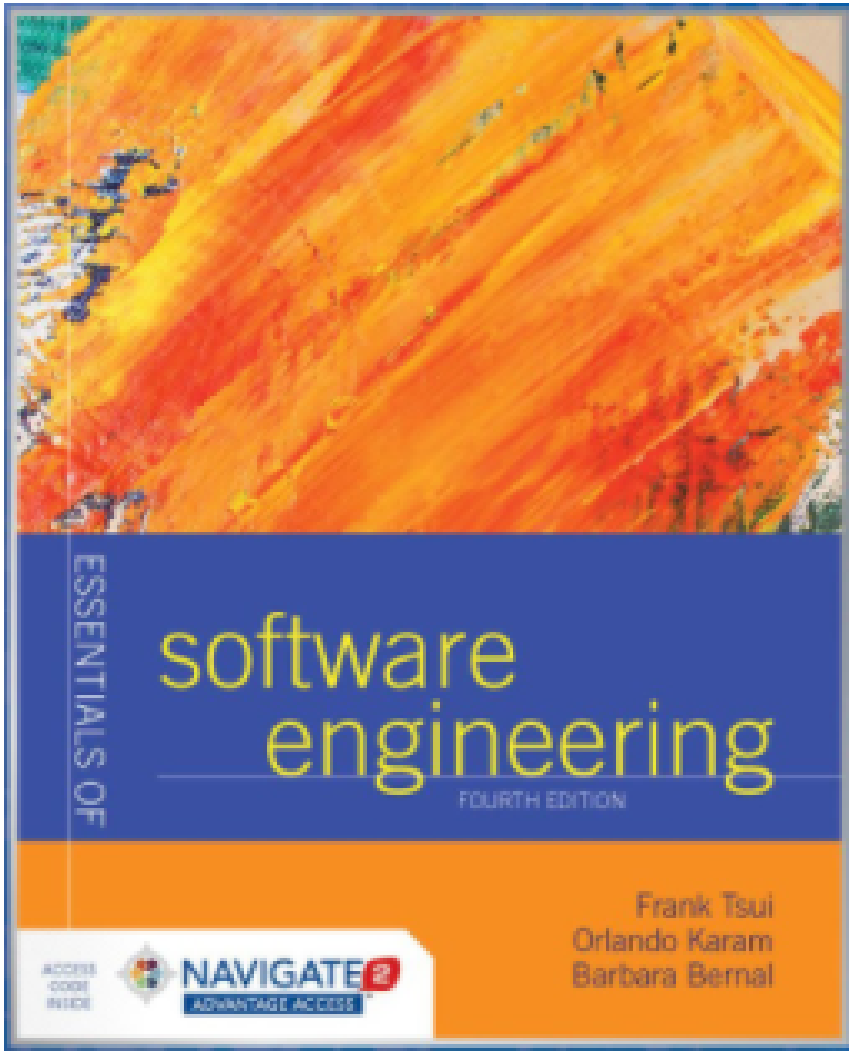
Building a System

CSC 354, Software Engineering I

Chapter 3: Building a System



Class Questions

- What is the most complex problem you have worked on?
- What was the most difficult part?
- How much design did you do before coding?

Building a System

- We are moving from writing a *program* to building a *system*
- The difference is complexity
  - Breadth of complexity
  - Depth of complexity

Complexity

- An increase in the size and complexity of a problem leads to
- An increase in *effort* due to the size and complexity, which leads to
- An increase in the size and complexity of the solution.

Complexity (continued)

- More functionalities
- More features within each functionality
- More varieties of interfaces (internal and/or external)
- More users and varieties of users
- More data, varieties of data, and/or data structures
- More linkages and connections
  - Data sharing among functionalities and logic
  - Control passing among functionalities

Complexity (aside)

- Essential complexity: how difficult something is to do regardless of how experienced you are, what tools you use or what new and flashy architecture pattern you used to solve the problem.
- Accidental complexity: non-essential complexity that is introduced into the design by mistake.
- Incidental complexity: non-essential complexity that is introduced into the design by mistake or on purpose.

Ways of Handling Complexity

- Simplification
- Improving technology and tools
- Improving processes and methodologies

Handling Complexity: Simplification

- Decomposition of the problem and the solution
- Modularization of the solution
- Separation of concerns of the problem and solution
- Incrementally resolve problems

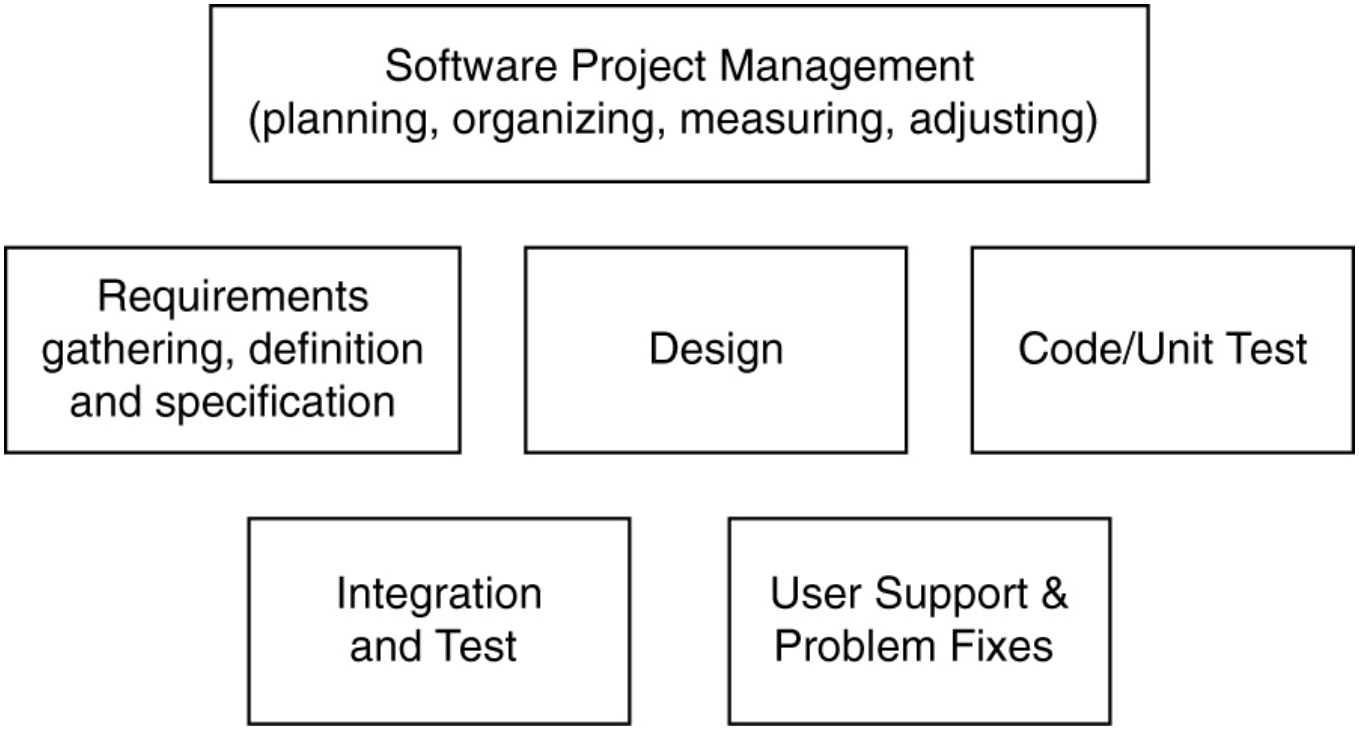
Handling Complexity: Improving Technology and Tools

- Database management systems to handle information and structures of information
- Programming and development platforms
- Computing network
- Multi-developer configuration management
- Modeling techniques of the problem and solution
- Automated testing
- Note: there is a learning curve, so the first time you use these it will initially be more complex

Handling Complexity: Improving Process and Methodologies

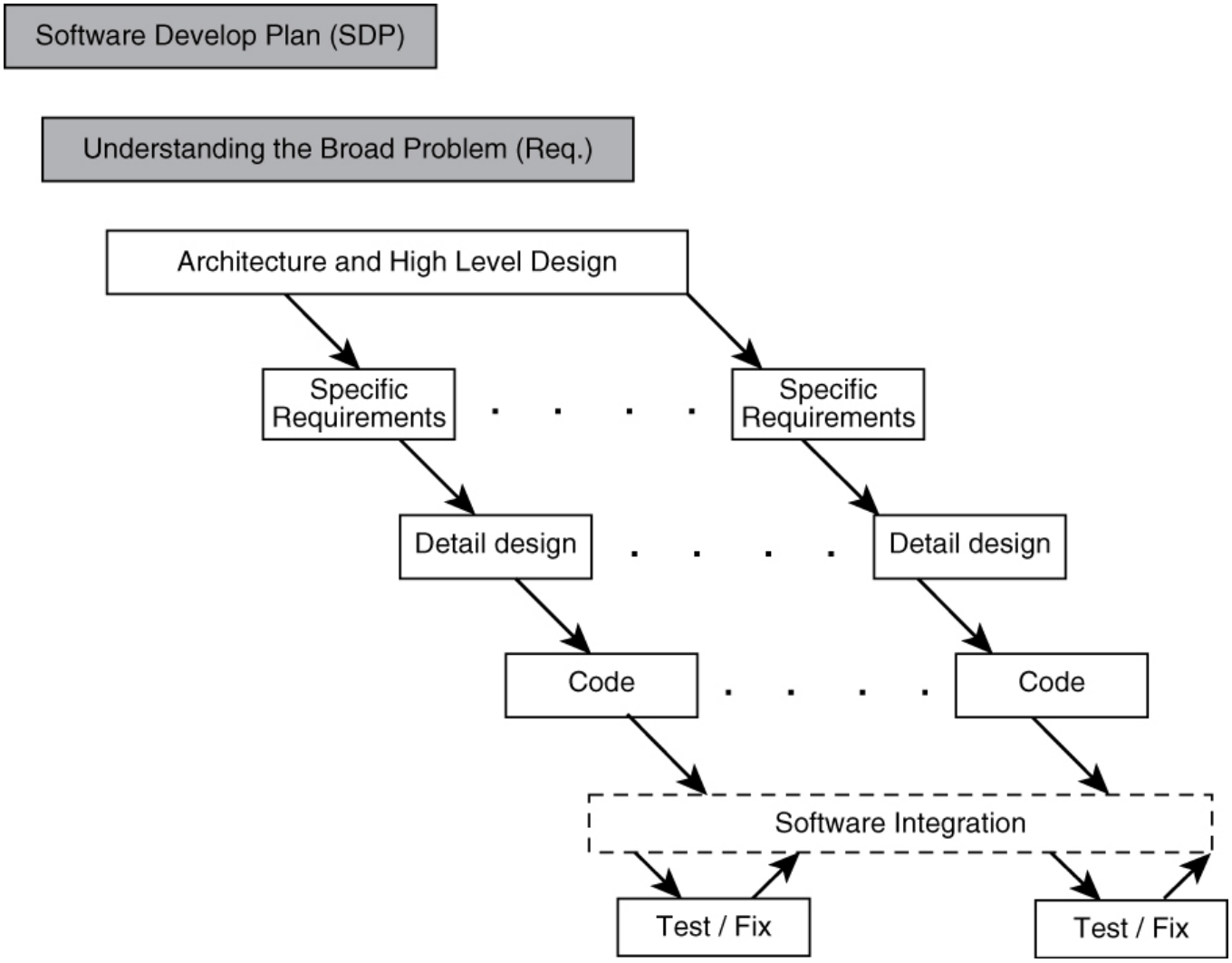
- Coordinate multiple different people performing different tasks
- Guidance for overlapping incremental tasks
- Guidance for measuring separate artifacts and outcomes
- Note: the first time you add a process it will initially be more complex

Task Breakdown Example



- Questions:
  - Who performs what task?
  - How is the task completed and with what technique or tool?
  - When should each task start and end?
  - Who should coordinate the people and the tasks?

Iterative Process Example



“Non-Technical” Considerations

- Effort and Schedule Expansion
  - How does one estimate and handle this?
- Assignment and Communications Expansion
  - More people, more communication issues
  - Do we need some process?
  - Do we need some tools?

Building a Large, Complex System

- Mission-critical systems require
  1. Several separate activities performed by
  2. Multiple people
- Tasks
  - Requirements: gathering, analysis specification, and agreement
  - Design: abstraction, decomposition, cohesion, interaction, and coupling analysis
  - Implementation: coding and unit testing
  - Integration and tracking of parts
  - Separate testing: functional, performance, etc.
  - Packaging and releasing the system

Supporting the System

- Pre-release: prepare for education and support
  - number of expected users
  - number of known problems and expected quality
  - amount of user and support personnel training
  - number of fixes and maintenance cycle
- Post-release: user and customer support
  - Call center and problem resolutions
  - major problem fixes and code changes
  - Functional modifications and enhancements

Coordination

- Because there are
  - more parts,
  - more developers, and
  - more users to consider in large systems
- There is the need for coordination (3 Ps):
  - *Processes* and methodologies to be used
  - Final *product* and intermediate artifacts
  - *People* (developers, support personnel, and users)