

Coupling Measurement in Multi-Kernel-Based Software with Its Application to Darwin

Liguo YU*, Stephen R. SCHACH**, Kai CHEN⁺ and Srini RAMASWAMY⁺⁺

Abstract— This paper addresses software dependency by presenting a multi-kernel-based software model and introducing the concept of directional coupling (d-coupling). A six-level d-coupling model to represent the dependencies of a kernel module on other modules in a multi-kernel-based software system is defined. This model can be used to represent the dependencies induced by all types of software coupling. The dependency model is applied to evaluate Darwin, a dual-kernel-based operating system. The study shows that (1) few strong dependencies exist between Darwin kernel modules and other modules; (2) from version XNU-517 to XNU-792, Darwin has been restructured to reduce the number of high level dependent modules induced by high level global variables to mitigate the effect of the increase of the dependency due to the growth of the kernel size and the product size.

Index Terms— Dependency, Coupling, Common Coupling, Multi-Kernel-Based Software, Darwin

1. INTRODUCTION

Component-based software engineering is the production of software products through the systematic integration of existing software components. Quite frequently, existing components are not like ready-to-use building blocks, especially in the case of large-scale design-level reuse. Instead, these components need to be modified to meet the specific requirements of the new product. Furthermore, reused software components also need to be updated to meet new requirements or changes in the environment. Therefore, reusability and maintainability are two important properties of a software component [1, 2].

A considerable amount of work has been done to try to characterize reusable [3–7] and maintainable [8–10] software components. Both reusability and maintainability are related to coupling. If a software component is relatively independent, that is, if there are only a few dependencies of this component on other components, it would be easy to understand, maintain, and reuse this component. Coupling reflects the modifiability, maintainability, and reusability of a software product [11]. Certain types of coupling, especially common coupling, are considered to present risks for software development, reuse, and maintenance [12–15]. Hence, a maintainable and reusable component should be as independent as possible.

Many software products, including most operating systems, are kernel-based. That is, as is further explained in Section 3, each implementation consists of the required kernel components, together with specific optional nonkernel components. The word “kernel” is overloaded. It can refer to a nucleus that can execute certain instructions [16, 17], or to a set of modules that are included in every installation. In this paper, “kernel” is used in the latter sense. In previous work [13–15], a categorization of common coupling was presented to measure the maintenance effort [13] and reuse effort [14] of a single-kernel-based software system. However, in software product lines and certain operating systems, closely related reused components are not always assigned to a single kernel but may be distributed among several kernels. This kind of system is called multi-kernel-based software. This paper presents a model to evaluate the dependencies induced by all types of coupling in systems with zero or more kernels. This model is then used to evaluate the dependencies of Darwin’s two kernel-based components.

The remainder of the paper is organized as follows. Section 2 discusses software coupling. Section 3 describes multi-kernel-based software. Section 4 presents the coupling model for multi-kernel-based software. Section 5 analyzes various kinds of coupling, with special attention given to common coupling. Section 6 presents the application study of Darwin.

2. SOFTWARE COUPLING

Coupling is a measure of the degree of interaction between two software components (classes, modules, packages, or the like). Many different types of coupling have been identified, including data coupling, stamp coupling, control coupling, and common coupling [11, 18, 19]. Table 1 lists the definitions of several major types of coupling. The degree of dependency is considered in increasing order from top (data coupling) to bottom (common coupling). A good software system should have low coupling between components. Common coupling is considered to be a strong form of coupling, that is, it induces strong dependencies between software components, making software components difficult to understand, maintain, and reuse [12].

Manuscript received December 2, 2007, Revised March 12, 2008.

* Liguo Yu is an assistant professor of the Computer Science Department at Indiana University South Bend (e-mail: ligyu@iusb.edu).

** Stephen R. Schach is an associate professor in the Department of Electrical Engineering and Computer Science at Vanderbilt University (e-mail: srs@vuse.vanderbilt.edu).

+ Kai Chen is with Google, Inc. (e-mail: kai.chen@gmail.com).

++ Srinivasan Ramaswamy is professor and chairperson of the Computer Science Department at University of Arkansas at Little Rock (e-mail: srini@ieee.org / srini@acm.org).

