CS301 Software Development

Instructor: Peter Kemper

Agile Software Development

Extreme Programming

Test-Driven Development

Software Process

- Most software development projects follow recognized stages from inception to completion
- Useful definitions:

> "A software process is a set of related activities that leads to the production of a software product."
> "A software process model is a simplified representing of a software process."
> — Ian Sommerville, Software Engineering Vol 9.

- Example software process models:
  - Waterfall model
  - Extreme Programming (XP)

Software Process Models

- Waterfall Modell
  - Sequential top-down design, bottom-up implementation process
  - No feedback loops, no iteration
- The only constant in software development is change!
  - Customer requirements, priorities change over time
  - System environment changes over time
  - System hardware platform, operating system changes over time
- Leads to iterative process models

Scrum

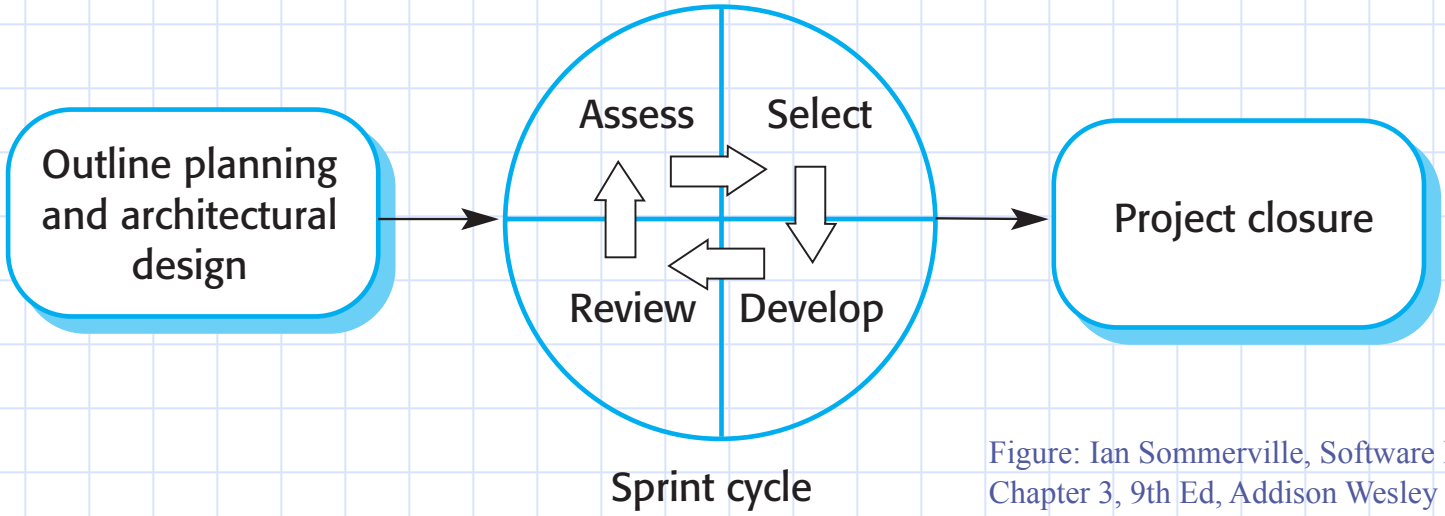
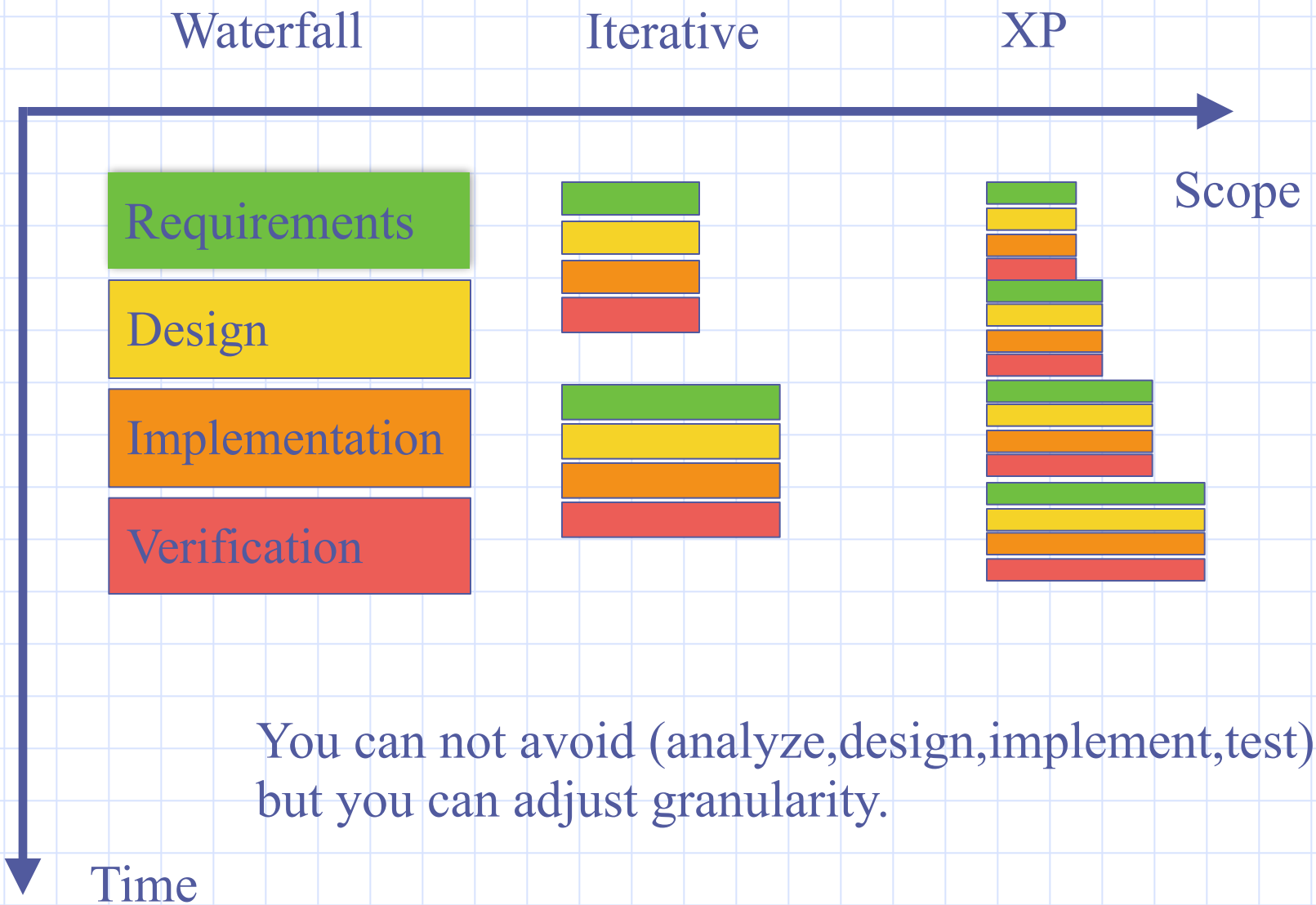


Figure: Ian Sommerville, Software Engineering Chapter 3, 9th Ed, Addison Wesley

- Scrum is a general agile sw development method with a focus on iterative development
- Three phases:
  1. Initial phase: outline planning phase that establishes general objectives and designs the overall software architecture
  2. Series of sprint cycles: each cycle develops an increment of the system
  3. Closure phase: wraps up, completes documentation, user manuals, assesses lessons learnt

Spectrum for Iterative Approaches



You can not avoid (analyze,design,implement,test) but you can adjust granularity.

Agile Software Development

Manifesto for Agile Software Development:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

By Kent Beck, Robert C Martin, Ward Cunningham, Martin Fowler, ... and many more

Reference: Robert C Martin, Agile Software Development, Principles, Patterns and Practices, Prentice Hall 2003.

Goals of Extreme Programming (XP)

- Minimize unnecessary work
  - Very little upfront planning and documentation
  - Keep design simple
- Emphasize communication and feedback
  - Keep customer in the loop
  - Collective ownership of code across team
- Recognize customer priorities
  - Developers work on most important aspect
- Embrace change
  - Make system flexible, ready to meet any change in requirements

XP Process: a Series of Short Cycles (2 weeks each)

- Meet with client to elicit requirements
  - User stories + acceptance tests
- Planning game
  - Translate user-centered stories into developer-centered tasks
  - Estimate costs for tasks (hours of work)
  - Client prioritizes stories
- Implementation follows Test-driven Design (TDD)
  - Write developer tests first
  - Develop simplest possible design to pass the test
  - Code in pairs
  - Refactor code to retain quality
- Evaluate progress and iterate
  - Includes customer feedback, time estimates for tasks, progress

XP: Planning Game

- User stories, user acceptance tests
  - Title and short customer-centered description of a user task
  - Customer must be able to test if implementation of story is ok
- Planning game
  - Customer priorities stories for next cycle
  - Break stories into smaller developer-centered tasks as necessary
  - Team members (sub teams) bid on tasks,
    - give completion date
    - until time budget is filled
  - Customer may reprioritize stories
- Simplicity
  - Just-in-time design, no premature optimization

What means Test Driven Development or Test First Design?

Production code is only written to make failing tests pass.

⇒ Write test first

⇒ Write production code that makes test pass

Think of the implications of this rather radical strategy!

TDD in the Context of Extreme Programming

- Customer is team member
- User stories give requirements
- Acceptance tests give details to user stories
- Planning game between customer and developer
- Short cycles on iteration plan (2 weeks),
  6 iterations to a release plan
- Pair programming with dynamic pairing & frequent change of roles
- Collective ownership
- Test driven development
- Continuous integration
- Simple design, simplest way possible, 1 class - 1 responsibility
- Refactoring to retain quality of code base, no duplicates!
- Sustainable pace

Test Driven Development

1. Check out sources & tests from code base
2. Create test suite / enlarge test suite for product feature, such that current version fails test
3. Implement functionality and make software pass the test
4. Refactor software for clarity of design, quality of coding
5. Make sure that software passes all new and existing tests
6. Check-in software & tests into code base

XP and TDD

- Design is
  - Testable (also for acceptance tests)
  - Maintainable (permanent need for changes + refactoring)
- Code base is always
  - Executable
  - Tested (since it passes all implemented tests)
  - Delivering a growing, never shrinking set of functionality
  - Documented and well coded (refactoring)
- Overall product
  - Grows incrementally with small granularity and short time lines
  - Close interaction and feed back from users
    - Requirements may change over and over again
  - With clear requirements for short term plan
    - User stories for one iteration (2 weeks)
  - With coarser requirements for mid term, long term plan
    - Agreement on overall functionality on an abstract level
  - Is available during the whole development process, surely with limited functionality, but user selects!

XP Summary

- XP is an incremental software process designed to cope with change
- With XP you never miss a deadline; you just deliver less content

Agile SW Development - Promising for CS 301

- Light weight on documentation and design
- Focus
  - On software development and code generation
  - On high quality code
    - Readable, maintainable, self-documenting
    - Simple to change
    - Testable and tested
- Requirements
  - Knowledge on OO Design
  - Knowledge on common solutions for common problems
  - Tool support for
    - Developing test suites, automated testing
    - Refactoring
    - Teamwork