CSE 218: Advanced Topics in Software Engineering
Software Architecture for Distributed and Reactive Systems

Ingolf H. Krueger
ikrueger@ucsd.edu
Always design your programs as a member of a whole family of programs, including those that are likely to succeed it.

Edsger W. Dijkstra
Overview

- Goals for this class
- Software Quality
- Why is High-Quality Software so Difficult to Build?
- Teamwork Projects
- What’s next?
Goals for this Class

• Get you interested in the field of software architecture

• Build understanding for the challenges in creating and documenting “good” software architectures for complex, distributed, reactive systems

• Study description techniques, formal foundations and methodologies for analyzing and constructing software architectures

• Consider current infrastructures and middleware technologies for implementing software architectures, such as .NET, JINI, and CORBA

Have Fun!
Overview

- Goals for this class
- **Software Quality**
- Why is High-Quality Software so Difficult to Build?
- Teamwork Projects
- What’s next?
What is Software Quality?

• Our goal:
  – Build high-quality software systems

• Quality attributes:
  – Reliability
  – Usability
  – Understandability/Modifiability
  – Efficiency
  – Testability/Verifiability
  – Portability
  – ...

What is Software Quality?

• Error:
  – Divergence from the customer’s requirements, or
  – Inconsistency within the requirements

• 55% of all errors are *made*, but less than 10% are *detected* during requirements capture and analysis

• Delayed error detection induces exponential cost increase for error correction

• Economic incentive for concentrating on early phases of software development
Overview

- Goals for this class
- Software Quality
- Why is High-Quality Software so Difficult to Build?
- Teamwork Projects
- What’s next?
Why is High-Quality Software Difficult to Build?

• Software-Quality is difficult to measure

• Software is easily changeable at any stage during the development process

• Implementation platforms and technologies change rapidly

• Complexity is rapidly increasing
  – For every 25% increase in problem complexity there is a 100% increase in complexity of the software solution*

• Adequate methods and tools do not exist or are only partially adopted

• Requirements change

*see [Glass02]
How Complex are These Systems Anyhow?

- Telecommunications
- Networking
- Embedded Systems, Automotive
- Business Information Systems (Web Services)
- Mobile, dynamic services, ad-hoc networks
Overview

- Goals for this class
- Software Quality
- Why is High-Quality Software so Difficult to Build?
  - Teamwork Projects
- What’s next?
Project Goal

• Foster practical application of concepts studied in class
• Experience teamwork
• Study “real” issues in architecture design, implementation and deployment beyond toy problems
• Investigate deployment options and technologies
• Perform architectural “spiking”
• Develop experience in service-oriented design
Artifacts

• All teams produce
  – Domain model
  – Architecture documentation
  – Executable (sub)system solution

• Result priorities
  – A solid, consistent, thought-through, understandable architecture is most important
  – Based on a thorough understanding of the system domain, as expressed in the domain model
  – Deliver a working solution following the created architecture
  – Focus on feasible solution, always as simple as possible (KIS), without harming good software engineering and architecture principles
Overview

- Goals for this class
- Software Quality
- Why is High-Quality Software so Difficult to Build?
- Teamwork Projects
- What’s next?
Overview of Introductory Material

- The Notion of Architecture and Architectural Aspects
- The Role of Software Architecture
- Modeling and Documenting Architectures
- Patterns for Architecture and Design
- Refactoring Techniques
Literature


Literature

[GOF95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns – Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995


