Dependency Management Techniques

Summary

Why is software so difficult to change? When you establish your initial vision for the software's design and architecture, you imagine a system that is easy to modify, extend, and maintain. Unfortunately, as time passes, changes trickle in that exercise your design in unexpected ways. Unlike what you had anticipated, each change begins to resemble nothing more than another hack, until finally the system becomes a tangled web of code that few developers care to venture through. Eventually, modifications to the software intended to improve the system have the opposite affect of breaking other parts of the system. The software is beginning to rot.

The most common cause of rotting software is tightly coupled code with a heavy dependency graph. In this session, we'll explore the most common symptoms of rotting design, examine their root cause, and present techniques and patterns that have been used on a number of real world projects to help manage dependencies across classes, packages, and the binary units of deployment.

Audience

Dependency Management Techniques will benefit **software architects** and **software developers**. They will learn new ways to minimize coupling at the class, package, and module level. In addition to discussing principles that help during initial design, we will also explore refactoring techniques that can help reduce coupling within legacy code.

Content Outline (90 minutes)

1. **Introduction (15 minutes)**
   Goal: Explore the challenges maintaining, testing, and growing a codebase with a complex dependency graph. Introduce examples supporting this position.
   Topics Include:
   - Hindering Maintainability
   - Preventing Extensibility
   - Restricting Testability
   - Inhibiting Reusability
2. **Class Level Dependencies (25 minutes)**
   Goal: Present types of class level dependencies, examining the challenges presented and exploring alternative designs that reduce complex class level dependencies.
   Topics Include:
   - Static Dependency and Inheritance
   - Bi-Directional Associative Relationships
   - Run-time versus Compile-time dependencies
3. **Package Level Dependencies (25 minutes)**
   Goal: Introduce package level dependencies, examining the challenges presented and exploring ways to minimize package level dependencies through design patterns and metrics.
   Topics Include:
   - Direct and Indirect Dependencies
   - Acyclic versus Cyclic Dependencies
   - Modeling Package Dependencies

- Analyzing Package Dependencies using JDepend

4. **Module Level Dependencies (25 minutes)**
   Goal: Introduce module level dependencies, examining the challenges presented. Numerous heuristics will be presented that offer ways to minimize module level dependencies.
   Topics Include:
   - Dependencies among Binary Unit of Deployment
   - Acyclic versus Cyclic Dependencies
   - Avoiding Cyclic Dependencies using Callbacks, Escalation, and Demotion
   - Inverting the Dependency Graph
   - Analyzing Module Dependencies using JarAnalyzer

Presenter Background

Kirk is the Senior Technology Strategist at TeamSoft, Inc. (http://www.teamsoftinc.com), where he leads based on his firm belief in the pragmatic use of technology. In addition to his work on enterprise development projects, Kirk shares his experiences through courseware development and teaching, writing, and speaking at seminars and conferences. Kirk has provided training and mentoring to thousands of software professionals, teaching courses on object-oriented development, Java, software architecture, software process, and UML.

Kirk is the author of Java Design: Objects, UML, and Process, and the founder of www.extensiblejava.com, a growing resource of design pattern heuristics that emphasize greater component modularity in large scale enterprise software projects. He is a frequent contributor to The Agile Journal, where he writes The Agile Developer column. He is also the creator of JarAnalyzer and AssAnalyzer, utilities for identifying and managing the physical dependencies between Java .jar files and .Net dll assemblies, respectively. files. His personal website is www.kirkk.com, his blog is http://techdistrict.kirkk.com, and his planet is http://planet.kirkk.com.

Tutorial History

This tutorial is presented on the NoFluffJustStuff Java Software Symposium Tour. It's origins are traced back to work conducted on multiple projects over several years where special emphasis was placed on minimizing dependencies between software components.

Presenter Contact Information

Kirk Knoernschild
email: pragkirk@kirkk.com