International Journal of Engineering Research and Technology.
ISSN 0974-3154 Volume 8, Number 1 (2015), pp. 1-6

http://www.irphouse.com

Software Quality Metrics for Aspect-Oriented Programming

Lovely Balani and Alok Singh

Asst.Professors, MCA
Thakur Institute of Management Studies, Career Development & Research (TIMSCDR) Mumbai, India-400101
lovely.lakhmani@gmail.com, Singh.alok@thakureducation.org

Abstract:

Aspect Orientation is one of the upcoming methodologies to enhance S/W design and reuse. Aspects are properties of a software system which tends to cut across its main functionality. Some of the examples of aspects are synchronization, resource sharing and distribution. Aspects are intertwined within core components of the S/W. this causes "Code-tangling" problem. Aspect Oriented approach addresses "Code-tangling "problem by modularizing these cross cutting concerns and encapsulate them in modules.
Major objectives of Aspect Oriented Approach are to enhance Software Design and reuse. To ensure that Aspect Oriented approach really accomplish its objectives sound S/W quality metrics are needed.
As with other approaches of SE aspect Oriented System can also be poorly designed. Sometimes it may be possible that a well-designed OO system might be made worse with introduction of aspects. Aspects may increase the system complexity and reduce its understandability.
There is a great need to develop Quality models and metrics to assess quality of software developed using Aspect Oriented methodologies. Some researchers attempted to build quality models and metrics for aspect-oriented Software, however all these works are still at an early stage. In this paper we are going to discuss introduction of SQA metrics for the evaluation of Aspect Oriented Programs.

Keywords: AOP, AORE, Aspect, AOSQA, AOM, code –tangling, quality metrics.

Introduction:

Aspect Orientation is one of the upcoming methodologies to enhance S/W design and

reuse. Aspects are properties of a software system which tends to cut across its main functionality. Some of the examples of aspects are synchronization, resource sharing and distribution. Aspects are intertwined within core components of the S/W. this causes "Code-tangling" problem. Aspect Oriented approach addresses "Code-tangling "problem by modularizing these cross cutting concerns and encapsulate them in modules.

Major objectives of Aspect Oriented Approach are to enhance Software Design and reuse. To ensure that Aspect Oriented approach really accomplish its objectives sound S/W quality metrics are needed.

As with other approaches of SE aspect Oriented System can also be poorly designed. Sometimes it may be possible that a well-designed OO system might be made worse with introduction of aspects. Aspects may increase the system complexity and reduce its understandability.

Increasing complexity of software worked as motivation to develop new software development methodologies to handle these complexities efficiently and effectively. One of the most popular software development paradigms is object oriented programming. Object-oriented paradigm specifies concepts, or concerns, as objects and such an approach allow programmers to decompose high level requirements into a set of functional modules. To overcome these limitations of object-oriented programming (OOP) a new paradigm known as the Aspect-oriented programming (AOP) has been developed. Aspect-oriented programming is an approach designed to overcome limitations of object-oriented programming. A new paradigm requires software engineers to define new metrics and quality models to measure the quality of programs in this paradigm. Any new development paradigm and associated design practices need to be evaluated in term of quality. Currently it is quite difficult to determine good design and implementation decisions for aspect-oriented programs.

There is a great need to develop Quality models and metrics to assess quality of software developed using Aspect Oriented methodologies. Some researchers attempted to build quality models and metrics for aspect-oriented Software, however all these works are still at an early stage.

Here in this paper we propose a rigorous approach to develop Aspect-oriented programming metrics on the basis of object-oriented metrics. Since AOP has close similarity with OOP paradigm and object oriented languages and henceforth on the metrics dedicated to these languages our proposed model and quality metrics will be developed on the foundation of OOPs Quality metrics and models.

Aspect Oriented Framework:

Aspect Oriented Software Engineering (AOSE) framework contains all the phases of Software Development Life Cycle (SDLC). AOSE has all the phases of SDLC Namely:

1. Aspect Oriented Requirement Engineering (AORE).
2. Aspect Oriented Modeling (AOM).
3. Aspect Oriented Programming (AOP).
4. Aspect Oriented Software Quality Assurance (AOSQA).

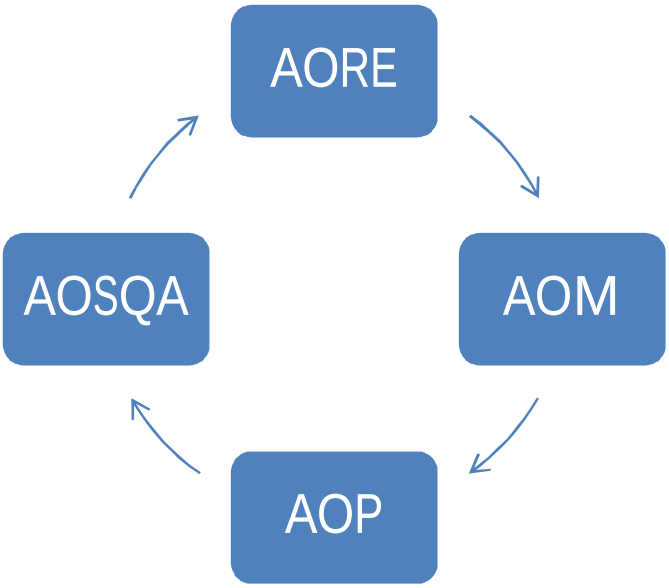

Figure 1: Aspect Oriented Software Engineering Frame Work

1. Aspect Oriented Requirement Engineering (AORE):

Aspect-oriented requirement Engineering (AORE) is the first phase in Aspect Oriented Software Development Model (AOSDM). AORE consist of

- Identification of cross cutting concern.
- Specification of cross cutting concern.
- Representation of cross cutting concern.

at the requirement level. Cross cutting concern include security, mobility, availability and real-time constraints. Crosscutting concerns are requirements, use cases or features that have a broad effect on other requirements or architecture components.
AORE is a collection of techniques that recognize the importance of addressing both functional and non-functional crosscutting concerns in addition to non-crosscutting ones. Therefore, these approaches focus on systematically and modularly treating, reasoning about, composing and subsequently tracing crosscutting functional and non-functional concerns via suitable abstraction, representation and composition mechanisms tailored to the requirements engineering domain.

Aspect Oriented Requirements Analysis deals with:

a) Aspect-oriented requirements process.
b) Aspect-oriented requirements notations.
c) Aspect-oriented requirements tool support,
d) Adoption and integration of aspect-oriented requirements engineering, and
e) Assessment of aspect-oriented requirements.

2. Aspect-Oriented Modeling and Design(AOMD):

AOMD involves identifying, analyzing, managing, and representing crosscutting concerns. AOM targets a simplified, abstract description of an aspect-oriented design. An aspect-oriented modeling method requires three types of constructs for modeling: base elements, crosscutting elements, and crosscutting relationships. Objective of Aspect-oriented design is to characterize and specify the behavior and structure of the software system. AOAD contributes in software design through modularizing concerns that are necessarily scattered and tangled in more traditional approaches. Generally, such an approach includes both a process and a language. The process takes requirements as input and produces a design model that represents separate concerns and their relationships. The programming language provides constructs to support concern modularization and the specification of concern composition.

AOAD deals in following areas:

a) Aspect-oriented design process itself.
b) Aspect-oriented design notations.
c) Aspect-oriented design tool support.
d) Adoption and integration of aspect-oriented design.
e) Assessment/evaluation of aspect-oriented design.

3. Aspect-Oriented Programming(AOP)

A software system is as a set of structured modules, representing a concern (functionality or a requirement). Object-oriented paradigm (classes, objects, methods, and attributes) offers abstractions which are not sufficient to represent all the concerns of a software system.

Concerns such as logging, tracing, or persistence tend to be scattered and tangled all across the objects of the system. These concerns are known as crosscutting concerns because they cut across other functionality of the program. Aspect oriented paradigm address the problem of crosscutting concerns through separation of concerns and ensure a good modularization. In AOP aspect is used to encapsulate a crosscutting concern and AOP also facilitates composition of aspect and components such as class, methods and attributes. AOP provides mechanism which allows the aspect to crosscut object-oriented abstractions. AOP paradigm must be implemented for all object oriented languages such as C++, JAVA etc. Although AOP is a relatively new Software Development Paradigm, some research addressing metrics and quality has already been conducted.

Metrics for AOP:

After extensive survey of existing literature on Aspect Oriented Software Quality Metrics we found some of the key contributions made by contemporary software Engineering researchers. Following are the accounts of some of these key contributors work, these noted work will form the basis of our approach for Aspect Oriented Software Quality Assurance, metrics and measurements:

Metric suite based on a ***Dependence model*** for aspect- oriented software:

This model consists of a group of dependence graphs representing various dependence relationships at different levels of aspect-oriented programs.

Coupling measure for aspect-oriented programs:

This metric is considered as the degree of interdependence between aspects and classes.

Aspect cohesion based on dependence analysis:

Cohesion for an aspect is tells how tightly the attributes and modules (methods and advices) of aspects cohere.

Since studying AOP and its implementations is necessary engineers must define metrics and models to assess the quality of aspect-oriented system. On the other hand, the impact of AOP over object-oriented languages, such as Java must also be assessed and understood.

Methodology:

We intend on extending the work of all the researchers discussed above.

We want to fully evaluate effect of all the abstractions introduced by AOP, on object-oriented languages. We believe that a mechanism such as inter-type declaration has an important incidence on object- oriented metrics. For example, adding members or declaring that types extend on the Impact of Aspect-Oriented Programming. New types through inter-type declaration affect metrics like coupling, cohesion, or depth of inheritance.

During our review of the literature, we also noticed the lack of a common agreed-upon example of AOP program and aspect. We believe that using a unique example among all the papers published on the subject could help both the authors and the readers. So we plan to develop and to use an example complete enough to satisfy as many needs as possible among the researchers working on AOP and quality.

Conclusion and Future Work

Assessing the quality of software is an integral process of software engineering. The problem of separation of concerns fueled the growth of the aspect-oriented paradigm. This new paradigm raises questions about quality, due to its close relations with object-oriented programming.

Our work of developing AOP metrics on object-oriented metrics and the implementation of a measurement framework shall enable us to build quality models to assess the quality of aspect-oriented programs.

The proposed work shall be validated through empirical studies. In fact, our framework shall enable us to appraise the quality of an AOP program.

References :

1. Meyer, B.: Object-Oriented Software Construction (Book/CD-ROM) (2nd Edition). Prentice Hall PTR (2000)
2. Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J. In: Aspect-Oriented Programming. Volume 1241. Springer-Verlag, Berlin, Heidelberg, and New York (1997) 220–242
3. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An overview of AspectJ. Lecture Notes in Computer Science 2072 (2001) 327–355
4. Gradecki, J.D., Lesiecki, N.: Mastering AspectJ: Aspect-Oriented Programming in Java. Wiley (2003)
5. Zhao, J.: Towards a metrics suite for aspect-oriented software. Technical Report SE-136-25, Information Processing Society of Japan (IPSJ) (2002)
6. Zhao, J.: Measuring coupling in aspect-oriented systems. Technical report, Information Processing Society of Japan (IPSJ) (2004)
7. Zhao, J., Xu, B.: Measuring aspect cohesion. In: Proceeding International Conference on Fundamental Approaches to Software Engineering. Volume 2984., Springer Verlag (2004) 54–68
8. G´linas, J.F., Badri, L., Badri, M.: Aspect cohesion measurement based on depenedence analysis. In: Proceedings of the Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE). (2004) 121–125
9. Dufour, B., Goard, C., Hendren, L., Moor, O.d., Sittampalam, G., Verbrugge, C.: Measuring the dynamic behaviour of aspectj programs. In: OOPSLA ’04: Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications. Volume 39., New York, NY, USA, ACM Press (2004) 150–169
10. Sant’Anna, C., Garcia, A., Chavez, C., Lucena, C., Staa, A.v.: On the reuse and maintenance of aspect-oriented software: An assessment framework. In: Proceedings XVII Brazilian Symposium on Software Engineering. (2003)
11. Chidamber, S., Kemerer, C.: A metrics suite for object oriented design. Software Engineering, IEEE Transactions on 20 (1994) 476–493
12. Lopes, C.I.V.: D: A Language Framework For Distributed Programming. PhD thesis, College of Computer Science of Northeastern University (1997)
13. Gu´h´neuc, Y.G.: Un cadre pour la tra abilit´ des motifs de conception. PhD thesis, Ecole des Mines de Nantes et Universit´ de Nantes (2003)