Aspect-oriented requirement analysis based on Formal method

Can Qu \textsuperscript{1,a}, Xuhui Zhang \textsuperscript{2,b}, Haihua Chen \textsuperscript{3,c}, Lichen Zhang \textsuperscript{4,d}

\textsuperscript{1} School of Information City College of Huizhou Huizhou, China
\textsuperscript{2} School of Electrical and Computer Engineering Nanfang College Guangzhou Guangzhou, China
\textsuperscript{3} School of data science Guangzhou Huashang College Guangzhou, China
\textsuperscript{4} School of computers Guangdong University of Technology Guangzhou, China

\textsuperscript{a} 505609551@qq.com, \textsuperscript{b} 1451607395@qq.com, \textsuperscript{c} alice.hua@qq.com, \textsuperscript{d} lczhang@gdut.edu.cn

Abstract. Aspect-Oriented Programming (AOP) can effectively solve the code-tangling and code-scattering which are caused by crosscutting attributes. However, much work is presented on the programming and the implementation phase, and few works have addressed the early model stages. In this paper, we do research on the requirements engineering phase of software by using AOP technology. First in the requirements analysis phase, we identified the functional requirements and non-functional requirements by the separation of concerns, then to realize the functional requirements with components and the non-functional requirements with aspects. Then we represent the components and aspects by combining with the formal language Aspect-Z which have characteristics of accurate description; As actions accompany with conflicts while two or more aspects effect on the same joint point synchronously, we propose a method to solve by defining aspects priority level.

Keywords: ASEC-T-Z; formal verification; aspect oriented; reasoning proof.

1. Introduction

The software requirements phase is the first stage of the software development process. A well-defined requirements model avoids costly refactoring of the system in the later stages of design coding and implementation, thus reducing software development risk [1-2]. Software requirements include functional requirements and non-functional requirements. Functional requirement is the function that the software must realize, while the non-functional requirement is the quality attribute and function constraint provided to the software system to satisfy the user's business requirement, including the system's performance, reliability, security, guaranteeability and so on. Traditional demand specifications [3-4] often use non-formal abstractions language to describe non-functional requirements, but ambiguity-prone or neglected by developers, resulting in a mixture of functional requirements and non-functional requirements, making the code scattered, confusing, difficult to maintain.

Aspect-Oriented Software Programming (AOP) [5-7] separates functional requirements and non-functional requirements through crosscutting concerns. In the non-functional requirements, the aspect
is used to define and realize the cross-cutting concerns, the non-functional requirements of modular, loose coupling between modules to improve the software development process, improving system maintainability and reusability. Formal methods [8] have a description of the accuracy, no ambiguity, which is based on mathematics and used to describe the technology of the system nature. Formal specification method is through a clear mathematical definition of the semantics or language on software behavior for accurate, concise description. In the paper [10], an aspect-oriented non-functional requirements modeling approach is proposed, which extends UML to express aspect-oriented concepts. In the paper [11], the AOP technique is combined with the AspectJ language to analyze the system real-time system. In the paper [12], the Aspect-Z technology is used to analyze the simple example.

In this paper, AOP technology and the aspect of the formal language Aspect-Z are combined, and Aspect-Z is expanded. The formal specifications are used to the aspect-oriented requirements modeling, and the system can be non-functional requirements of the precise expression and processing, for the same joint point that needs to knit a number of aspects and aspects of conflict, through defining the priority of aspect to solve. The basic idea is to express the components and aspects through the Aspect-Z language, and realize the integration of the system through the weaving mechanism.

2. Related knowledge

2.1. The basic principles and related concepts of AOP

AOP is essentially a separation of concerns technique that achieves independent concerns in a loosely coupled fashion, and then combines these concerns to achieve the final system [3]. In traditional object-oriented development methods, these crosscutting concerns are scattered across multiple classes of software, as shown in Figure 1. The three different colors on the left of Figure 1 represent three different points of concern for the non-functional requirements of the system, which are scattered in the system. The programming or maintenance is not only cumbersome and easy to miss, but also brings to the development and maintenance of great difficulties. The right side of Figure 1 shows that AOP technology modularizes are used to divide 3 different concerns into 3 different aspects, which are encapsulated in crosscutting concerns, and are weaved into other functional modules. The maintenance and reuse of the program is becoming very simple, which is the essential characteristics of AOP.

Aspect-oriented Requirement Engineering (AORE) is a requirement engineering development method for AOSD. It defines and deals with requirements by separation of concerns, which separates functional requirements into the core concerns of the system, and non-functional requirements are separated into crosscutting concerns of the system by component language, encapsulated in the aspect to achieve some concern of the module unit. If there is no conflict between the components and the aspects, the two languages implement the system by the weaver weaving seamlessly; if multiple aspects need to be implemented at the same joint point of the component, it is necessary to determine the priority between the aspects to solve the conflict, and then weaving to achieve the system. As shown in figure2.

![Fig.1 The schematic diagram of cross cutting concerns in AOP](image-url)
2.2. Formal development methods and language
The formal method of software development is the Backus Normal Formula (BNF) proposed by Backus in the 1950s as a meta-language for describing programming grammar [7]. The basic meaning of Formal Method (Formal Method) is the use of mathematical methods to study the relevant issues of computer science. From a broad perspective, the formal method is the software engineering process analysis, design and implementation of systems engineering methods. In the narrow sense, it is the software specification and verification method. In this paper, the main discuss is the method of the software specifications. Formal specification is an accurate and concise description of the desired characteristics or behavior of the software by a grammatical and semantic method or language with well-defined.

Z is a formal language based on set theory and first-order predicate. It supports formal specification, normative reasoning and refinement of software, and is one of the most widely used formal languages so far. Schema is the basic structure of Z language, a better description of the software system's abstract state and operational functions. A pattern includes the name of the pattern, the declaration part, and the predicate part. The schema name (SchemaName) can be called anywhere in the specification process, and it can be used as a type name as well; Declaration of pattern introduce variables and their types, which are internal variables of the schema; Predicate describes the invariant relationship between the global variables among these local variables, or local variables before the model's declaration. As shown in figure 3.

![Fig.3 Original Z Schema](image)

3. Aspect-oriented formal language Aspect-Z
The Aspect-Z language is an extension of the traditional Z language by adding symbolic aspects [8]. In the phase of software development requirements analysis, a formal and complete modularization of the system is achieved by combining the formal language Aspect-Z and AORE. Aspect-Z adds the aspect mode to the Z-related mode of operation to indicate system-related properties and behavior.

The main idea of Aspect-Z support AOSD is based on requirement analysis by the concerns separation of components and aspect. Only considering the main steps of the operation, the operation mode of the functional requirement is set up as a component mode; the operation mode of the non-functional requirement is set up as an aspect mode, and the aspect mode is the crosscut of the component mode. The statement part of the aspect model can accurately describe the position of the joint point in the system, and the predicate part mainly expresses the function that needs to be realized at the joint point. Finally, the component mode and the aspect mode are integrated. The basic pattern of the Aspect-Z language is shown as follows [9]. As shown in figure 4.
When a pattern is named, it can be called in another mode by means of $\Delta$SchemaRef.

$\Omega$SchemaRef indicates that the current aspect pattern crosscut the schema named after SchemaRef.

Aspect-Z in the joint point is a precise point in the implementation of the system, which is the need to perform the relevant operations of aspect. PointcutDecl is a description of a series of joint points and the relevant properties between the joint points.

Advice is used to describe the pointcut in each joint point which should be the implementation of specific related operations, including insert operation and replace operation.

If two or more aspects of the same pointcut crosscut this pointcut, the priorities should be defined priorly between the two different aspects, avoiding conflicts in the aspect weaved. In this mode, $W(b, pc)$ is a binary operator, where $b$ is the name of the basic module, $pc$ is the name of the pointcut; the symbol "$\leq$" means that when the same joint point corresponds to many different aspects, the symbol shows the order between the different aspects and the preparation with the basic module [10]. Its structure is formalized as shown in the following figure 5.

For example, aspect-Z represents a basic module BaseModel and an aspect module AspectModel, and a simple example is given. As shown in Figure 6 and Figure 7.
From the AspectModel aspect, it can be seen that the advice of the Pointcut PC is to insert the condition \(\forall x, y \in A \cdot x \neq y \rightarrow * (x) \neq * (y)\) at each PC element by the special integrated symbol " + " to indicate the final weaving pattern, then the integrated mode IntegratedModel is:

\[
\text{IntegratedModel} = \text{BaseModel} + \text{AspectModel}.
\]

As shown in Figure 8.

4. Requirement analysis based on Aspect-Z

4.1. Formal requirement analysis method

In the requirement analysis of the system, the software system is composed of object layer and layer. Without modifying the object model, the function of the object is increased, and the original function of the object is not affected. The requirements analysis process based on Aspect-Z is as follows.

1) First of all, the requirements of the system are obtained, then the business model is constructed;

2) According to the business model, the system use case model is established;

3) According to the system use cases, the functional requirements and non-functional requirements of the system are extracted. Specific process as shown in figure 9.

4) Aspect-Z language is used to describe the object layer and aspect layer of the system. If we need to weave many aspects at the same connection point, we can solve the problem by defining the priority;

5) Weave the aspects into the object and integrate the complete system;

6) Finally, we prove the correctness of the specification and the description of the Aspect-Z.
Fig. 9 Framework for the realization of non-functional requirements of the system

4.2. Conclusion
This paper mainly studies and discusses the requirement analysis of software system by using AOP technology. It crosscuts the non-functional requirements of the system function and builds up the aspects, which improves the modularity and maintainability of the non-functional requirements in the requirement analysis. And the use of Aspect-Z for the aspect-oriented formal language combines with AOP technology, compared to non-formal lengthy requirements document, which can be clear, precise, abstract, concise specification to describe the software requirements, greatly improving Software security and reliability. This paper is to make full use of Aspect-Z language to aspects modular, by defining the priority to resolve the possible conflicts among the aspects. For the requirements analysis of the system, further research is needed. Such as, Combined with specific examples to analyze the reliability of the non-functional requirements of the system, the fault-tolerance of the system and aspect-oriented multidimensional cross-cutting formalization based on Aspect-Z.

Acknowledgments
The paper is supported by funds from Department of Science and Technology at Guangdong Province and Guangzhou City (2015A030313490).

References
[1] TIAN L, YANG Y. Some Thoughts On Improving The Quality Of Software Requirement Analysis [J]. Electronic Technology And Software Engineering, 2015(5):72-72.
[2] Gorgan V. Requirement Analysis for a Higher Education Decision Support System. Evidence from a Romanian University [J]. Procedia - Social and Behavioral Sciences, 2015, 197:450-455.
[3] Keshav Bharadwaj A, Gopalakrishnan Nair T R. Mapping General System Characteristics toNon-Functional Requirements[C]//Proceedings of the 2009 IEEE International Advance Computing Conference, Patiala India, 2009: 1634 — 1638
[4] YUAN SH J, XU SH P, DU S P, et al. A Method Based Software Requirements Specification Description Quality Evaluation[J]. Electronic Design Engineering,2013, 21(7):11-13.
[5] JIN Y, LIU H B,ZHANG P. An Approach to Analysingand Verifying Aspect-Oriented Requirement Model.Chinese Journal of Computers, 2013, 36(1):63-73.
[6] Lan H Y. Aspect-Oriented Programming[J]. ACM SIGSOFT Software Engineering Notes, 2001, 26(5):468.

[7] WANG SH H, XING J CH, LI J L, et al. Application Research of AOP Technology In Information Management System [J]. Computer Application And Software, 2012, 29(6): 189—195.

[8] Rashid A, Sawyer P, Moreira A, et al. Early Aspects: A Model for Aspect-Oriented Requirements Engineering[C]// 2013 21st IEEE International Requirements Engineering Conference (RE). IEEE Computer Society, 2010:199.

[9] YAO Q Z, WANG J. Software Formalization Requirements Analysis And Verification Based On UML[J]. Computer Engineering, 2010, 36(13):30-33.

[10] LIU J Y, ZHONG Y, ZHANG L. The Aspect-Oriented Modelling of Software Non-Functional Requirements[J]. Computer Application and Software, 2010, 27(12):40-41.

[11] Zhang L. Aspect-Oriented Formal Techniques of Cyber Physical Systems[J]. Journal of Software, 2012, 7(4).

[12] Silva C V, Saens R, Del Río C, et al. Aspect-Oriented Modeling: Applying Aspect-Oriented UML Use Cases And Extending Aspect-Z[J]. Computing & Informatics, 2013, 32(3):10.