Analysis of Software Quality Attributes Through Aspect-Oriented Programming

Garima Soni, Pradeep Tomar, Amrita Upadhyay

Abstract – Design and development of software is difficult task due to the changing requirements of usres. To develop good quality software, our main focus is on analysis of quality attributes according to the users' requirement, but it increases the complexity. So it's a complex task to establish a relationship between users' requirements and quality attributes due to the frequent change in requirements. However, we do not have systematic approaches to design software considering quality attributes and requirements. Throughout the history of software development, new development approches were developed to deal with the chaninging requiremens and ever-growing complexity of software. The object-oriented paradigm approach allows programmers to decompose high level requirements into a set of functional modules. The problem is that Object-Oriented Programming (OOP) does not offer a systematic and elegant way to recompose those modules to maintain the the quality by ensuring the modular continuity. This problem of OOP gave birth to a new paradigm: the Aspect-Oriented Programming (AOP). AOP approach is used to modularize the crosscutting concern and provide better analysis of quality attributes like performance, reliability, availability, reusability, testability modularization and maintainability as compare to OOP in the form of aspects. This paper presents modified analysis process in which designer and developer can separate the aspects related to quality attributes by using AOP and get feedbacks to improve the quality attributes by providing a modularization way to separate crosscutting concerns from non-crosscutting ones. This helps in developing software product more efficiently.

Index Terms – Object, aspect, analysis, quality attributes, aspect-oriented programming, class, concern, analysis process.

1 INTRODUCTION

Software development requires various quality attributes like performance, reliability, availability, maintainability as well as required functions that have to be taken into considerations. Considering quality attributes along with functions makes the software design problem more complex. According to researchers and practitioners many quality models and processes are introduced to maintain the quality of the software, for example *McCall's Model* helps in measuring the software quality, in which many other quality attributes associated with each attribute. For each quality attribute there exist some software properties, which we have to examine while designing the quality attribute during the design phase. Researchers and practitioners call these properties as an aspect. *Natsuko Noda et al.* [1] have examined a design technique in which they consider the problem of each aspect individually and then integrate with the final design output. So, it is better to identify aspects which need to be modularized, in designing phase rather than implementation or programming phase, if developers are developing software from the scratch. But many issues and problems occur while incorporating these quality attributes in designing. AOP solve these issues and problems by allowing the separation of concerns as appropriate for a host language and provides a mechanism for the description of concerns that crosscut other quality attributes of components. AOP isn't meant to replace OOP or other object-based methodologies. Instead, it supports the separation of components, typically using classes, and provides a way to separate aspects from the components. AOP is designed to support the separation of concerns and to allow both a *Logger* and a *Product* class; it also handles the crosscutting that occurs when logging is required in the components supporting another concern.[2] AOP can be implemented in design phase to get a better approach of a system to be developed. It is always good to analyze the quality attributes as early as possible so that we can implement it in a separate way which provides better performance, reliability, availability, reusability, testability modularization and maintainability.

• *Garima Soni, School of Information and Communication Technology Gautam Buddha University, Greater Noida-201 310, Uttar Pradesh, INDIA garima.soni2804@gmail.com*

• *Pradeep Tomar, School of Information and Communication Technology Gautam Buddha University, Greater Noida-201 310, Uttar Pradesh, INDIA parry.tomar@gmail.com*

• *Amrita Upadhyay, School of Information and Communication Technology Gautam Buddha University, Greater Noida-201 310, Uttar Pradesh, INDIA engg.amrita1987@gmail.com*

2 PROBLEM IN DESIGNING QUALITY ATTRIBUTES

OOP has been presented as a technology that can fundamentally aid software engineering, because the underlying object model provides a better fit with real domain problems. But researches have found many programming problems where OOP techniques are not sufficient to clearly capture all the important design decisions regarding quality attributes. Instead, it seems that there are some programming problems that fit neither the OOP approach nor the procedural approach it replaces. [3] In object-oriented technology, the object is an instantiation of a class. The class is an abstract data-type used to model the objects in a system. A *class* is built based on a requirement extracted through an analysis phase. The class might be built on the fly during coding of a solution, with the requirement written in the comments of the class. These requirements and classes can be linked by a concern. A *concern* is some functionality or requirement necessary in a system, which has been implemented in a code structure. This definition allows a concern to not be limited to object-oriented systems - a structured system can also have concerns. In a typical system, a large number of concerns need to be addressed in order for the system to accomplish its goals. A system designer is faced with building a system that uses the concerns but doesn't violate the rules of the methodologies being used. When all the concerns have been implemented with system code as well as related functional tests, the system is complete to maintain and improve the quality attributes. [2] To analyse the implementation of quality attributes according to the users' requirements we face few problems. Quality attributes can be divided into two categories: one is concerned with runtime system attributes such as *performance, security, functionality, availability* and another is not concerned with run time behaviour such as *modifiability, reusability, testability* etc. First problem is difficulties in the implementation of these quality attributes occur because of lack of systematic design methods. There should be no systematic way for designing because change of requirements may changes the architecture design that may involve a great amount of time and cost which are most important factors associated with any software product development. Another problem is related to the solution catalogue, there should be a catalogue of solutions for each problem so that it could be solved with any of the provided solutions. For example, a catalogue of design patterns in which some focus on functionality, performance while some are concerned with reusability, integrability. So to improve the quality attributes problem this paper tries to present the analysis of these problem through the modified analysis process during the design phases.

3 Analysis of Problem

Design and development of software is difficult task due to the changing requirements of usres. To develop good quality software, our main focus is on analysis of quality attributes according to the users' requirement, but is increase the complexity. So it's a complex task to establish a relationship between users' requirements and quality attributes due to the frequent change in requirements. However, we do not have systematic approaches to design software considering quality attributes and requirements. Throughout the history of software development, new development approches were developed to deal with the chaninging requiremens and ever-growing complexity of software. [4] The object-oriented paradigm approach allows programmers to decompose high level requirements into a set of functional modules. The problem is that Object-Oriented Programming (OOP) does not offer a systematic and elegant way to recompose those modules to maintain the the quality by ensuring the modular continuity. This problem of OOP gave birth to a new paradigm: the Aspect-Oriented Programming (AOP). AOP approach is used to modularize the crosscutting concern and provide better analysis of quality attributes like performance, reliability, availability, reusability, testability modularization and maintainability as compare to OOP in the form of aspects. *Natsuko Noda et al.* [1] have observed some nature of attributes which might solve the above mentioned problem. There are many factors which affect the quality attribute and it's not possible to identify each of them and find out the relationship between them and quality attributes, but researchers and practitioners can find out primary factors which affect software quality and determine the general relationship between those factors and quality attributes. One design decision may affect multiple attributes simultaneously. So it's important for designers to focus on different attributes as well as relationship between them. They should also focus on impact of one on another and how one design affects multiple attributes. Designers should also keep in mind that the design decision is made under some assumption and underlying technology.

4 Analysis of Software Quality Through Analysis Process

If we perform the quality analysis during design phase of the software development then it can save much more time and cost rather than performing it after implementation phase. Almost all the already existing techniques inter-mingle the functionality concern and quality concern in the design phase and also make maintenance more complicated. *Daesung Park et al.* [5], have proposed a simulation based design phase analysis method based on AOP. In their proposed method they have kept quality and functionality aspect separate from each other in the design model and the implementation code for simulation is automatically obtained by injecting quality requirements into the skeleton code generated from the design level functionality model. Authors have used an approach to model a system in the aspect-oriented way which implements an executable program for simulation. To develop the simulation model, quality analysis model is required besides the model which exhibits functionality. All the functionality concerns are gathered in design model and all the concerns related to quality will form quality model. In design phase, whole system is divided into two parts, first part is a bunch of core concerns which form a functionality model and rest all the aspects for quality are used in quality analysis model.

But we modify the analysis process of *DaesungPark et al.* with the following steps:

- Requirements Analysis
- Analysis of Quality Attributes
- Modelling through Functionality Model,
- Modelling Quality Attributes through Quality Analysis Model,
- Analysis of Quality Attributes through code concerning core funcationality,
- Simulation with Java Executables
- Feedback.

According to *Fig. 1*, In first two steps we gather the requirements and on the basis of the requirements we decide, what are the main quality attributes which we are going to analyse during the design phase. In next two steps we decompose the system in to separate aspects and third step is the aspectual composition for analysis of quality attributes through code concerning core funcationality. According to *Fig. 1*, in the next step we construct a funcational and design model based on functionality of a system. Here we are only considering quality attributes of the system, which can be defined as the ability of a system to perform the function it is intended to. It is easier to generate skeleton program code from class diagrams rather than directly from given requirements. Many UML tools can generate code for class, attribute, signature of operation, and relationship from class diagrams. Sequence diagrams can model object interactions arranged in time sequence and to distribute use case behaviour to classes. As a consequence we get file with .java extension. In the next step we use Quality Model Analysis to analyse the quality attribute. The proposed approach is scenario-based analysis. A scenario is a sequence of component interactions triggered by a specific input stimulus. Here we will use operational profiles for example Musa's Operational profile for description of scenarios with specified input variables. It takes operational profiles and quality metrics as inputs. Profiles or metrics differ from quality to quality.

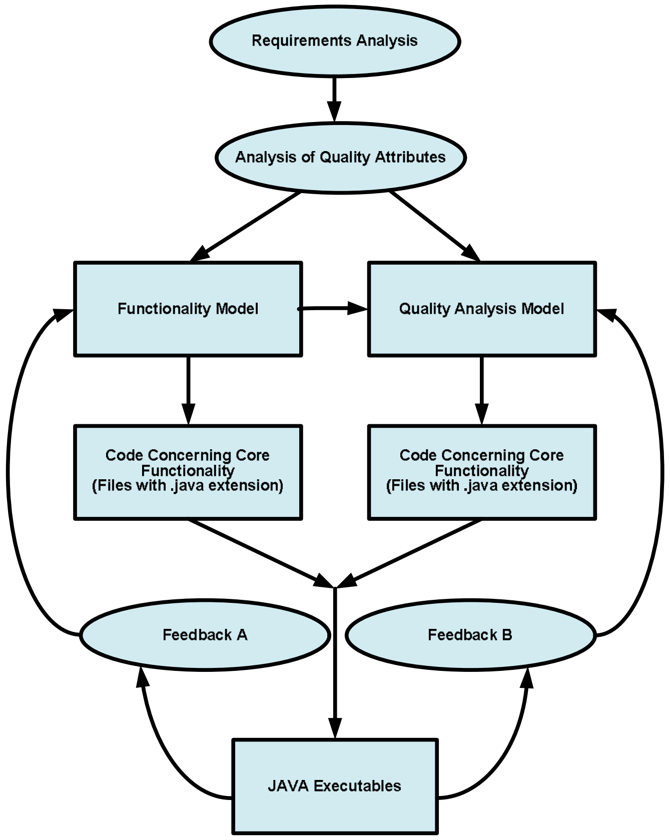
Requirements Analysis
Analysis of Quality Attributes
Functionality Model
Quality Analysis Model
Code Concerning Core Functionality (Files with .java extension)
Code Concerning Core Functionality (Files with .java extension)
Feedback A
Feedback B
JAVA Executables

Fig. 1 Modified Analysis Process

After this step we get file with .aj extension. In next step we make a simulation program With AOP, quality analysis aspects can be developed independently. Then AspectJ compiles code files from Step (3 and 5) and code files from Step (4 and 5) together. This compilation process is called weaving in the aspect-oriented terms. This gives files with the extension ".class". To perform next step we execute the result of Step (6 and 7) and observe the result. Simulation model is based on a number of assumptions about the quality attributes of the real system's behaviour. The simulation model can be implemented to produce a simulation program. We can measure values of the quality attribures of interest by executing it. Then we get feedback by executing the simulation model. In analysis process, feedback can result in redesign of software or reorganization of resource demands when analysis result does not meet the requirements. So through the feedback process during the design phase in the modified analysis process we can easily check the quality attributes are according to the requirement analysis or not. To develop good quality software, our main focus is on analysis of quality attributes according to the users' requirement, through the modified analysis process. So this process helps in develpong software product more efficiently by analsing the quality attributes.

5 Conclusion

It's always a difficult task to design software to meet its goal based on quality attributes but there are many factors related to quality attributes and it's a complex task to establish a relationship between those factors and quality attributes. Modified analysis process are dicusses above for software quality analysis which shows how factors related to quality attribute can be separated from core concern of a system using AOP in design phase. AOP can be implemented in design phase to get a better approach of a system to be developed. It is always good to analyze the quality parameters as early as possible so that we can implement it in a separate way which provides better performance, reliability, availability, reusability, testability modularization and maintainability. Development becomes much easier than the case in which various concerns for simulation are interwoven with functionality. So with above discussion we can say that factors related to quality attribute can be distinguished from core functionality to provide better modularization and maintainability in design phase using AOP. With modified analysis process, design method discussed above, helps in develpong software product more efficiently.

References

[1] Natsuko Noda & TomojiKishiOn, "Aspect-Oriented Design- An Approach to Designing Quality Attributes", proceddings of Asia Pacific Software Engineering Conference. (APSEC '99), 1999.
[2] httpmedia.johnwiley.com.au/product_data/excerpt/44/04714310/0471431044, 0471431044_01.qxd , 2003.
[3] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, John Irwin, Aspect-Oriented Programming, proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finland. Springer-Verlag LNCS 1241. 1997.
[4] Jean-Yves Guyomarc'h and Yann-Gael Gueheneuc, "On the Impact of Aspect-Oriented Programming on Object-Oriented Metrics, http://www-etud.iro.umontreal.ca/~ptidej/yanngael/Work/Publications/Documents/ECOOP05QAOOSEa.doc.pdf.
[5] Daesung Park, Sungwon Kang and Jihyun Lee. "Design Phase Analysis of Software Qualities Using Aspect-Oriented Programming", proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Com puting (SNPD'06) , 2006.