An Establishment of Aspect-Oriented Software Development

Ashwin Kumar[1], Debabrata Samanta[2], Mousumi Paul[3]

[1]Student, MCA Dept, Acharya Institute of Technology, Bangalore
[2]Asst. Prof., MCA Dept, Acharya Institute of Technology, Bangalore
[3]PhD Scholor, National Institute of Technology, Durgapur, West Bengal
Email:research.k04@gmail.com

Abstract

Aspect oriented software measurement has been given a foundational, hypothetical origin by the research of Chidamber and Kemerer, partly based on Wand and Weber's construal of the ontology of Bunge. Aspect oriented software enlargement lacks like a sound theoretical basis, with the result that there is some puzzlement over what constitutes an aspect and how best to measure. This paper express Dooyeweerd's theory of aspects which provides a theoretical basis upon which may be created an ontological approach to the understanding of aspect oriented programs. The paper presents an overview of Dooyeweed's theory and uses it to put forward a plan for developing for aspect oriented software.

Keywords: Aspect oriented software development, software measurement.

1. Introduction

Aspect oriented software development is a new advance to software development that addresses limitations inherent in other approaches such as object oriented software development. In traditional software development common concerns are identified and used to modularize a program.

Traditional engineering disciplines manage the complexity of systems by identifying and separating the system's concerns and treating each concern in isolation; such an approach known as separation of concerns (SOC), leads to systems that are easier to implement, verify, evolve, and understand. The craft of software development quickly adopted this approach; programming languages were forerunners in the advancement of software technology with languages progressively providing abstractions such as functions, procedures, abstract data types, and objects to help achieve higher levels of SOC. Aspect-oriented programming (AOP) technology is a possible next step in this progressive trend.

2. Aspect-Oriented Programming

In computing, aspect-oriented programming (AOP) is a programming paradigm that endeavors to increase modularity by allowing the separation of cross-cutting concerns. AOP forms a basis for aspect-oriented software development.

AOP includes programming techniques and tools that support the modularization of distress at the level of the source code, while "aspect-oriented software development" refers to a whole engineering discipline.

Despite ongoing and productive dialogue amongst the AOP community, a common consensus on what constitutes an AOP approach is yet to be reached (though significant efforts have been made).

Quantification means that programs can include quantified statements (i.e. statements that apply to more than one place) of the form "In programs P, whenever condition C arises, perform action A"; obliviousness means that authors of a program P need not be aware of quantified statements that reference them.

Aspect-oriented programming entails breaking down program logic into distinct parts (so-called concerns, cohesive areas of functionality). Nearly all programming paradigms support some level of grouping and encapsulation of concerns into separate, independent entities by providing abstractions (e.g., functions, procedures, modules, classes, methods) that can be used for implementing, abstracting and composing these concerns. But some concerns defy these forms of implementation and are called crosscutting concerns because they "cut across" multiple abstractions in a program.

3. Case Study

The broad domain of the case study is crisis management systems, i.e. software that facilitate coordination of activities and information flow between all stakeholders and parties that need to work together to handle a crisis.

- Textual requirements describing the general functional and non-functional requirements of crisis management systems
- Feature diagrams describing many variations of crisis management systems
- Textual requirements describing the functionality of a car crash crisis management system
- A use case model describing some of the use cases of the car crash crisis management system

• A domain model of the car crash crisis management system
• An informal description of a possible architecture for the car crash crisis management system
• Some detailed design models for the car crash crisis management system backend

4. Module

The module uses these formal approaches to
■ define and compare declaring and weaving aspects,
■ specify the properties an aspect adds and determine whether these are true when the aspect is woven to a system, and
■ show that composing an aspect doesn't disturb the system's desirable properties.
The module also surveys existing work on defining and analyzing interactions among multiple aspects.

5. Objectives

The objective is to introduce on aspect-oriented software development, which is based on the separation of concerns.
■ understand why the separation of concerns is a good guiding principle for software development;
■ have been introduced to the fundamental ideas underlying aspects and aspect-oriented software development;
■ understand how an aspect-oriented approach may be used for requirements engineering, software design, and programming;
■ be aware of the difficulties of testing aspect-oriented systems

6. Conclusion

Aspect-oriented software development (AOSD) is a budding knowledge that supports multi-dimensional separation of concerns throughout the software development cycle. The aspect-interaction trouble has been identified as an under-researched area in AOSD; we wish to investigate practical and formal approaches for the detection of aspect interactions at the design phase.

7. References

[1] Chidamber, S.R.; Kemerer, C.F., "A metrics suite for object-oriented design," Software Engineering, IEEE Transaction son, vol.20, no.6pp.476-493, Jun 1994
[2] Wand, Y. and Weber, R., An ontological model of an information system, IEEE Transactions on Software Engineering, vol. 16, issue 11, pp. 1282-1292, 1990.
[3] Bunge, M.Treatise on Basic Philosophy: Ontology 1: The Furniture of the World, Boston: Riedel, 1977.
[4] K. Lieberherr, D. Oleans, and J. Ovlinger. Aspect-Oriented programming with adaptive methods.Communications of the ACM, 44(10):39--41, Oct. 2001.
[5] H. Ossher, and P. Tarr. Using multidimensional separation of concerns to (re)shape evolving software.Communications of the ACM, 44(10):43--50, Oct. 2001.
[6] L. Bergmans, and M. Aksit. Composing crosscutting concerns using composition filters. Communications of the ACM, 44(10):51--57, Oct. 2001.
[7] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finland,June 1997. Springer-Verlag.
[8] I. Aracic et al., "An Overview of CaesarJ," Trans. Aspect-Oriented Software Development, vol. 1, 2006, pp.135–173