Lightweight Dependency Models for Product Lines

Neeraj Sangal
Lattix, Inc.
neeraj.sangal@lattix.com

Abstract

This tutorial presents a practical technique for managing the architecture of software product lines using inter-module dependencies. We will show that the Lightweight Dependency Model approach, based on dependency structure matrices, is highly scalable compared to the directed graph approaches that are common today. We will also show a variety of matrix algorithms and transformations that can be applied to analyze and organize the system into a form that reflects the architecture and demonstrates the importance of managing dependencies in product lines. We illustrate our approach by applying it to real applications each consisting of hundreds or thousands of files. We will show how dependency models can be created for product lines and how formal design rules can be specified to manage the evolution of these architectures.

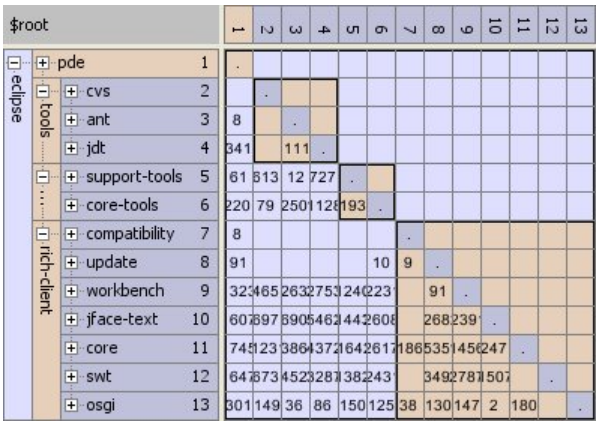
1. Introduction

Excessive inter-module dependences in software can make modules hard to develop and maintain because they cannot be understood easily in isolation, and the effects of changes to one module propagate to many others. Managing inter-module dependencies is important in product line architecture. When a module is shared across multiple products, all modules that this module depends on will also have to be shared or replicated in all of those products.

2. Dependency Structure Matrix for Software Architecture

The traditional application of the dependency structure matrix, or DSM, has been for the improvement of discrete product development processes. The structure of dependences amongst the tasks to be performed is a strong indicator of the efficiency of the process as a whole [1]. If tasks are tightly coupled, with many cyclic dependences, rather than working as a smooth pipeline, the tasks stall frequently, and will need to be repeated because of dependence on tasks that follow them.

In the software domain, tasks naturally correspond to software modules with the matrix representing module dependences [2]. Partitioning and clustering algorithms provide an automatic mechanism for architectural discovery in a large code base. The distinction between acceptable and unacceptable dependencies is expressed using design rules, which are provided by the user, and applied to the displayed DSM, in order to highlight the dependences that violate the intended architectural design.



DSM for the Eclipse Platform

3. References

[1] Steven D. Eppinger, "Innovation at the Speed of Information", *Harvard Business Review*, January 2001.

[2] Neeraj Sangal, Ev Jordan, Vineet Sinha, and Daniel Jackson. Using Dependency Models to Manage Complex Software Architecture. Proceedings of the 20thACM/SIGPLAN OOPSLA, San Diego, CA, October 2005. ISBN1-59593-031-0.