Modularity, Agility, and Architecture’s Paradox

AOSD 2011
March 25th, 2011
Kirk Knoernschild
pragkirk@kirkk.com

“First they ignore you, then they ridicule you, then they fight you, then you win.”

--- Mahatma Gandhi

Modularity is disruptive and will transform how enterprise applications are designed, developed, and managed!
1972
(or a bit before)

Modularity, Agility and Architecture’s Paradox

- Complexity
- Architectural Agility
- Paradox
- Modularity
Modularity, Agility and Architecture’s Paradox

Complexity

Why is software so complex?

How do we tame complexity?

What’s the role of modularity?

- 120 billion loc in 1990
- 250 billion loc in 2000
- loc doubles every 7 years
- 50% of development time spent understanding code
- 90% of software cost is maintenance & evolution

Source: http://users.jyu.fi/~koskinen/smcosts.htm

Perspective: Not only double the past 7 years, but more than total amount ever written combined!
Lehman’s Law

As a system evolves, its complexity increases unless work is done to maintain or reduce it.

www.lumaxart.com/
We are often asked to design solutions to problems that require knowledge we currently do not possess.

**Gall’s Law**

A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.
Modularity, Agility and Architecture’s Paradox

Architectural Agility

What is architecture?

What is the goal of architecture?

What is architectural agility?

Architectural Agility

- What is Architecture?

  An architecture is the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural elements and behavioral elements into progressively larger subsystems, and the architecture style that guides this organization -- these elements and their interfaces, their collaborations, and their composition.


© 2009 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. or its affiliates.
Architectural Agility

• What is Architecture?

- In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called ‘architecture.’ This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers...Architecture is about the important stuff. Whatever that is.


Architectural Agility

• What is Architecture?

- The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.

Source: ANSI/IEEE Std 1471-2000
Architectural Agility

• What is Architecture?
  - A formal description of a system, or a detailed plan of the system at component level to guide its implementation
  - The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time.
  

Architectural Agility

• What is Architecture?
  - Architecture embodies the critical design decisions that typify a system.
    • Relates to cost of change, organizational structure, structure of code, capabilities of a system, etc.
  - The significance of decisions needs to be understood and assessed
    • A heavy-weight approach is likely to reduce understanding and our ability to assess

Architectural Agility

What if we were able to reduce the impact and cost of change?

We need to eliminate architecture!

---

First and foremost, we should try to make most decisions reversible, so they can be made and then easily changed.

-- “Implementing Lean Software Development: From Concept to Cash”

Irreversible Decisions should be made as late as possible!
Architectural Agility

- Reduce Impact and Cost of Change
  - Makes Decisions Reversible
    - Encourages Evolutionary Architecture
  - Lehman’s Law
    - Goal of Architecture
      - systems that work

Modularity, Agility and Architecture’s Paradox

Paradox

- How do we design more flexible software systems?
- How do we realize the dream of reuse?
- Are we doomed to failure?
**Paradox**

*Flexibility* (Reuse, Compose, Extend, Lightweight, Fine-Grained, ...)

*Complexity* (Use, Maintain, Understand, ...)

... making everything easy to change makes the entire system very complex...

- Ralph Johnson in "Who Needs an Architect"

**Paradox**

Maximizing reuse complicates use

- AOP
- SOLID
- Design Patterns
- Modularity
- SOA Principles
Paradox

Flexibility
(Reuse, Compose, Extend, Lightweight, Fine-Grained, ...)

We must recognize which areas of the system demand the increased complexity that will bring greater flexibility!

Increasing evolvability decreases survivability

Complexity (Use, Maintain, Understand, ...)

... making everything easy to change makes the entire system very complex...

- Ralph Johnson in "Who Needs an Architect"

Paradox

Reduce impact and cost of change

Makes decisions reversible

Encourages evolutionary architecture

Lehman's Law

Goal of Architecture

Gal's Law

Effort

Systems that work

Flexibility

Complexity
Modularity, Agility and Architecture’s Paradox

Modularity

How does modularity help increase architectural agility?

How does modularity help us realize reuse?

How does modularity help us overcome the paradox?

Question

How do we manage software complexity and increase architectural agility?

Answer

Modularity
Modularity

- reuse
- reduce complexity
- ease maintenance
- increase extensibility

Increases architectural agility!

Umm...we can already do this with objects, aspects, methods, and services!

Modularity

Granularity

Unit of Deployment

Unit of Composition

Unit of Inter-Process Reuse

Unit of Intra-Process Reuse
Modularity

What does architecture have to do with turtles?

"You're very clever, young man, very clever", said the old lady. "But it's turtles all the way down!"

-- A Brief History of Time

Modularity

Granularity

<table>
<thead>
<tr>
<th>Services</th>
<th>Modules</th>
<th>Packages</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Unit of Deployment          Unit of State

Unit of Composition

Unit of Inter-Process Reuse

Unit of Intra-Process Reuse

Reuse Release Equivalence: Unit of reuse is the unit of release!
Modularity

A module system provides a runtime environment for modules

- unit of reuse
- unit of composition
- unit of deployment
- unit of management

Hey, it's a JAR file!

Adapted from http://www.rendell.org/amt/upload/20091/tower-1209-4833.jpg
Modularity

A nice interface!

Which area of the system demands more flexibility?
Modularity as change occurs
Modularity

Infrastructure
- Runtime platform support helps enforce modular architecture.

Programming Model
- The frameworks and technologies that allow us to create modular software

Design Paradigm
- The techniques used to identify and create the right set of modules

The Design Paradigm
- What’s the right granularity for a module?
- What the right weight for a module?

Few teams are designing modular software systems today!

POLL:

**Question:**

- How many design class relationships?  
- How many design package relationships?  
- How many design service relationships?  
- How many design module (JAR, Assembly) relationships?

**Response:**

- 98%  
- 25%  
- 75%  
- < 10%
Why aren’t we designing more modular software?

Platforms discourage modularity!

This is the next generation application platform!

- Dynamic deployment
- Multiple versions
- Explicit dependencies
These, and many other ideas, are discussed in my upcoming book: 

“Patterns of Modular Architecture”

Draft manuscript available at:

http://modularity.kirkk.com

Code available at:

https://github.com/pragkirk/poma

- Additional Resources

  - Patterns of Modular Architecture - Book in Progress
    - http://modularity.kirkk.com/
  - http://www.osgi.org - OSGi HomePage
  - OSGi in Practice by Neil Bartlett
    - http://neilbartlett.name/blog/osgibook/
  - Modular Java by Craig Walls
  - OSGi and Equinox: Creating Highly Modular Java Systems by Jeff McAffer, et. al.
  - Tried to develop a modular architecture without OSGi?
    - JarAnalyzer - http://code.google.com/p/jaralyzer/