Agile Architecture

Summary

Traditionally, we attempt to make the right architectural decisions early due to the significant anticipated cost affiliated with making incorrect decisions. But this contradicts agile practices which have taught us to embrace change. So how do agile and architecture come together? Conceptually, the goal of agile architecture must be to eliminate the architectural significance of change by crafting software that can easily adapt to change. In practice, developing agile architecture is much more difficult.

Software architecture is organic. The architectural goals you set to achieve early in the development effort differ from those you'll need to satisfy later. Change occurring throughout the software development lifecycle impacts architecture. The ability to accommodate shifts in architecture is directly related to the dependencies between software modules. In this session, we examine patterns and principles that lead to agile architecture. Extensive discussion is devoted to modularizing units of deployment, and how code can be created and organized to create more flexible, reusable, maintainable, extensible, and testable software components. Topics of discussion range from coupling between classes to the dependency structure between your modules of deployment. Examples gleaned from real world J2EE development effort will be used in a pragmatic case study that examines the evolution of a system from its initial deployment through a number of refactorings that positively influenced the architectural resiliency of the final product. A suite of heuristics will be introduced that allow you to apply many of these concepts immediately.

Audience

Agile Architecture will benefit **software architects** and **software developers**. They will learn new ways to structure large software systems, resulting in extensible and architecturally resilient software. Using an extensive case study, numerous examples will be shown that illustrate how a system's architecture can evolve while maintaining a high degree of stability. A suite of heuristics will be presented as we progress through the presentation, offering attendees ideas that can be applied immediately to their projects.

Content Outline (90 minutes)

1. **Introduction (15 minutes)**
   Goal: Define software architecture using popular definitions by Grady Booch and Ralph Johnson
   Topics Include:
   - Architecture as a structural entity
   - Architecture as a social construct
   - Subsystems and Interfaces
2. **Logical versus Physical Design (25 minutes)**
   Goal: Explore the differences between logical and physical design. Examine how logical design influences physical design and vice versa.
   Topics Include:
   - Logical Design: Maintenance and Extensibility
   - Physical Design: Modularity and Reuse

- Physical Units of Deployment
- Component Relationships
- Direct versus Indirect Relationships

3. **System Case Study (50 minutes)**
   Goal: Examine a system's initial architecture and explore through numerous refactorings how the architecture is modified. A suite of heuristics will be presented as each example is built on the previous, offering an experience very much like one would experience on a real world project.
   Topics Include:
   - Layering Modules
   - Concrete versus Abstract Components
   - Leveling Relationships
   - Separating Abstractions
   - Inverting Component Relationships

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 was presented at numerous conferences, including Software Development Best Practices, Architecture & Design World, SD Expo, and 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