On Aspect-Oriented Design
- Applying “Multi-Dimensional Separation of Concerns” on Designing Quality Attributes -

Natsuko Noda and Tomoji Kishi
Software Design Laboratories, NEC Corporation
(Igarashi Building) 11-5, Shibaura 2-chome,
Minato-ku, Tokyo 108-8557, JAPAN
TEL +81 3 5476 1089, FAX +81 3 5476 1113,
e-mail n-noda@ccs.mt.nec.co.jp

Abstract

“Multi-dimensional separation of concerns” is one of the techniques to solve large and complicated problems. We are studying aspect-oriented design (AOD), in which the technique is used to design software to attain required quality attributes, such as performance and reliability. In this paper, we briefly introduce AOD and discuss how “multi-dimensional separation of concerns” is applied to quality design.

1. QUALITY DESIGN

In software development, it is very important to design software to meet its goal on quality attributes[1], such as performance and reliability, as well as to realize the required functions. However, it is not easy to attain required quality attributes, and many problems, such as performance problems, arise in actual software development.

One of the difficulties in quality design is that we have to consider multiple requirements on quality attributes simultaneously. Separation of concerns is a technique to modularize the problem into handy chunks, and it can make software development easier. We are studying the design method in which we apply the technique to the quality design. In our method, what we call aspect-oriented design (AOD), we identify indispensable "aspects" that have to be considered when we design quality attributes, and examine each aspect separately instead of examining them simultaneously.

In this paper, we introduce AOD, and discuss how “multi-dimensional separation of concerns” is applied to quality design. In chapter 2, we explain the key idea of AOD. In chapter 3, we outline AOD using a simple example. In chapter 4, we have discussion on the relationship between AOD and the workshop theme.

2. ASPECT-ORIENTED DESIGN

2.1 Quality Design and Aspects

For each quality attribute, we empirically know that there exist indispensable software properties we have to examine, when we design software in order to meet its goal on quality attributes. For example, when we are to examine the performance, we may consider efficiency of data-access and communication behavior. We call these properties as aspects. Table 1 shows some examples of quality attributes and aspects.

Table 1. Quality Attributes and Aspects

quality attributes	aspects
performance	efficiency of data-access communication behavior …
reliability	accuracy of calculation hardware fault tolerance software fault tolerance reliability of data …

reusability	object creation object structure object behavior …
…	…

2.2 Design Steps

Though describing the detailed design steps of AOD[4] is out of scope of this paper, the basic design procedure of AOD is as follows (Figure 1): Firstly, for each aspect, extract the requirements on the aspect from the original problem statements. That is to abandon the information that has nothing to do with the aspect, and to pick up the necessary information to examine the aspect. Secondly, we design the software architecture for each aspect, namely, for each aspect, design architecture based on the requirements on the aspect, extracted in the previous step. Thirdly, we "weave" these architectures into the final architecture, considering the conflicts and tradeoffs between aspects. Unlike aspect-oriented programming (AOP)[3], this weaving process is manually done.

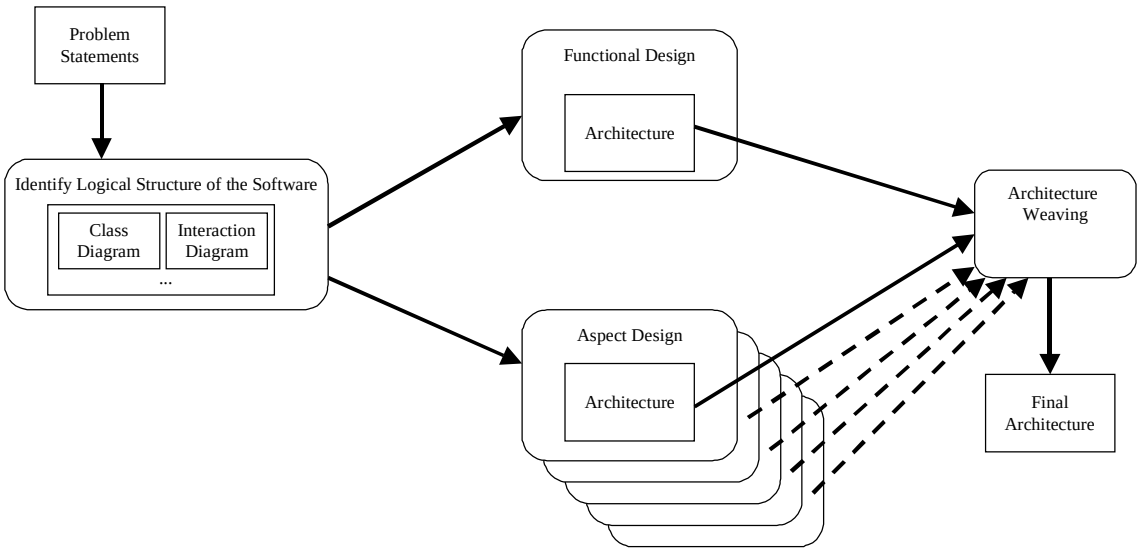

Figure 1. The Overview of AOD

3 AOD IN SIMPLE EXAMPLE

In this section, we explain AOD using a simple example.

3.1 Required Services

The example is design of software used in a car navigation system that retrieves information stored in a CD-ROM database and shows the position of the car. The information is about a map that consists of areas such as cities, and landmarks such as restaurants and shops included in each area. Table 2 shows the required services.

Table 2. List of Services

	Input (specified by user)	Output
S1	area name *x*, landmark name *y*	addresses of landmarks in area *x* whose name is *y*.
S2	area name *x*, landmark name *y*	descriptions of landmark in area *x* whose name is *y*.
S3	area name *x*, string *s*	list of landmark names in area *x* whose descriptions contains string *s*.
S4	-	current location

3.2 Identifying Aspects

Firstly, we have to identify aspects that have to be examined in designing this software. As there are requirements on performance and reliability for the software, we are to find out indispensable aspects that have to be examined when we consider the attainment of these quality attributes.

In this example, each retrieval service (S1, S2, and S3) is required to attain enough performance. As data-access is critical factor for performance (because access speed of CD-ROM is quite slow), data-access has to be examined carefully when we attain required performance. Therefore, we identify "efficiency of data-access" as one of the aspects. Similarly, we take "accuracy of calculation" as an aspect that is important to be examined when we attain required reliability.

3.3 Designing Aspects

In our method, we design architecture from each aspect separately.

Figure 2 shows the result of functional design. In functional design, we just concentrate on the realization of the required services, without considering the attainment of required quality attributes. In order to attain requirements on quality attributes, we design the software from each aspect separately.

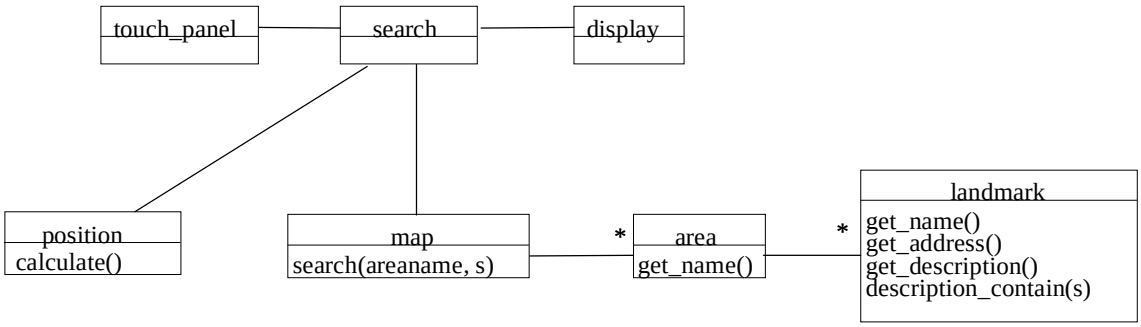

Figure 2. Functional Design

In aspect design, we firstly identify important factors that characterize each service from the aspect. Secondly, we decide the appropriate architecture for each service. For example, for "efficiency of data-access" aspect, access pattern (sequential, random, and so on), the arrangement of data in the data storage, and the number of data to be accessed is considered the important factor. On the contrary, meaning of the data is not important for the aspect. We characterize each service for this aspect, based on these factors. Table 3 shows the characterization of each service from data-access aspect.

Table 3. Characterization of Services from Data-access Aspect

	Access pattern	Arrangement	Number
S1	-	-	1
S2	-	-	1
S3	Sequential	In continuous sectors	100~10000

Figure 3 shows the architecture that is designed to attain the required efficiency of data-access. In this example, S3 has to sequentially access a large number of data that are placed in continuous sectors. Therefore, architecture is carefully designed to attain enough performance for this kind of data-access.

Similarly, figure 4 is the architecture that is designed to attain required accuracy of the calculation.

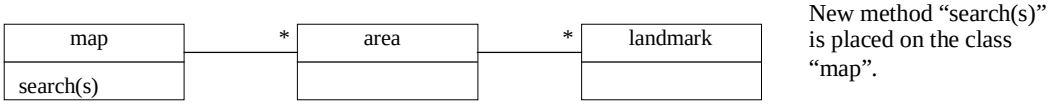

Figure 3. Designed Architecture from Data-access Aspect

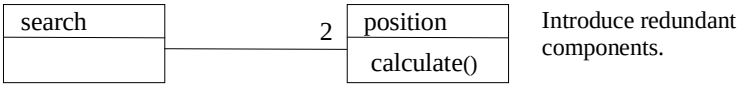

Figure 4. Designed Architecture from Efficiency of Calculation Aspect

3.4 Weaving Architectures

To get the final architecture, we "weave" architectures obtained by each aspect design and functional design. If there exists any conflict between the results of aspect design, considering tradeoffs and decide the final architecture.

Figure 5 shows the result of the architecture weaving for this example.

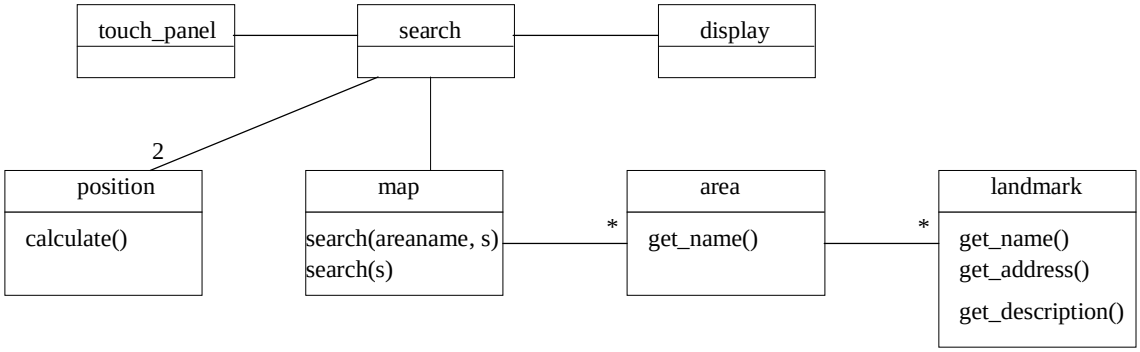

Figure 5. The Result of Weaving – The Final Architecture

4. DISCUSSION

In this paper, we outline our AOD, in which we apply “multi-dimensional separation of concerns” onto quality design. AOD is multi-dimensional separation of concerns in the following sense;

- It handles multiple arbitrary kinds (dimensions) of concerns, namely aspects. In AOD, aspects are determined based on the required quality attributes and characteristics of the target software and technologies used. Therefore, number and kinds of aspects are not pre-defined but decided by the designer in the design context.
- We can design each aspect independently, because each aspect design does not require any output from other aspect design.
- Though we can design multiple aspects simultaneously, they are not orthogonal and there can be conflicts between aspects. For example, to introduce the redundancy into the design to improve the reliability can reduce the performance. We have to consider these conflicts and find out the proper solution to accommodate them.

Though many researches, such as AOP and subject-oriented programming[2], mainly focus on applying “separation of concerns” on examining functions, our approach is to apply the technique on considering quality attributes. It may be difficult to find out what aspects are necessary to be examined when we design certain quality attribute, because we do not have clear understanding on the nature of the quality attributes. However, we empirically know important aspects to be considered for each quality attribute. Though these “knowledge” is neither exhaustive nor complete, we believe that we can utilize these “knowledge” to make AOD really work.

5. CONCLUSION

In this paper, we have introduced aspect-oriented design (AOD), in which “multi-dimensional separation of concerns” is applied onto the design of quality attributes. In AOD, we identify multiple aspects that have to be considered to attain required quality attributes, and examine each aspect simultaneously to design architecture that is suitable for the aspect. Architectures obtained by aspect design are woven into the final architecture, considering the conflicts between aspects. AOD is currently

in the experimental stage, and we have to examine many issues such as how to weave architectures, and how to solve the conflicts between the aspect. We believe AOD is one of the typical applications of "multi-dimensional separation of concerns". In the workshop, we wish to provide issues relate to the workshop theme and to discuss on these issues.

REFERENCE

[1] Bass, L., et.al.: Software Architecture in Practice, Addison-Wesley, 1998.

[2] Harrison, W. and Ossher, H.: Subject-oriented programming (a critique of pure objects), In Proceedings of the Conference on Object-Oriented Programming: Systems, Languages, and Applications (OOPSLA), Sep. 1993.

[3] Kiczales, G., et.al.: Aspect-Oriented Programming, In proceedings of the European Conference on Object-Oriented Programming (ECOOP), Jun. 1997.

[4] Noda, N. and Kishi, T.: On Aspect-Oriented Design – An Approach to Designing Quality Attributes -, In proceedings of the 6th Asia-Pacific Software Engineering Conference (APSEC '99), Dec. 1999. (To appear.)