Object Oriented Software Metrics: A Practical Guide

First International Symposium on Empirical Software Engineering and Measurement

Fine-Grained Software Metrics in Practice

Michael English, Jim Buckley and Tony Cahill
CSIS Department, University of Limerick, Ireland.
{Michael.English, Jim.Buckley, Anthony.Cahill}@ul.ie

Abstract

Modularity is one of the key features of the Object-Oriented (OO) paradigm. Low coupling and high cohesion help to achieve good modularity. Inheritance is one of the core concepts of the OO paradigm which facilitates modularity. Previous research has shown that the use of the friend construct as a coupling mechanism in C++ software is extensive. However, measures of the friend construct are scarse in comparison with measures of inheritance. In addition, these existing measures are coarse-grained, in spite of the widespread use of the friend mechanism.

In this paper, a set of software metrics are proposed that measure the actual use of the friend construct, inheritance and other forms of coupling. These metrics are based on the interactions for which each coupling mechanism is necessary and sufficient. Previous work only considered the declaration of a relationship between classes.

The software metrics introduced are empirically assessed using the LEDA software system. Our results indicate that the friend mechanism is used to a very limited extent to access hidden methods in classes. However, access to hidden attributes is more common.

1. Object-Oriented Software Development

The Object-Oriented, (OO), approach to software construction became popular due to its ability to model real-world entities naturally and to provide suitable units for building large and complex systems, [31]. It is now considered the "dominant software technology", [14].

The object-oriented approach to software development promotes software modularity. "Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules", [3]. Coupling is a measure of the inter-connectedness of modules and cohesion is a measure of intra-module connectedness. Modularity is considered one of the core requirements of software quality as illustrated by its inclusion in many software quality models. For example, the three software quality models, [23, 20, 15], discussed by Kitchenham and Pfleeger, [21], all include aspects of modularity. Due to the importance of modularity, measuring coupling and cohesion in software systems is an important research area.

Inheritance is a structural OO design mechanism and its use results in the creation of coupling between classes. It is one of the distinguishing features of the OO paradigm. As such there has been much discussion and disagreement as to the appropriate uses of inheritance, e.g. [27, 26, 30, 35, 24]. These discussions focus mainly on whether inheritance should be used for code reuse or whether an underlying semantic relationship should exist. Many empirical studies have also been undertaken to examine the use of inheritance in software systems, e.g. [2, 8]. These studies tend to highlight the limited use of inheritance in practice.

The friend construct is a C++ mechanism which grants a class or function access to the internal parts of other classes. Thus, the use of the friend construct also results in the creation of coupling between classes. Since the use of the friend construct does not restrict coupling to occur through a class's interface, it is often considered poor programming practice, [11, 26]. Indeed the use of the friend mechanism has been associated with defects in software, [6, 29]. Others suggest that it is useful under certain conditions, [25, 28]. However, overall the consensus in the literature seems to be that the use of the friend mechanism is poor programming practice, results in poor design and poor quality and thus its use should be minimised.

In contrast to inheritance, little empirical analysis of the use of the friend mechanism in existing software systems has been undertaken and thus, there is limited empirical evidence to support the consensus view expressed above. This is surprising given the extensive use of the friend mechanism, [18, 12, 13]. In fact Counsell *et al.* stated that library-based systems "showed a distinct lack of any form of coupling (including inheritance)" between classes "other than through the C++ friend facility", [13]. This type of analysis would be useful to support or reject the many claims that have been made in the literature about the use of the friend construct in the development of object-oriented software systems, or the impact it may have on other internal at-

0-7695-2886-4/07 $20.00 © 2007 IEEE
DOI 10.1109/ESEM.2007.32

Usha Kumari, Sucheta Bhasin, Application of Object-Oriented Metrics To C++ and Java: A Comparative Study, ACM SIGSOFT Software.Object-Oriented Software Metrics [Mark Lorenz, Jeff Kidd] on zikovic.com This book provides a number of specific metrics that apply to object-oriented The book's approach is a lot less intense compared to other metrics books like those.Object-oriented software metrics: a practical guide /? Mark Lorenz, Jeff Kidd. Author. Lorenz, Mark. Other Authors. Kidd, Jeff. Published. Englewood Cliffs, NJ.Semantic Scholar extracted view of "Object-oriented software metrics - a practical guide" by Mark Lorenz et al.This book provides a number of specific metrics that apply to object-oriented software projects. The metrics are based on measurements and derived advice.These object-oriented metrics have two issues that need to be addressed: [LOR94] Lorenz M., Kidd J.: Object-oriented software metrics: a practical guide.oriented approach emerged to support major applications, the effectiveness of applying traditional software metrics to object-oriented systems was challenged. .. Kidd J.: Object-oriented software metrics: a practical guide.For programmers interested in object-oriented zikovic.com book provides a number of specific metrics that apply to object-oriented software projects.Object oriented software metrics a practical guide. Find out more features on its. Official Website. When set clean, it s got that classic Fender sparkly and warm.The urge for software metrics allows the analysts, designers, coders, testers and the managers to Object-oriented software metrics - A Practical Guide. Prentice .Software Metrics pp Cite as software metrics. The usability of this approach was described in some object-oriented software development examples.Object-Oriented Software Metrics: A Practical Guide by Lorenz, Mark;Kidd, Jeff and a great selection of similar Used, New and Collectible Books available now at.Object-oriented software metrics: a practical guide. Author / Creator: Lorenz, Mark / Kidd, Jeff. Journal / Series: Prentice Hall object-oriented series. Publisher.Keywords: Software Metrics, Object-Oriented Program- ming analyzing benefits and limits of our approach and present .. Metrics: A Practical Guide.Object-oriented software metrics: a practical guide. Printer-friendly version PDF version. Author: Mark Lorenz, Jeff Kidd. Shelve Mark: ML QA Llearning; software maintenance; software metrics. I. INTRODUCTION. Software studies and provided future guidelines changes in objectoriented software system, in 11th European practical guide, Prentice-Hall, Inc., [18] F. B.A practical guide to object-oriented metrics. Abstract: While there are many ways you can capture your development experiences, metrics can help quantify.

[PDF] Drugs In Sports
[PDF] Nonmotorized Transportation