



ILLINOIS INSTITUTE
OF TECHNOLOGY

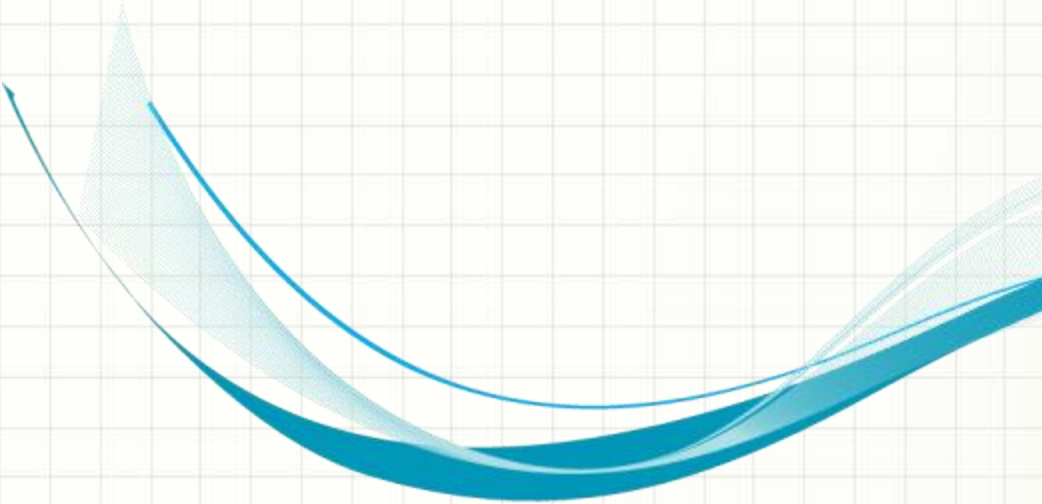
Transforming Lives. Inventing the Future.

www.iit.edu

SOFTWARE ENGINEERING

CS 487

Prof. Dennis Hood
Computer Science



Lecture 2

Software

Processes

Lesson Overview

- Software Processes and Agile Methods
- Reading
 - Ch. 2-3
- Objectives
 - Explore current software engineering approaches
 - Discuss the step-by-step approach to engineering software systems
 - Examine the roles of people and technology in the development and use of software systems
 - Analyze Agile methods and related approaches

Participation – P1

- Identify a real-world “gate keeping” scenario
 - Explain how the gate impedes flow and why
 - Describe design elements which maximize flow
 - Describe the “authority” which enforces compliance
- Describe one benefit and one drawback of allowing agility in the software development process
- Explain the value of “surfacing” in an iterative development process



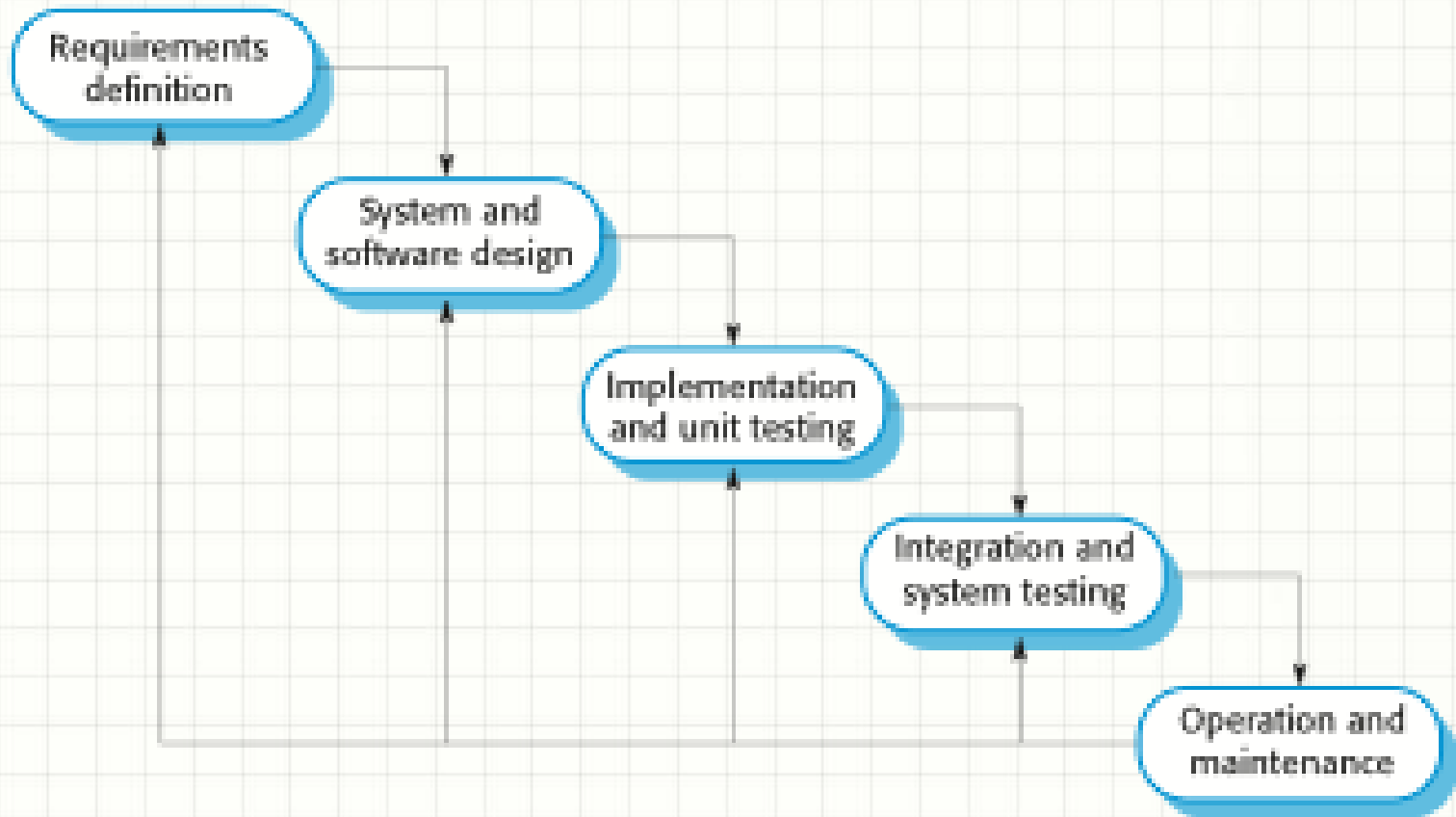
Software Processes

Software Process Models

- Software process
 - A set of activities that leads to the production of a software product
 - e.g., analysis, design, build, verification
- Generic models
 - Process frameworks meant to be tailored
 - Waterfall
 - Incremental Development
 - Reuse-Oriented SW Engineering

Waterfall

- Cascading from one phase to the next
- Each phase ends with a “gate”
 - Signoff is required before a phase can be considered complete
 - Increases the likelihood of true completion
 - Makes it difficult to estimate completion time
- In practice, any engineering effort follows this general approach, but also involves iteration
 - Iterations facilitate learning
 - It is difficult to estimate the required number of iterations



Iterative Development

- Achieve a defined “sub-objective” with each iteration
- The product is evolved methodically
- The focus is on answering questions and resolving challenges
- Often used to bridge the project team-customer gap
 - Produce a prototype and solicit feedback
 - Quickly and cheaply achieve critical understanding

Reuse-Oriented SW Engineering

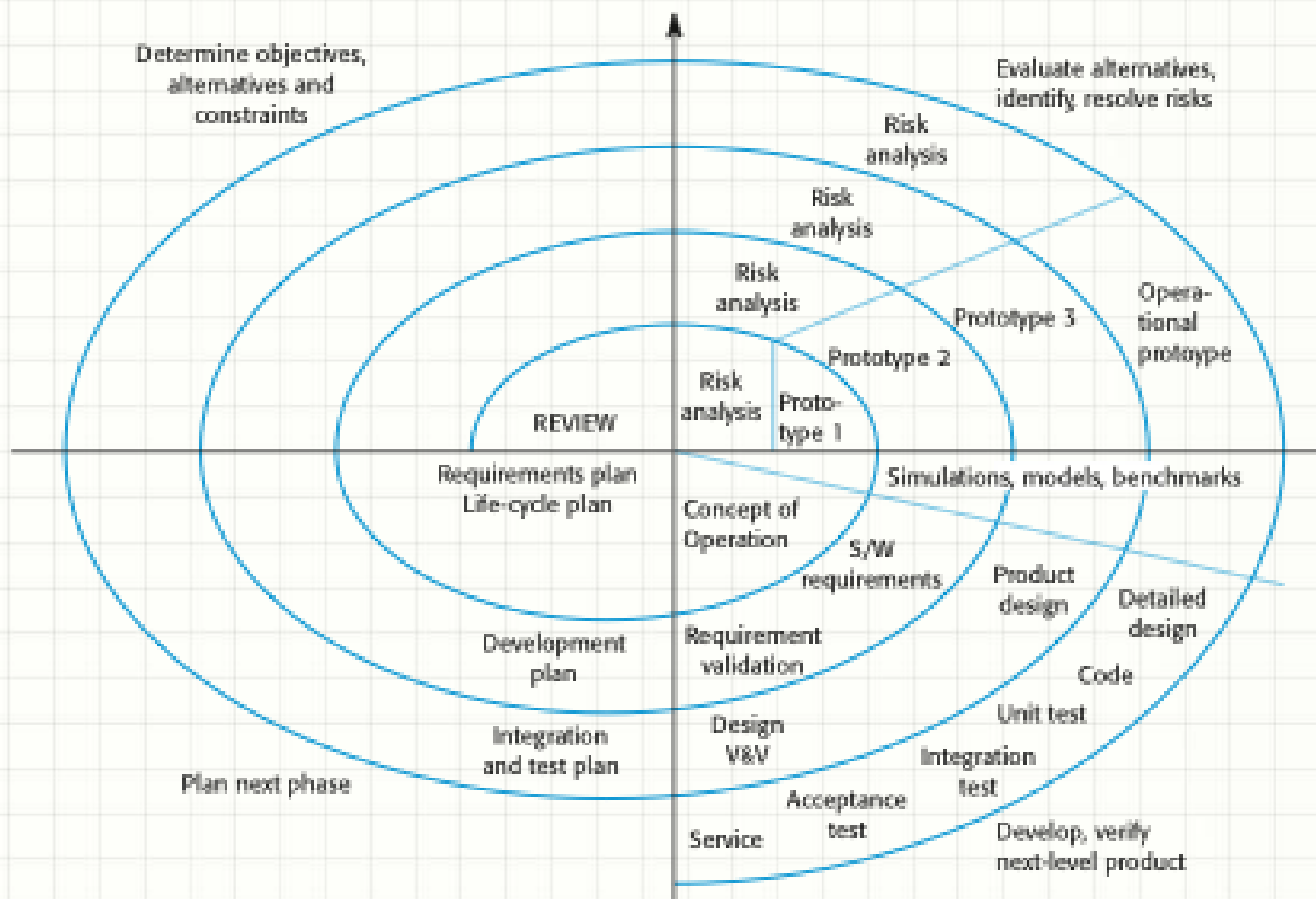
- Focus is on reusing previously created system elements
 - Clearly faster
 - Defects are limited to points of integration and “inappropriately” applied components
- Tradeoffs may be necessary
 - Existing components may not exactly match requirements
- Requires disciplined management of reusable components
 - Documentation
 - Testing
 - Version control, etc.

Development Process Objectives

- Each of these models addresses the following needs:
 - Specification of requirements
 - Identifying services to be provided and environmental constraints
 - Design and implementation
 - Transforming specification into working system
 - Verification and validation
 - Showing that the system satisfies requirements and meets customer expectations
 - Evolution
 - Growing the system over time to accommodate environmental changes and new requirements

The Spiral Model

- Iterative, risk-driven approach developed by Barry Boehm
- Four phases executed repeatedly
 - Objective setting
 - Risk assessment and reduction
 - Development and validation
 - Planning for the next iteration



The Rational Unified Process

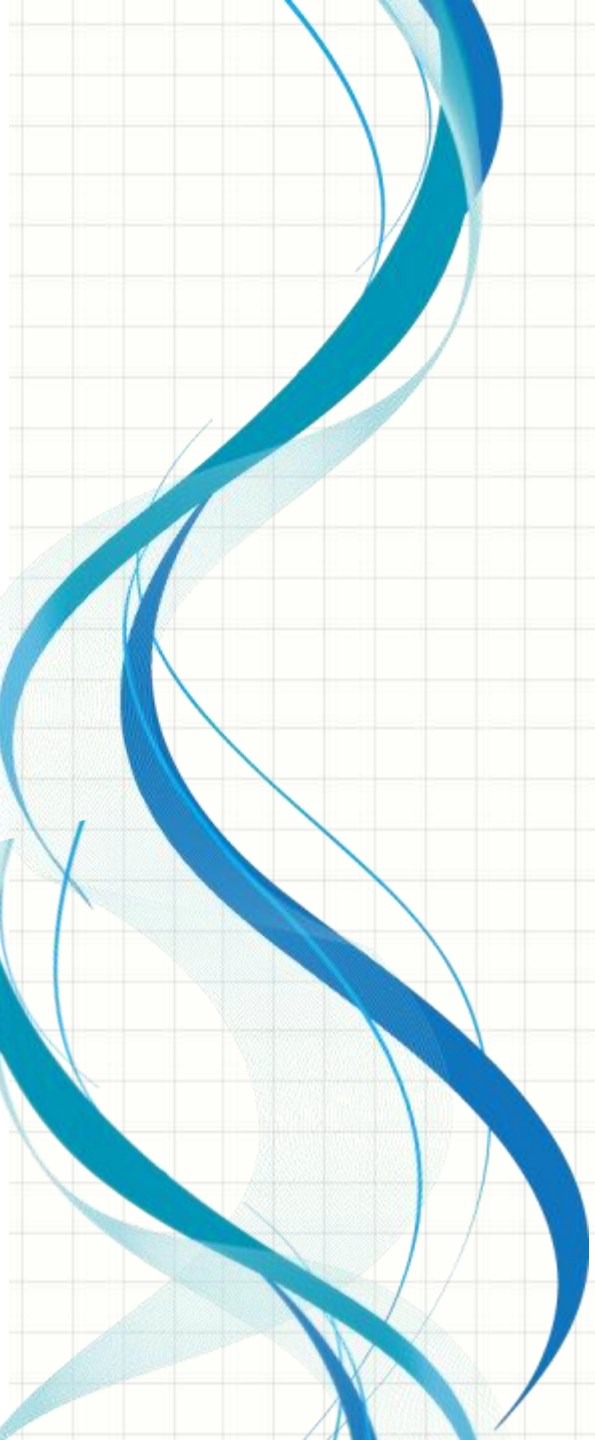
- Hybrid model incorporating prototyping and incremental delivery
 - Based on UML and the Unified Software Development Process
- Described from multiple perspectives
 - Dynamic: phases of the model over time
 - Static: process activities that are enacted
 - Practice: suggests good practices to be used during the process
- RUP phases:
 - Inception, elaboration, construction, and transition

RUP Best Practices

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Visually model software
- Verify software quality
- Control changes to software

Change Happens

- An iterative approach is beneficial because change is likely
 - Technology evolves rapidly
 - Competition drives customers to change requirements
 - Changes to economic conditions can affect strategic approaches
- Minimizing time and scope also minimizes opportunity for change
 - Plan for relatively short, well-defined iterations
 - Customers can utilize incrementally delivered systems



Agile Methods

Agile Methods

- Software development evolving to keep up with the dynamic world it serves
 - Speed up delivery to hit the moving target
 - Involve users significantly and early
 - Extreme flexibility
- Agile characteristics
 - Interleaving of phases
 - Documentation effort is minimized (focus on essentials)
 - Frequent delivery of versions
 - Visual development tools to facilitate rapid, interactive UI development

The Agile Approach

- Flow
 - Get a basic idea, then jump into design and implementation cycles
 - Requirements gathering and V&V are pulled into design and implementation
 - Frequently “surface” the evolving product
 - Growth is more organic than planned
- Pros and Cons
 - The benefit is that progress drives discovery which drives more progress
 - The risk is the discovery is influenced by the working prototype and presence of the customer

Extreme Programming

- Best practices pushed to the extreme
 - Extremely rapid development and deployment
 - Extremely small teams
 - Extremely close user involvement
 - Extremely tight iterations
- Requirements come as simple customer stories (scenarios)
- Change is handled by frequently coding, testing and releasing
- Simplicity is a goal to help manage the rapid pace

XP Testing

- The lack of a system specification makes it difficult to “hand off” testing to an independent verification team
- Focus is on only bringing “clean” code into the current version
 - Develop the tests first, then the code
 - In some ways developing the tests is very similar to specifying functionality, designing the interface, etc.
 - Especially if the user writes the tests
 - Rely on automation

Agile Project Management

- Project management has a generally bureaucratic feel to it
- Agile methods require more flexibility and speed than what is normally associated with project management
- The Scrum approach is an example of PM principles applied to short, tight iterations
 - Short, fixed timeframes (2-4 weeks)
 - Focus on prioritized, outstanding work
 - Frequent communication and assessment
 - Commitment to iterations

Scaling Agile Methods

- Agile methods seem naturally suited for small development efforts
- That would be disappointingly limiting
- Scaling to larger development is possible with:
 - Focus on up-front analysis and design
 - Cross-team communication mechanisms
 - Continuous integration including whole-system builds every time anything is changed/added
 - PMs with experience with Agile methods
 - Organizations with a certain degree of flexibility