Introduction to Software Architecture

CSCE 742 - Lecture 1 - 08/23/2018

Today's Goals

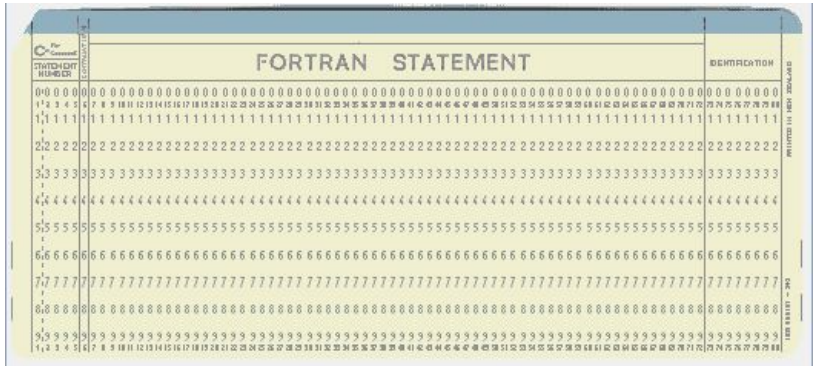
Introduce The Class

- AKA: What the heck is going on?
- Go over syllabus
- What you should already know
- Clarify course expectations
- Assignments/grading
- Answer any questions
- Cover the basics of software architecture

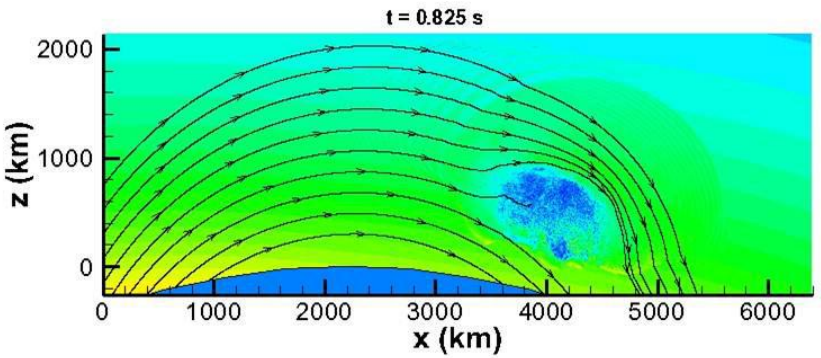
In the beginning…

Software Was Small

- Both physically…



- And in scope.



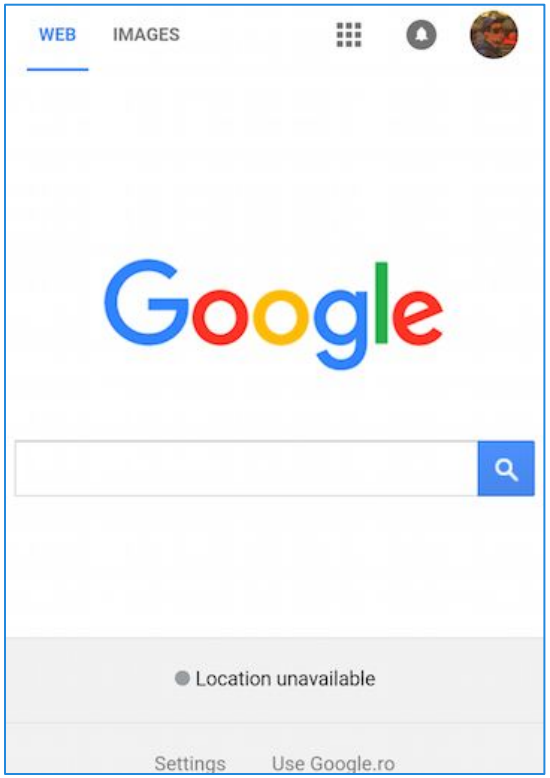
Software Starts to Grow Up

- Languages like C introduce file linking.
  - Enables organization of code and reuse of code.
- SIMULA-67, Smalltalk introduce objects.
  - Enables organization of code into focused units that work with other objects to perform larger tasks.
    - Sections of the code “activate” when needed.
    - We can group together related functionality, ignore unrelated functionality, and find what we need when making changes.
    - Code can be reused in future projects.

Flash forward to the present day...

Our Society Depends on Software

This is software:

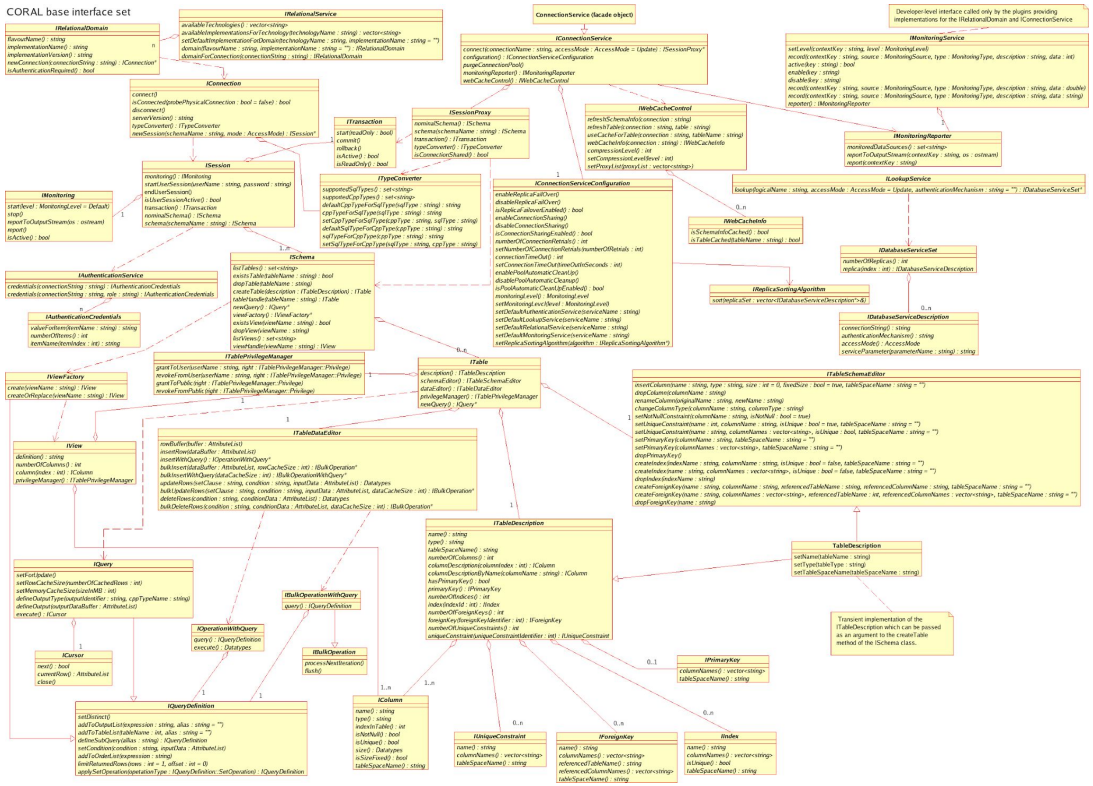


So is this:

Also, this:

Under the Hood

- Systems have millions of lines of code.
- Divided into hundreds of classes.



Growing Pains

- No person can understand an entire million-line codebase.
- Classes organize code, but how can you find the right clases when there are thousands?
- Results in chaos.
  - Only 16.1% of projects delivered on time and within budget, with all planned features complete as specified.
  - 31.1% of projects are cancelled before delivery.
  - Delivered projects may be slow, insecure, missing features, have duplicate code, go down often, etc.

The Concept: “Architect” Software

- The key to delivering robust software?
  - Designing an understandable, organized system.
  - **AKA: “taming the complexity”**
- **Architecting software** is the practice of partitioning a large system into smaller ones.
  - That can be created separately
  - That individually have business value
  - That can be straightforwardly integrated with one another and with existing systems.

Architectural Styles

What is Software Architecture?

"Software architecture is the **fundamental organization** of a system, embodied in its **components**, their **relationships** to one another and the environment, and the **principles** governing its design and evolution."

- IEEE Definition

Architectural Design

- First stage of design.
- **Partitions** the requirements into self-contained subsystems.
  - Later, each subsystem will be decomposed into one or more classes.
- Plan how those subsystems cooperate and communicate.

Architecture Parallels

- Architectural plans are the technical interface between the customer and the contractor building the building.
  - (and the software)
- A bad architectural design for a building cannot be rescued by good construction.
  - (same for software)
- There are specialist types of building architects and architecture styles.
  - (you get the point)

Why Explicitly Plan Architecture?

- Enable stakeholder communication
  - High-level presentation of the system.
- Enables system analysis
  - Can look for problems before coding.
- Enables large-scale reuse
  - Planning subsystems as independent entities allows their reuse in other systems.
- Bad architectural design means bad security
  - Controlling access is the first line of defense.

In This Course, We Will Discuss...

- The purpose and role of architecture in the overall process of software development.
  - As a process (architecting a system) and an artifact (the architecture of a system).
- Similarities and differences between "design" and "architecture".
- Notations/tools to assist software architects.
- Processes that lead to good architectural outcomes and architectural refactoring.

Desired Course Outcomes

1. The students will be able to work from stakeholder requirements to create partitionable systems.
2. ... use different viewpoints to document software architectures to different stakeholders.
3. … understand architectural quality attributes and how to use perspectives to assess how well the architecture meets them.
4. … apply and understand architectural patterns to quickly examine architectural alternatives and choose between them.
5. … clearly present and advocate architectural ideas.

Lecture Plan (approximate)

- Introduction and Fundamentals of Architecture (1 week)
- Introduction to Viewpoints, Perspectives, and Architectural Definition (1 week)
- Context, Concerns, Stakeholders, and Quality Attributes (1 week)
- Scenarios and Components (1 week)
- Architectural Styles (4 weeks)
- Viewpoints (2 weeks)
- Perspectives (2 weeks)
- Other Topics (1-2 weeks)

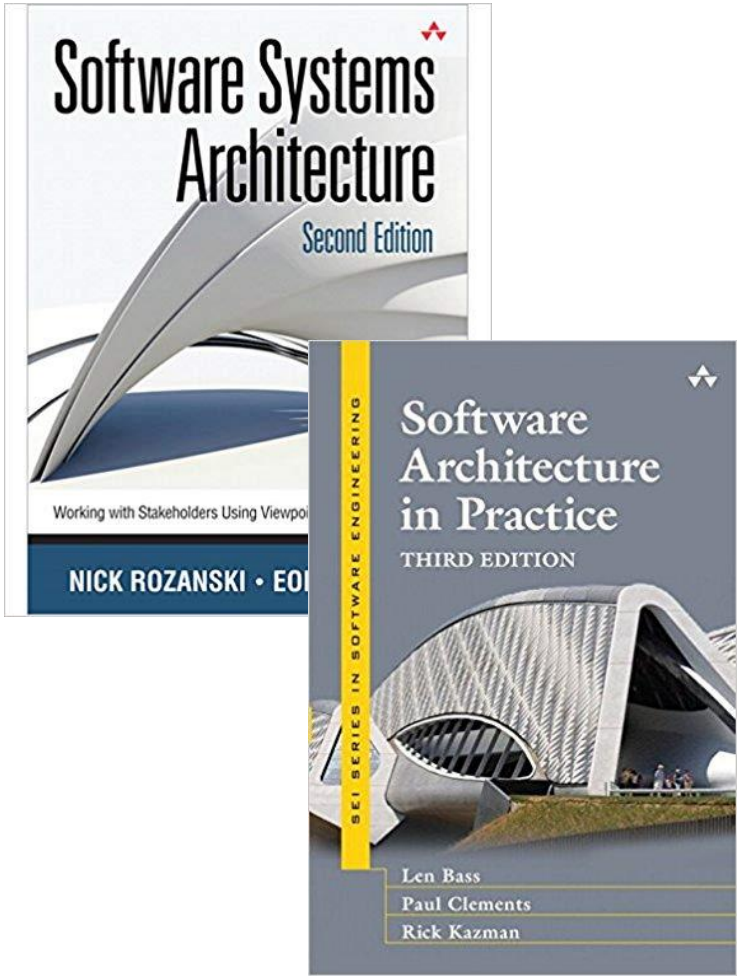
Contact Info

- Instructor: Greg Gay (Dr, Professor, $#*%)
  - E-mail: ggay@cse.sc.edu
  - Office Hours: T/Th, 4:00-5:00 PM, 2247 Storey Engineering and Innovation Center
- Website:
  - http://dropbox.cse.sc.edu/course/view.php?id=95
    - (Main website - will be used for course material and assignment submission)
  - http://greggay.com/courses/fall18csce742/
    - (Static backup - somewhat behind, but useful if above is down)

Textbook

There is no required book.
Source material is drawn from:

- *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives* (Second Edition). Nick Rozanski and Eoin Woods.
- *Software Architecture in Practice* (Third Edition). Len Bass, Paul Clements, Rick Kazman.
- Additional readings
  - 4-5 over the semester
  - Will be made available on webpage

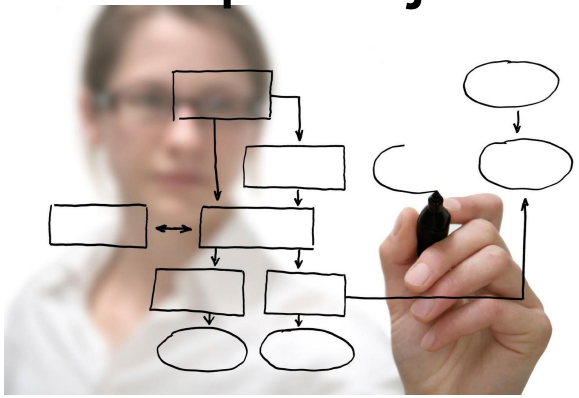


Learning Modes

Lectures/Textbook

Class Discussions

Group Project



Prerequisites

CSCE 740 - Software Engineering

- Not essential, but very helpful.
- We will use UML diagrams.

You need to be proficient in programming

- Examples: Java, C, C++, JavaScript
- You should be able to read and write programs without additional instruction.
- This is **not** a programming language class.

You need a basic understanding of algorithms, logic, and sets.

Assignments and Grading

- Group Assignments (40% in total)
  - Groups of 3.
  - Frequent peer evaluations.
- Individual Assignments (10%)
  - Reading + 1 page summary
- Midterm/Final Exams (20% each)
- Participation (10%)
  - In-class activities.
  - Group participation.
  - Answering questions.

Expected Workload

This class can be time consuming.

- Understanding the material takes time.
- Project work requires team coordination.

Do not underestimate the project work.

- Good engineering is hard.
- Planning and scheduling your time is essential.
- Do NOT delay getting started.
- Appoint a team leader (and rotate the role)

Feedback

Problems with assignments, course questions, feedback?

- Contact me! I like feedback!

Problem with instructor

- Also contact me
- Contact CS front office

Other Policies

Integrity and Ethics:

The homework and programs you submit for this class must be entirely your own. If this policy is not absolutely clear, then please contact me. Any other collaboration of any type on any assignment is not permitted. It is your responsibility to protect your work from unauthorized access.

Classroom Climate:

All students are expected to behave as scholars at a leading institute of technology. This includes arriving on time, not talking during lecture (unless addressing the instructor), and not leaving the classroom before the end of lecture. Disruptive students will be warned and potentially dismissed from the classroom.

Other Policies

Make-Up and Late Homework

- Make-ups for graded activities may be arranged if your absence is caused by a documented illness or personal emergency.
- Homework assignments are due at the time noted on the assignment handout. Late work is not accepted without prior approval. Any assignment turned in after the due date will be considered late and will be subject to a penalty of 10% per day, including weekends and holidays.

Other Policies

Diversity

Students in this class are expected to respectfully work with all other students, regardless of gender, race, sexuality, religion, or any other protected criteria. There is a zero-tolerance policy for any student that discriminates against other students.

Special Needs

We will provide, on a flexible and individual basis, reasonable accommodations to students that have disabilities that may affect their ability to participate in course activities or to meet course requirements Students with disabilities should contact their instructor early in the semester to discuss their individual needs.

Stakeholders, Viewpoints, and Perspectives

A Simple Problem?

- We need to replace the current stock management system. It should be more interactive and allow employees to process stock movement in real time, rather than waiting for next-day results.
- First step: Gather the system requirements.
  - A **requirement** is a singular documented need that a particular product must be able to perform, or a property the product must fulfill.
    “The software shall be able to calculate the sum of a column of integers.”

A Simple Problem?

- We talk to different people in the company:
  - The users at the distribution depot need different information than the users at the head office.
  - Commercial managers have a crucial need for real-time summary reporting.
  - This would slow main transaction flow, which is unacceptable to the logistics group.
  - The IT auditors need a two-year archive of all stock release authorizations. This introduces additional hardware requirements.
  - IT operations employees are not sure they can provide the needed server resources to handle two years of archives.

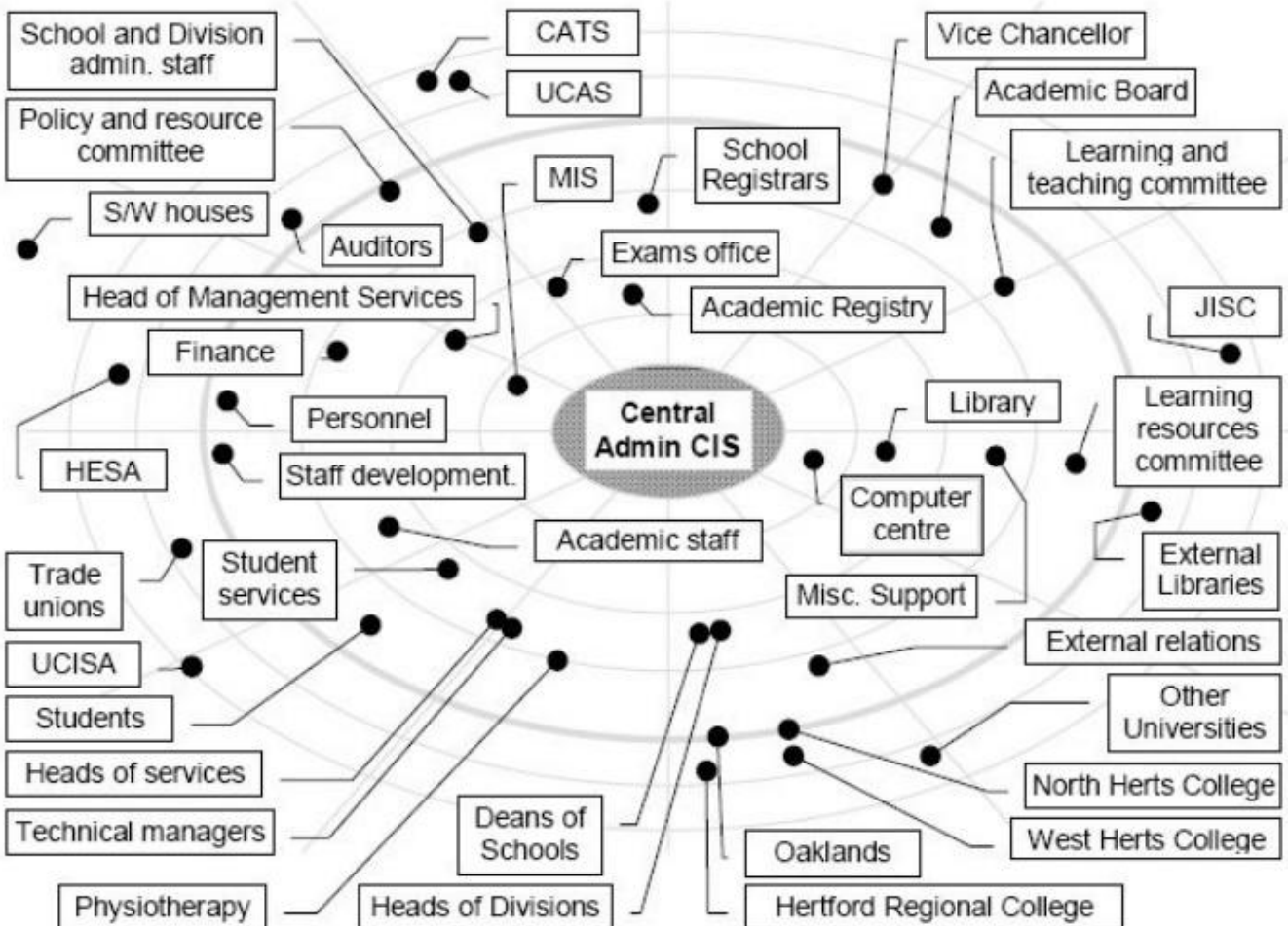
The Stakeholders

- Depot users, commercial managers, logistics, IT auditors, etc. are **stakeholders** in the project.
  - “A person, group, or entity with an interest in or concerns about the realization of the architecture.”
- The people for whom we build systems.
  - Often broad groups, rather than specific people.
- The architecture must meet their complex, overlapping, and contradictory needs.

Stakeholder Concerns

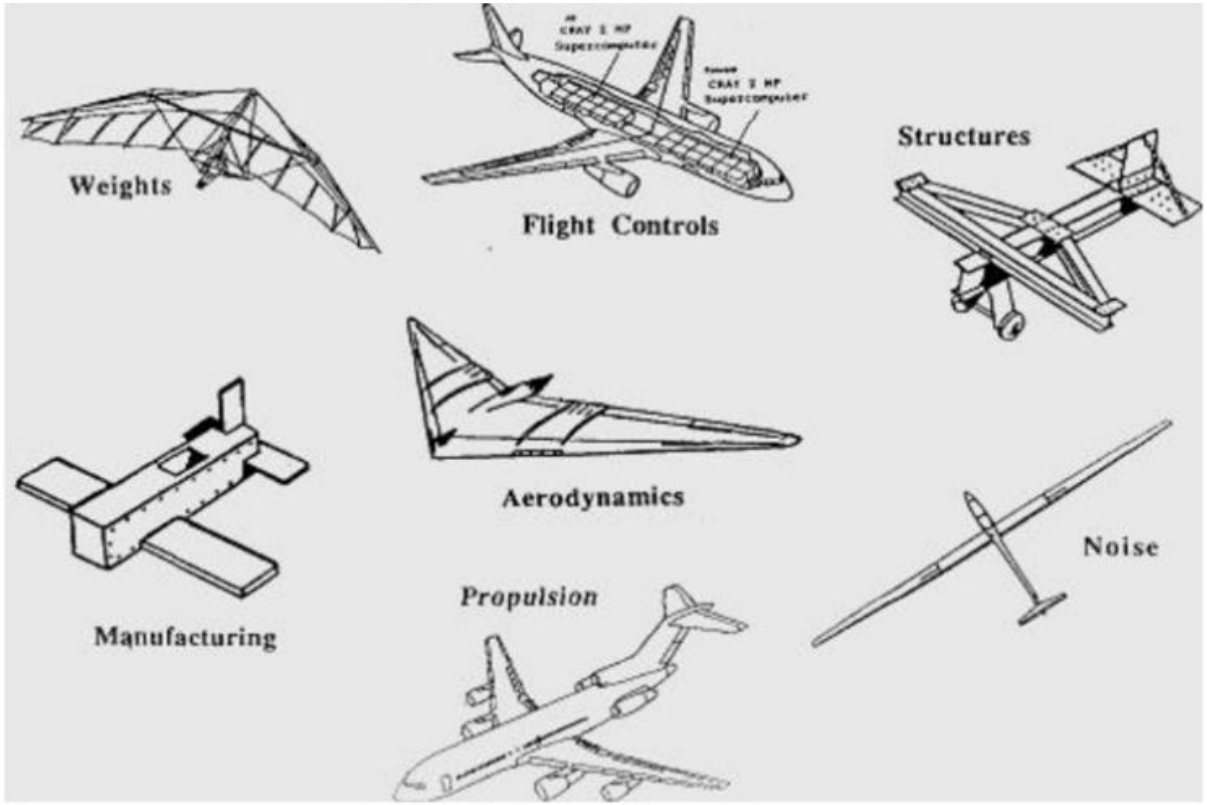
- A good architecture satisfies the concerns that stakeholders have about the project.
  - A **concern** about an architecture is a requirement, objective, intention, or an aspiration a stakeholder has for the architecture.
- If a system does not meet the needs of its stakeholders, it cannot be considered a success.
- Resolving conflicting concerns is a significant challenge.

There Can Be Many Stakeholders



The Stakeholders

The architecture can have many meanings depending on the stakeholder's perspective.



Continuing The Story...

- Armed with the requirements, you sketch the internal design of the system - identify key classes and how they work together.
  - The system will run in parallel, so you also describe processes and how they communicate.
  - You add system management components to the design to make it more manageable.
  - You add data stores and annotate the data flow between classes.
- Now, you have an architectural model to show everybody!

First Stab...

- Is this model likely to please stakeholders?
- Will this model represent a complete, feasible, optimal architecture?
- How can you describe something as multifaceted as the architecture of a complex system to the different people who need to understand it?

Architectural Views

- **Architectural views** describe *one aspect* of a system's architecture.
- By considering a system's architecture through distinct views, you can understand, define, and communicate a complex architecture in a partitioned fashion.
  - System's functional structure.
  - Information organization.
  - Deployment environment.
  - Concurrent processes.

Continuing The Story...

- You develop the architecture by exploring it from different viewpoints. However…
  - You realize that the program is *slow*. Performance didn't come up earlier - it wasn't related to what the system was supposed to do.
  - The auditing group is concerned about how data access will be controlled.
  - Management reminded everyone that systems must recover within eight hours of a major failure.
- Architecture is delivering promised features, but stakeholders might still be upset.

Quality Attributes

- The architecture not only dictates *what* the system does, but *how* it does it.
  - How *quickly* it runs.
  - How *secure* it is.
  - How *available* its services are.
  - How easy it is to *modify*.
- **Quality attributes** describe desired non-functional properties of systems.
- An architect must prioritize quality attributes and design a system that meets thresholds.

Quality Attributes

- Performance
  - The ability of a system to meet timing requirements. When events occur, the system must respond quickly.
- Security
  - The ability of a system to protect information from unauthorized access while providing service to authorized users.
- Safety
  - The ability of a system to avoid hazardous situations and act when components fail.

Quality Attributes

- Availability
  - The ability of the software to carry out a task when needed, to minimize “downtime”, and to recover from failures.
- Modifiability
  - The ability to enhance software over time by fixing issues, adding features, and adapting to new environments.
- Testability
  - The ability to easily identify faults in a system. The probability that the system will fail on its next execution.

Quality Attributes

- Interoperability
  - The ability of the software to exchange information with and provide functionality to other systems.
- Usability
  - The ability of the software to enable users to perform desired tasks and provide support to users.
  - How easy is it to use the system, learn its features, adapt the system to meet user needs, and increase confidence and satisfaction in system use?

Architectural Qualities Conflict

These qualities often conflict. It is hard to achieve multiple qualities at once.

- Using fewer subsystems improves performance, but hurts modifiability.
- Introducing redundant data improves availability, but makes security more difficult.
- Localizing safety-related features usually introduces more communication between subsystems, degrading performance.

Architectural Perspective

- Quality properties are *cross-cutting* across all architectural viewpoints.
- **Architectural Perspectives** are used to discuss how particular quality attributes affect each view of the overall architecture.
  - For example, how security impacts the functional structure, information organization, deployment environment, and concurrent processes.

Framing Device for the Class

- **Stakeholders** are the people impacted by the architecture, who have differing expectations and needs.
- **Viewpoints** are used to structure architecture definition by focusing on aspects of the system being designed.
- **Perspectives** focus on how a particular quality attribute impacts each viewpoint of the architecture.

Next Time

- More introduction:
  - Basic Architectural Definition and Styles
- Reading:
  - David Garlan and Mary Shaw. *An Introduction to Software Architecture*.
    - On course website.
  - Rozanski & Woods: ch. 1 and 2
- Plan your team selection.
  - The earlier, the better!