IRM PRESS
701 E. Chocolate Avenue, Hershey PA 17033-1117, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.irm-press.com
ITB9322

Chapter IX

Software Metrics, Information and Entropy

Jana Dospisil
Monash University, Australia

ABSTRACT

This chapter describes the foundation and properties of object-oriented software measures. Many software measures for object-oriented applications have been developed and tested in the development environment. However, the process of defining new measures is still alive. The reason for such development lies in difficulties associated with understanding and maintaining object-oriented applications. It is still difficult to relate the measures to the phenomena we want to improve. Do our measurements indicate problems in reliability, maintenance, or too much complexity of some portions of the application? In order to reduce the complexity of software, new development methodologies and tools are being introduced. An example of the new approach to development is separation of concern. The tools, such as Aspect/J (Kiezales et al., 1997) or Hyper/J (Ossher & Tarr, 1998), facilitate the development process. There does

not seem to be a sound metrics suite to measure complexity and efficiency of applications developed and coded with Aspect/J or Hyper/J. In this chapter, we attempt to review the current research into object-oriented software metrics and suggest theoretical framework for complexity estimation and ranking of compositional units in object-oriented applications developed with Hyper/J.

INTRODUCTION

Software metrics is a term used to describe a wide range of techniques concerned with measurement of the quality of software products. These techniques range from collecting quantitative aspects of code (e.g., Lines Of Code) to models used for prediction of software quality. The objectives of software engineering are to improve the quality of developed software and provide tools for reducing software complexity. These objectives can lead to reduced cost for software development, facilitate maintenance and allow evolution and extension of the software.

For some time it has been estimated that over 70 percent of software development is spent in testing and maintenance of software (Zuse, 1994). The reports from large commercial projects, which utilize the object-orientated techniques, indicate that the expected cost savings in maintenance have not been delivered. The increased **complexity** and size of software projects have led to the development of many different concepts for breaking a system into less complex and manageable modules (Dijkstra, 1976).

The principle of **separation of concerns** also made its way into object-oriented design. The source of the problem in software development is that some kinds of behavior or functionality cross cut or are orthogonal *to* classes in many object-oriented components, and they are not easily modularized to a separate class. Examples of such behavior include the following: synchronization and concurrency, performance optimization, exception handling and event monitoring, coordination and interaction protocols, and object views.

Recent research at Xerox PARC in aspect-oriented programming (AOP) and a multidimensional separation project at IBM seek to alleviate this. This new programming model language constructs and compiles interleave components and aspects or concern definitions (programs) appropriately to formulate a unified and executable program. The objective is to reduce the complexity and promote easy maintenance.

The chapter is organized as follows: The chapter begins with an introduction of object-oriented structures and provides an overview of established metrics. Then, it introduces the notion of complexity and **entropy-based metrics** for complexity. It is followed by a section that deals with the concept of separation of concern, and provides the theoretical underpinning of Hyper/J. Furthermore, the proposed metric suite for Hyper/J is presented.