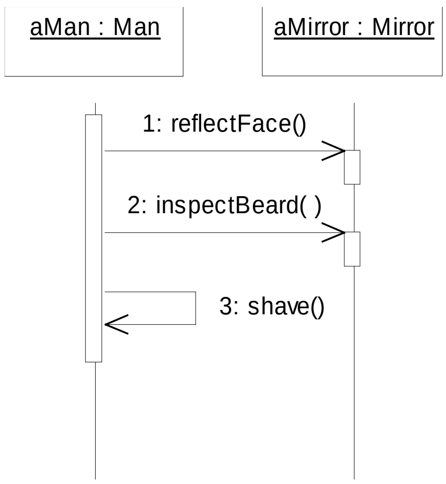

Figure 2: A sequence diagram for the reflection pattern with a shaving example.

with the corresponding metaobject, because the information on the baseobject is reified into the metaobject. Like this example, the reflection pattern can be applied effectively when a baseobject cannot or should not refer itself directly. Figure 2 shows a sequence diagram between a baseobject named `Man` and a metaobject named `Mirror`. The method `reflectFace()` reifies the `aMan` and creates `aMirror`. Standing in front of a mirror is considered as invoking this method. The method `inspectBeard()` checks whether the beard remains on the face by referring on the mirror. The three dimensional points are passed as parameters of this method. The method `shave()` shaves the beard in response to the result of `inspectBeard()`.

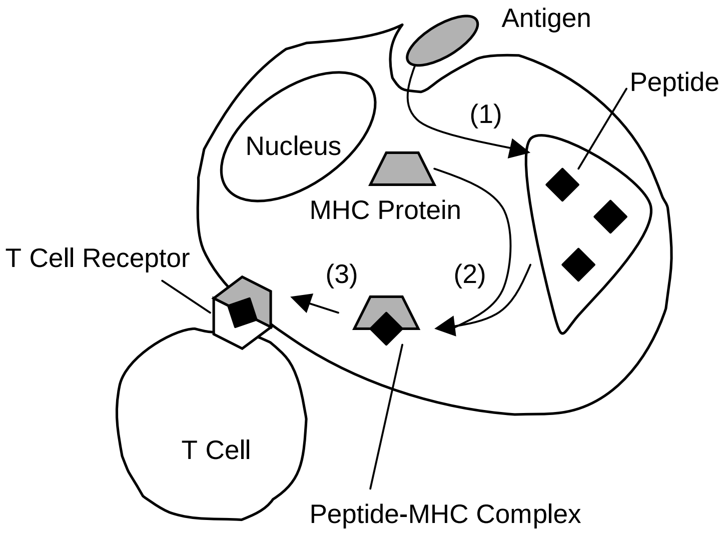

Figure 3: Processing and presentation of an antigen.

`reflectFace()` and `inspectFace()` are considered as metaobject protocols.

The type of reflection in this example is runtime and introspective. The binding between a baseobject and its metaobject is established at runtime, and a baseobject can observe its state though it cannot change itself.

3.2 An example in the immune system

The second example is in the immune system. The immune system can discriminate between foreign molecules (i.e. non-self) and the body's own cells and proteins (i.e. self). Once non-self is recognized, the immune enlist the participation of a variety of cells and molecules to mount an appropriate response in order to eliminate or neutralize it [8]. We focus on the mechanism how to recognize non-self called *antigens* (e.g. virus) and kill/reject them, because this mechanism is crucial to understand how the immune system works [9] and it is an example of the reflective capability.

The immune response involves two major groups of cells: *lymphocyte* and *antigen-presenting cell*. Lymphocytes are one of many types of white blood cells, and two major population of lymphocytes are *B lymphocytes* (a.k.a. *B cells*) and *T lymphocytes* (a.k.a. *T cells*). B cells have receptors on their surface, which can recognize antigens, and produce antibodies specific to the recognized antigen. T cells have receptors called *T cell receptor*, which can recognize antigen that is associated with cell-membrane proteins known as *major histocompatibility complex molecules* (MHC molecules). Once the T cell recognizes and interacts with an antigen-MHC molecule complex, the cell secretes various growth factor known collectively as *cytokines*. They activate B cells and helps T cells to kill the recognized antigen.

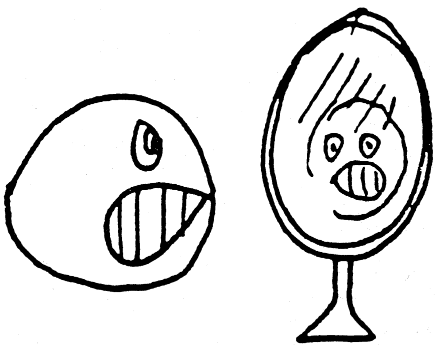

Figure 4: Umberto Eco's (author of “The Name of the Rose”) doodle of an immune system. A T cell is seeing self with a mirror (From [13]).

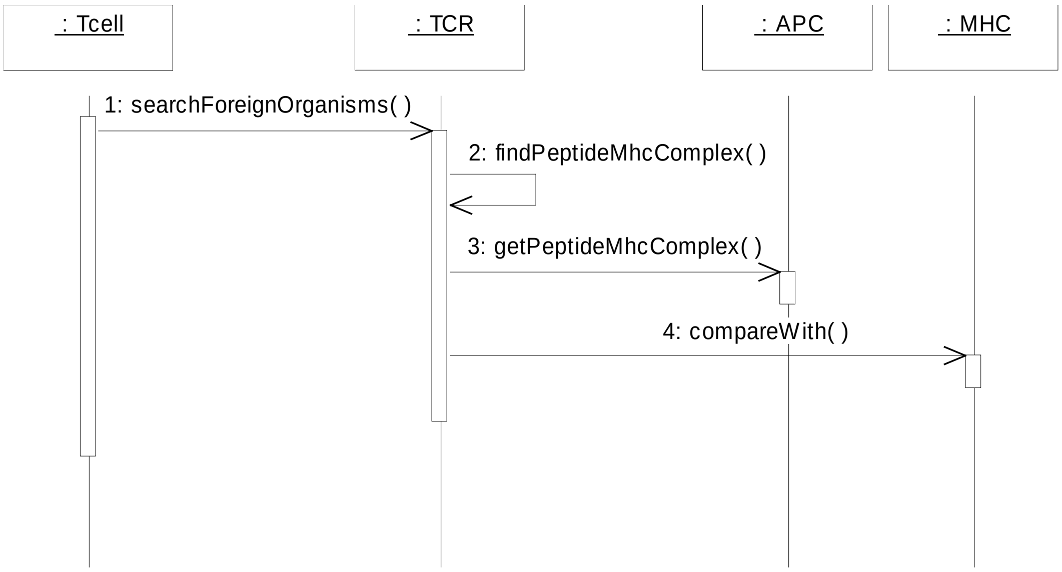

Figure 5: A sequence diagram for the reflection pattern with an example in the immune system to recognize antigens.

The recognition of antigen by T cells should be carefully regulated because an inappropriate T cell response to self-components can cause fatal autoimmune consequences (e.g. allergy) [8]. To ensure carefully regulated recognition, T cells only can interact with antigen that is displayed together with MHC molecules on the surface of *antigen-presenting cells* (APCs). These specialized cells, such as macrophages, roam the body ingesting the antigens.

Figure 3 depicts the process for an antigen-presenting cell to process and present an antigen. Once APCs find an antigen, it internalizes the antigen and fragments it into antigenic peptides (1) [10]. Pieces of these peptides are joined to MHC molecules (2) and are displayed on the surface of the cell (3). The T cell recognizes and interacts with this peptide-MHC molecule complex using its receptor (T cell receptor). Like this, T cells do not interact with antigen directly. Instead, they can attach to the only cell displaying foreign antigen complexed to an MHC molecule. Such cells are called *altered self-cells* [8]. MHC is a tightly linked cluster of genes, and MHC molecules are encoded with them. MHC genes are the same in all of an individual’s cells but differ from person to person. T cells can recognize the genetic difference in MHC molecules and does not attack cells encoded the same type of MHC [11].

The above recognition process can be considered as reflecting self in a mirror and compare it with a foreign component to discriminate self and non-self (Figure 4). In this metaphor, MHC molecules play a role of the mirror that T cells can refer to. In the context of the Reflection pattern, MHC molecules are metaobjects and APCs and other lymphocytes are baseobjects because MHC controls the major immune phenomena including the recognition of antigen by T cells and the activation of B cells.

Figure 5 is a sequence diagram that shows the recognition process of antigens. The class `MHC` is a metaobject that knows the information on self cells. The other classes are baseobjects. The class `TCR` represents the T cell receptor. `TCR` begins to search a peptide-MHC molecule complex when the method `searchForeignOrganisms()` is invoked. It calls `findPeptideMhcComplex()` to find a peptide-MHC molecule complex with its receptor. This method can implement an asynchronous polling loop. If `TCR` finds a peptide-MHC molecule complex, an instance of `APC` that the complex is displayed on is returned. Then, `TCR` invokes `getPeprideMhcComplex()` to obtain the found complex, and calls a metaobject protocol `compareWith()` to compare the self MHC with found one. If the difference is recognized, `searchForeignOrganisms()` returns the instance of APC and the subsequent immune responses are started.

The type of reflection in this example is compile-time and introspective. The binding between a baseobject and metaobject is statically established because the MHC genes are arrenged at the antenatal term. A baseobject can observe its property though it cannot change its metaobject.

4. Concluding remarks

This paper presents two real-world examples of the Reflection pattern: a simple and household example and an example in the immune system. We believe our work contributes to provide an insight and concrete understanding for the Reflection pattern.

5. Acknowledgements

We would like to thank Michael Duell and Linda Rising for their invitation to this workshop. We would also thank Hiroki Kamata for his initial input of this work.

6. References

[1] F. Buschmann et. al. A system of patterns: Pattern-oriented software architecture. WILEY, 1996.

[2] B. C. Smith. Reflection and semantics in Lisp. In *Proceedings of ACM POPL '84*, 1998.

[3] P. Maes. Concepts and experiments in computational reflection. In *Proceedings of OOPSLA '87*, 1998.

[4] D. P. Friedman and M. Wand. Reification: Reflection without metaphysics. In *Proceedings of Symposium on LISP and Functional Programming*, 1984.

[5] S. Chiba. A metaobject protocol for C++. In *Proceedings of OOPSLA'95*, 1995.

[6] J. McAffer. Engineering the meta level. In *Proceedings of Reflection '96*, 1996.

[7] D. Bobrow, R. Gabriel and J. White. CLOS in context - The shape of the design space.. In Object-Oriented Programming -The CLOS perspective, Chapter 2, MIT Press, 1993.

[8] J. Kuby, Immunology, third edition. Freeman, 1997.

[9] C. A. Janeway, Jr. How the immune system recognize invaders, In *Scientific American*, September, 1993.

[10] G. J. V. Nossal. Life, death and the immune system. In *Scientific American*, September, 1993.

[11] I Roitt. Essential immunology, ninth edition. Blackwell Science, 1997.

[12] G. Kiczales et. al. The art of the metaobject protocol. MIT Press, 1991.

[13] E. E. Sercarz, F. Celada, N. A. Mitchison, T. Tada. The Semiotics of Cellular Communication in the Immune System. NATO Asi Series, Series H : Cell Biology, Vol 23.