

Aspect Oriented Software Development: Towards A Philosophical Basis

Technical Report TR-06-01, Department of Computer Science, King's College London

February 17th 2006

Aspect Oriented Software Development: Towards A Philosophical Basis

Sue Black

Centre for Systems and Software Engineering
London South Bank University

103 Borough Road,

London SE1 0AA, UK

+44(0)20 7815 7471

blackse@lsbu.ac.uk

Mark Harman

Software Engineering Group
Department of Computer Science
King's College London, Strand,

London WC2R 2LS, UK

+44(0)20 7848 2895

Mark.Harman@kcl.ac.uk

ABSTRACT

Object oriented software measurement has been given a foundational, theoretical basis by the research of Chidamber and Kemerer, partly based on Wand and Weber's interpretation of the ontology of Bunge. Aspect oriented software development lacks such a sound theoretical basis, with the result that there is some confusion over what constitutes an aspect and how best to measure and assess the properties of an aspect oriented program. This paper describes Dooyeweerd's theory of aspects which provides a theoretical basis upon which may be constructed an ontological approach to the understanding of aspect oriented programs. The paper presents an overview of Dooyeweerd's theory and uses it to put forward a plan for developing an ontology for aspect oriented software development and measurement.

Categories and Subject Descriptors

D.2.0 [Software Engineering]: General

General Terms

Design, Languages.

Keywords

Aspect oriented software development, software measurement, ontology.

1. INTRODUCTION

Aspect oriented software development is a new approach 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 modularise a program. Aspect oriented software development adds crosscutting concerns to address those features that cut across modules, commonly cited examples are logging and security concerns (see Figure 1). Measurement of object

oriented software has been given a fundamental philosophical basis through the work of Chidamber and Kemerer [1] partly based on Wand and Weber's interpretation [2] of the ontology of Bunge proposed in his 'Treatise on basic philosophy' [3].

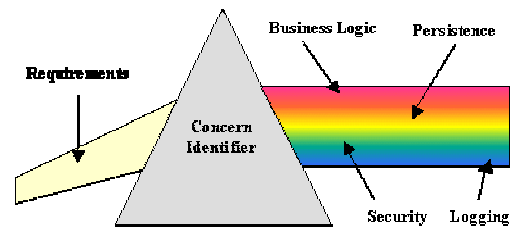


Figure 1: Concern decomposition: The prism analogy [4]

There is not, thus far, a corresponding theoretical basis for aspect oriented measurement. It is this need that this paper attempts to address by highlighting the philosophy of Herman Dooyeweerd and using it to put forward a plan for developing an ontology for aspect oriented software development. Research has been carried out recently which uses Dooyeweerd's philosophy to understand information systems [5]. Conclusions of the research were that this philosophy can help build new frameworks for understanding four areas of concern within IT, namely: the use of IT artifacts, the development of information systems (including both artifact and human beings), the development of technologies, and perspectives on information systems.

This paper suggests that a similar approach should be taken with aspect oriented software development to deepen our understanding of the paradigm. The contribution of this paper is to highlight an area of philosophical thought, until now unconnected with aspect oriented software development, and show how it may, with some research effort, provide a foundational underpinning via an ontology for the aspect oriented software development paradigm. Section one has provided a brief introduction to the idea of providing a philosophical basis for aspect oriented software development. Section two describes Dooyeweerd's theory of aspects and how they can be used. Ontologies and their contribution to our understanding of areas of common interest are described in section three and a short discussion of the applicability of Dooyeweerd's theory of aspects to producing an ontology for aspect oriented software development is discussed in section four. Conclusions and further work are detailed in section five.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '04, Month 1-2, 2004, City, State, Country.

Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

2. DOOYEWEERD'S THEORY OF ASPECTS

Herman Dooyeweerd (1894-1977) was a Christian philosopher based in the Netherlands whose work has until recently not been well known outside the Netherlands. Amongst other work, he is well known for his 'Theory of aspects' which is described briefly here. A much fuller description of Dooyeweerd's philosophy is given on the 'Dooyeweerd pages' [6] created by Andrew Basden at Salford University, UK.

To Dooyeweerd the behaviour of the parts is determined by the behaviour of the whole, whereas we assume that the behaviour of the whole is determined by the behaviour of the parts. Therefore, the whole is not seen as the sum of the parts, but rather the whole has meaning as qualified by different aspects. Aspects enable us and all other things to function, and each has its own set of related laws. We experience coherence through aspect relationships, and diversity, as all wholes comprise several aspects. The aspects are irreducible to each other and properties that are meaningful within one aspect can be seen as emergent properties of another. Life is multi-aspectual in nature, so no aspect can be neglected in real life, each aspect has its own way of knowing. Dooyeweerd's aspectual suite, defined in 1932, is as follows [7]:

QUANTITATIVE - quantity and amount
SPATIAL - space, continuous extension
KINEMATIC - movement
PHYSICAL - energy, mass
BIOTIC - life functions
SENSITIVE - sense, feeling, emotion
ANALYTICAL - distinguishing
FORMATIVE - history, culture, technology
LINGUAL - symbolic, communication
SOCIAL - social interaction
ECONOMIC - frugal use of resources
AESTHETIC - harmony, surprise, fun
JURIDICAL - what is due, retribution, rights and responsibilities
ETHICAL - self-giving love
PISTIC - vision, aspiration, creed, religion, commitment

The fifteen aspects are shown in upper case and the corresponding areas of relevance in lower case. Dooyeweerd said that a knowledge of the aspects helps in the analysis of complex situations, suggesting how to distinguish different meanings and functions. He used the symbol of a prism comparing the numerous aspects of our experience to the colours that are refracted by a prism. Each of the colours exists in coherence with all the others, but none of the colours can be derived from the other. Dooyeweerd's aspects can help in clarifying tangled issues.

3. ONTOLOGIES IN COMPUTING

Dooyeweerd's 'Theory of aspects' will provide a philosophical foundation on which to base an ontology for aspect oriented software development and measurement. An ontology defines a common vocabulary for researchers to be able to communicate

effectively about a topic of common interest. More formally it is defined as [8]:

"...a specification of a conceptualization."

which can also be used to enable reuse of domain knowledge, make domain assumptions explicit, analyse domain knowledge and separate domain knowledge from operational knowledge [9].

In computer science, an ontological model of an information system was proposed in 1990 by Wand and Weber [2]. They were seeking to define a set of core concepts that could be used to define the structure and behaviour of an information system. Their work is based on the ontology of Bunge [3] which forms the basis for the concept of objects. The ontology of Bunge however does not provide a specific ontological definition of object oriented concepts, Wand and Weber employed his generalised concepts to the object oriented domain. Bunge's ontology deals with the meaning and definition of representation of the world which are precisely the goals of the object oriented approach. [2]. Consistent with Bunge's ontology objects are defined independent of implementation considerations and encompass the notions of encapsulation, independence and inheritance [1].

Wand and Weber's research ideas are used and carried forward in the measurement domain by Chidamber and Kemerer [1]. In terms of software measurement they use an ontological model to measure code rather than a graph theory, information content or structural attribute approach. They extend Bunge's ontology which is intuitively similar to their approach and use it to form the basis for their suite of object oriented metrics. An ontology for the aspect oriented paradigm has been produced as part of the AOSD Europe EU network of excellence [10] namely: AOSD Ontology 1.0 – Public Ontology of Aspect-Oriented. It is not our aim to replace this ontology but to complement and possibly enhance it by adding a new perspective.

4. TOWARDS AN ONTOLOGY FOR THE ASPECT ORIENTED PARADIGM

As mentioned previously, measurement of object oriented software has been given a fundamental philosophical basis through the work of Chidamber and Kemerer partly based on Wand and Weber's interpretation of the ontology of Bunge proposed in his 'Treatise on basic philosophy'. Bunge did not provide specific ontological definitions for OO concepts, several researchers have applied his generalised concepts to the OO domain, most notably Wand and Weber. Bunge's ontology has considerable appeal for OO researchers as it deals with the meaning and definition of representations of the world which are precisely the goals of the OO approach. Consistent with this ontology, objects are defined independent of implementation considerations and encompass the notions of encapsulation, independence and inheritance. The world is viewed as composed of things: substantial individuals and concepts. Substantial individuals possess properties and an observer can assign features to an individual but these are attributes and not properties. A substantial individual and its properties constitute an object, therefore an object is not simply a bundle of methods but a representation of the application domain that includes the methods and instance variables that a designer assigns to that object.

Dooyeweerd's theory of aspects fits nicely with the aspect oriented software development paradigm. In everyday life we function in all aspects without being aware of them, they are integrated around us in a harmonious way. We could say that same about an optimal software system. An aspect oriented approach takes the disharmony that is created when aspectual properties of a system are modelled as objects and uses aspects to harmonise the system. There is a parallel between the way that Dooyeweerd's suite of aspects and the aspect oriented paradigm are commonly viewed, they are both commonly explained using the analogy of a prism splitting the area of interest into aspects. In the aspect oriented paradigm this is seen as system requirements being split into: business logic, logging, persistence and security for example, see Figure 1. Dooyeweerd used the symbol of a prism comparing the numerous aspects of our experience to the colours that are refracted by a prism. Each of the colours exists in coherence with all the others, but none of the colours can be derived from the other. This relates to the aspects within an aspect oriented program, the security aspect and the logging aspect exist in coherence with each other, but cannot be derived from one another.

When applying the aspects to any given area probably only a few will be relevant. Three of the aspects have thus far been identified as being relevant to the philosophy of aspect oriented development and the fundamental idea of separation of concerns:

SPATIAL – a software system may be located in geographically disparate locations and the user may be oblivious to this unless the system is viewed from the spatial aspect.

LINGUAL – A system may be written in a particular programming language and may be viewed as language independent if that concern is separated out.

SOCIAL – It has been put forward (anecdotally) that computer programmers have problems relating to the human aspect of system development. Separating out the social aspect may help us to understand the way that programmers look at the world.

A plan of work for developing and using the ideas presented in this paper is as follows:

1. Conduct a comprehensive study of Dooyeweerd's aspectual suite and investigate its relevance to the aspect oriented paradigm
2. Use this information to produce an ontology for aspect oriented software development and measurement. This may be built on top of the 'Public Ontology of Aspect Orientation' produced by the AOSD Network of Excellence [10].
3. Use the ontology developed to reason about aspect oriented software development and measurement.
4. Produce a suite of aspect oriented measures based on the philosophy and related ontology.

The ontology produced by this process will provide a foundational, theoretical basis for the aspect oriented paradigm. The corresponding metrics will provide a basis for the understanding of aspect oriented software. This will aid future research in this area and will help answer questions such as 'What is the constitution of an aspect?' or 'What is the best way to measure and assess the properties of an aspect oriented program?'

4. CONCLUSIONS AND FURTHER WORK

Measurement of the object oriented paradigm was given a fundamental philosophical basis through the work of Chidamber and Kemerer partly based on Wand and Weber's interpretation of the ontology of Bunge. Ontologies provide us with a specification of a conceptualisation and a common vocabulary with which to communicate effectively about a subject area. This paper has introduced and briefly described Dooyeweerd's theory of aspects which has been used within the information systems domain to gain a deeper knowledge and understanding of the fundamental ideas underpinning the area. It provides a description of a theoretical basis upon which an ontological approach to the understanding of aspect oriented programs can be constructed. Further work has been detailed in the plan outlined in section 4. The main focus will be to produce an ontology for aspect oriented software development and measurement based on the ideas discussed in this paper.

5. ACKNOWLEDGMENTS

Sue Black would like to thank London South Bank University for funding her sabbatical and King's College London for hosting it.

6. REFERENCES

- [1] Chidamber, S.R.; Kemerer, C.F., "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, 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 I: The Furniture of the World*, Boston: Riedel, 1977.
- [4] Laddad, 2002, I want my AOP, Separate software concerns with aspect-oriented programming <http://www.javaworld.com/javaworld/jw-01-2002/jw-0118-aspect.html> last accessed 26/01/06.
- [5] Basden, A., Christian Philosophy and Information Systems, Information Systems Institute, University of Salford, Salford, UK, October 2001. <http://www.basden.u-net.com/R/cpis/cpis.html> last accessed 26/01/06.
- [6] Basden, A., The Dooyeweerd Pages, <http://www.isi.salford.ac.uk/dooy/> last accessed 26/01/06.
- [7] Basden A., The critical theory of Herman Dooyeweerd? *Journal of Information Technology*, Volume 17, Number 4, December 2002, pp. 257-269(13).
- [8] Gruber, T. R., A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220, 1993.
- [9] Noy, N. F. and McGuinness, D. L., Ontology development 101: A guide to creating your first ontology, Stanford University, Stanford, CA 94305, USA, http://protege.stanford.edu/publications/ontology_development/ontology101.pdf
- [10] <http://www.aosd-europe.net/documents/index.htm> last accessed 25/01/06.