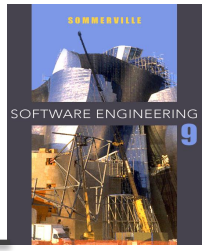
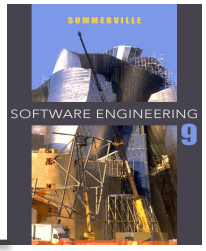
CSE 470 : Software Engineering



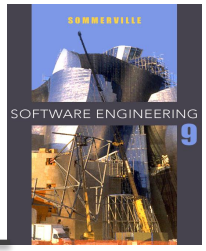
Agile Software Development

Topics



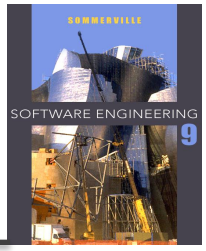
- Agile Software Development
- Agile manifesto
- Agile Characteristics
- Existing Agile Methods
  - Extreme Programming
  - Agile Unified Process
  - Scrum

Agile Software Development (cont.)

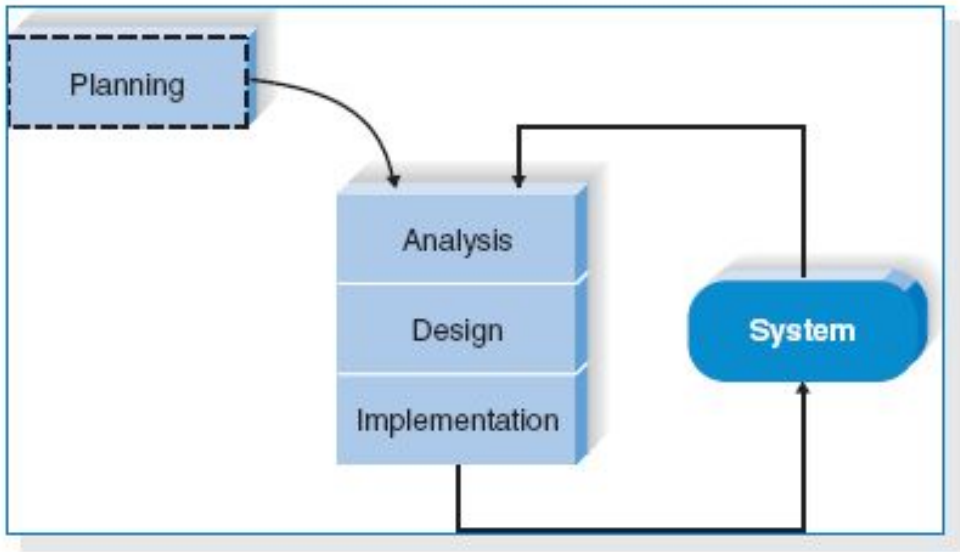


- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - Focus on the **code** rather than the design
  - Are based on an **iterative approach** to software development
  - Are intended to **deliver working software quickly** and evolve quickly to meet changing requirements.
- The aim of agile methods is to **reduce overheads** in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

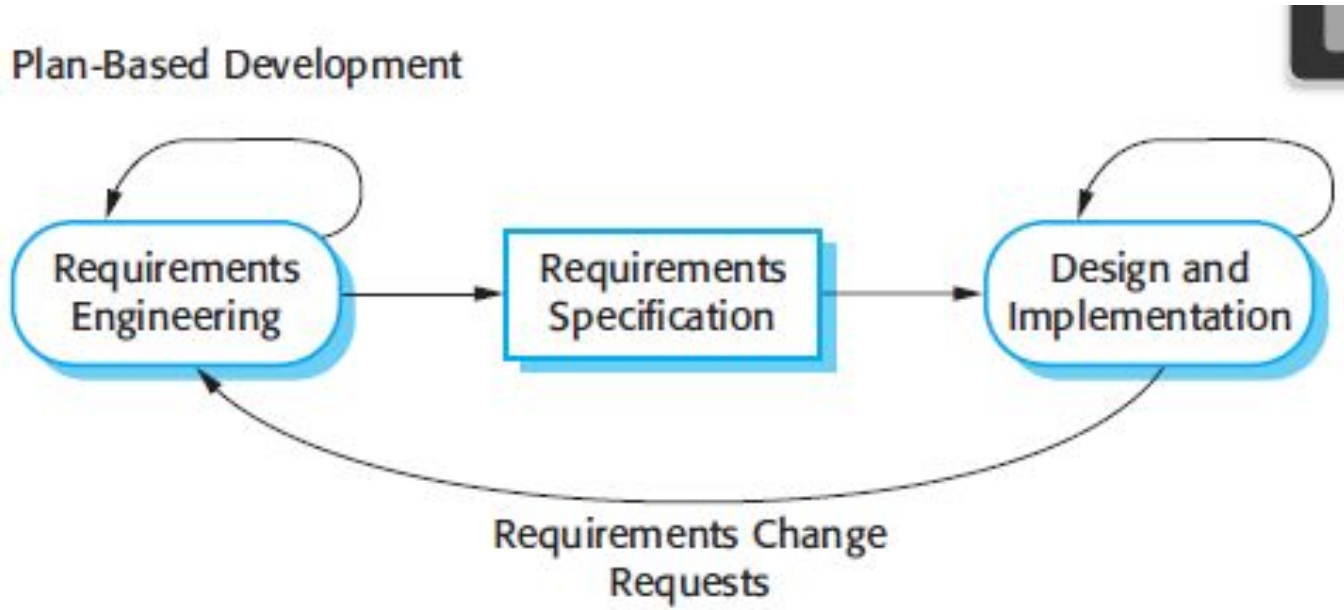
Agile Software Development (cont.)

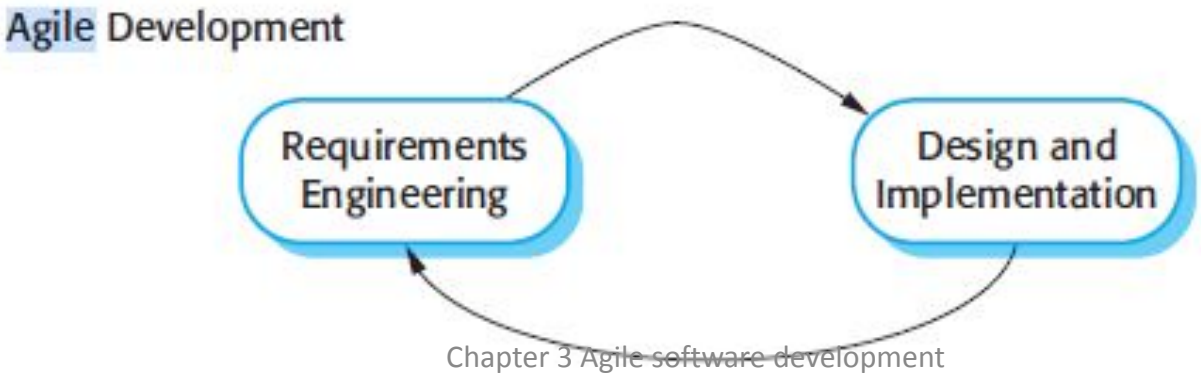


- **Agile software development** is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.
- Software developed during one unit of time is referred to as an iteration, which may last from one to four weeks.
- Agile methods also emphasize working software as the primary measure of progress

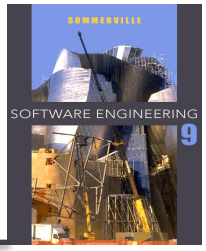


Plan-driven and agile specification



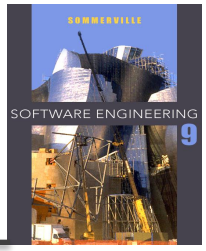


Agile Software Development (cont.)



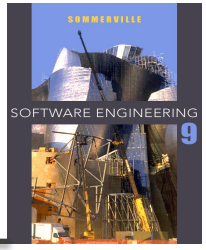
- Characteristics of Agile Software Development
  - -- Light Weighted methodology
  - -- Small to medium sized teams
  - -- vague and/or changing requirements
  - -- vague and/or changing techniques
  - -- Simple design
  - -- Minimal system into production

Agile manifesto



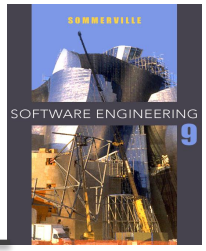
- *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
  - ***Individuals and interactions*** *over processes and tools*
    Working software *over comprehensive documentation*
    Customer collaboration *over contract negotiation*
    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.*

Agile Characteristics



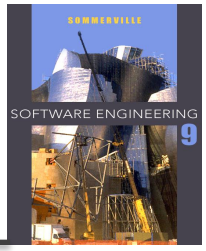
- Modularity
- Iterative
- Time-bound
- Incremental
- Convergent
- People-oriented
- Collaborative

Existing Agile Methods



- Extreme Programming (“XP”)
- Agile Unified Process
- Scrum

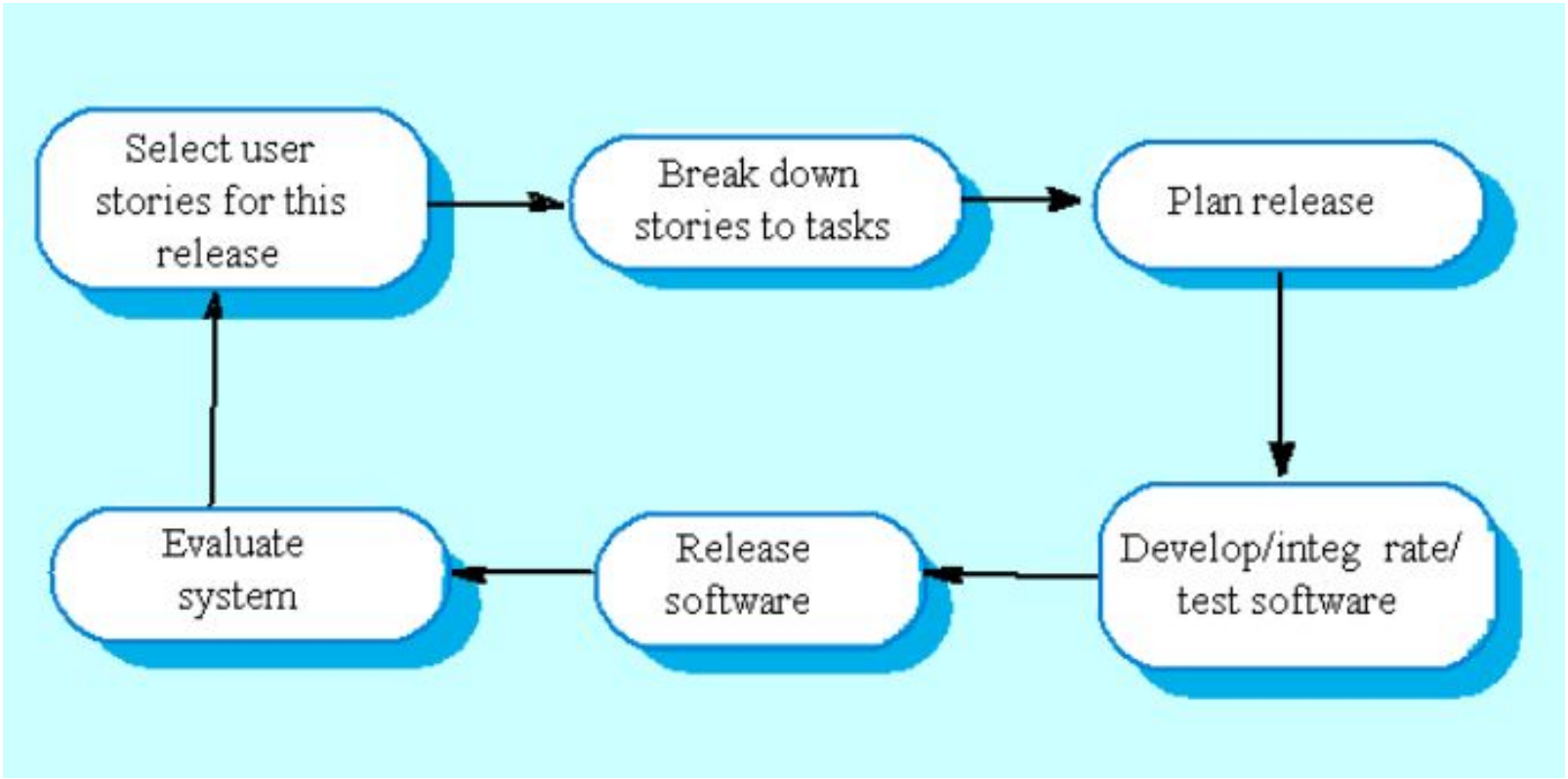
Extreme Programming



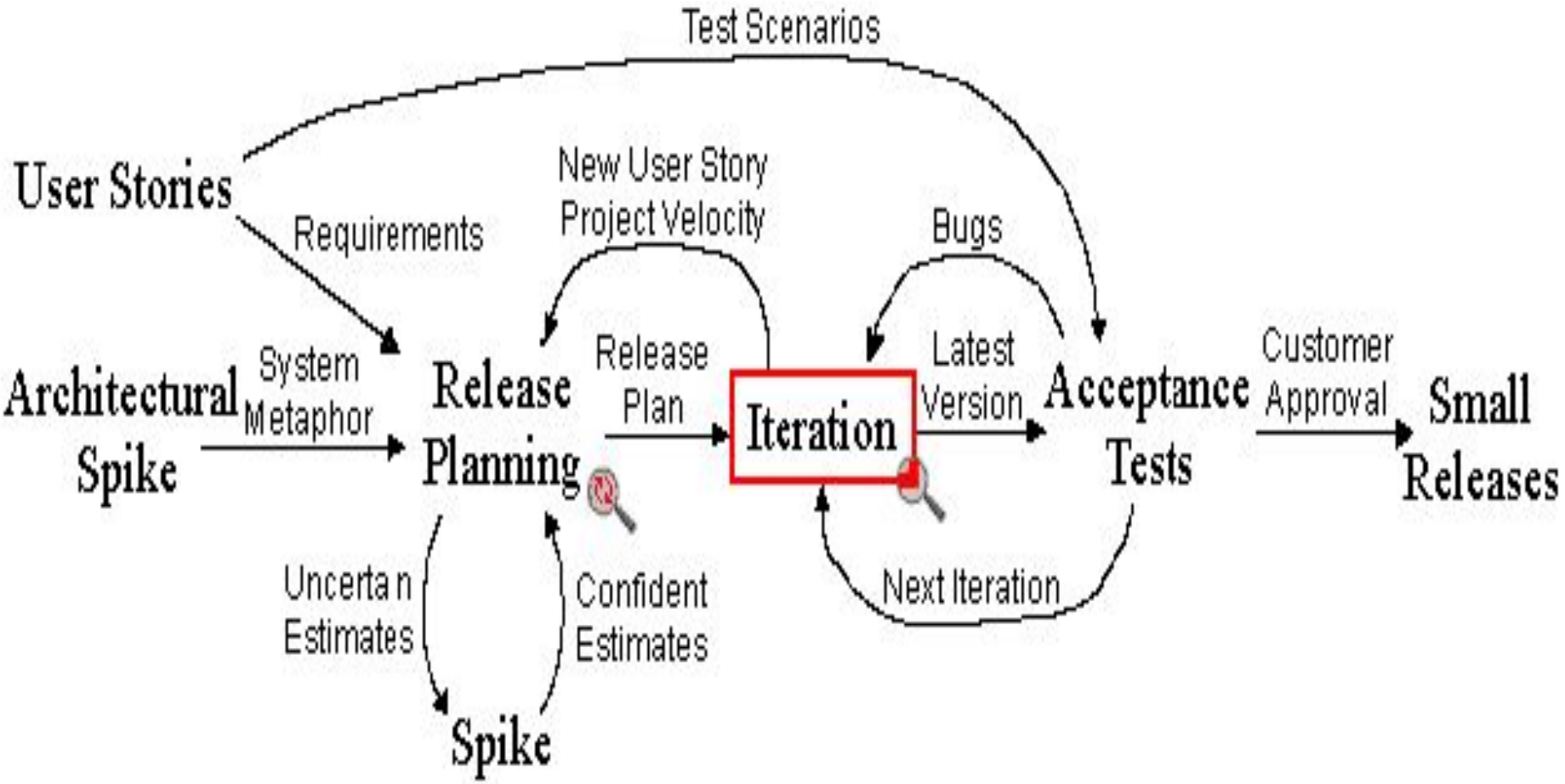
- Perhaps the best-known and most widely used agile method.

- Extreme Programming (XP) takes an 'extreme' approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

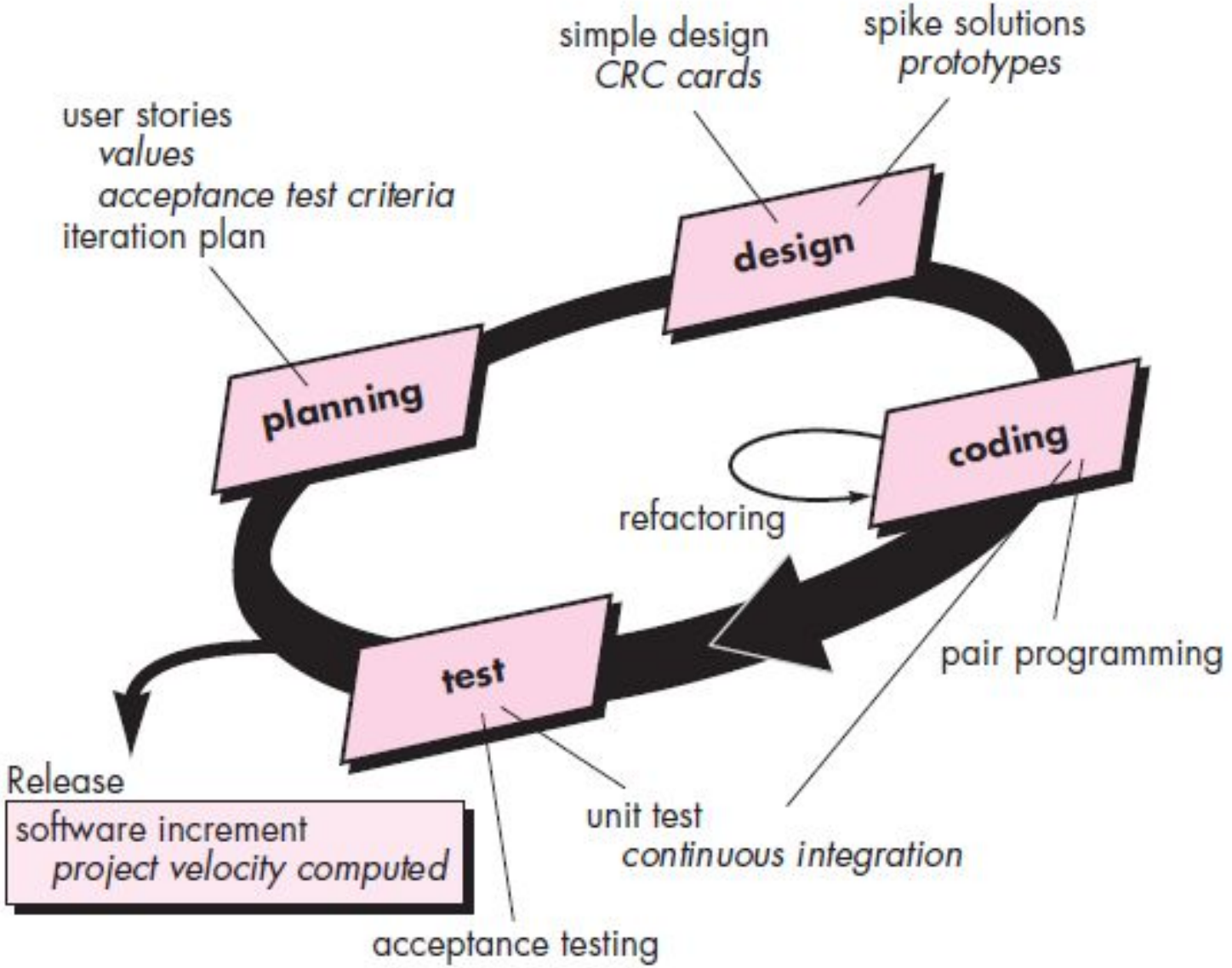
Extreme Programming



Extreme Programming Project

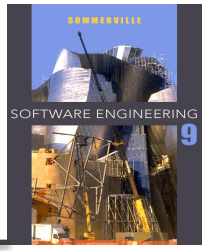


Extreme Programming Project



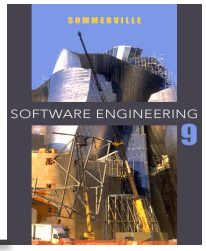
CRC - class-responsibility collaborator

XP Principle or Practice



Incremental planning	Refactoring	
Small releases	Pair programming	Sustainable pace
Simple design	Collective ownership	On-site customer
Test-first development	Continuous integration	

Agile Unified Process

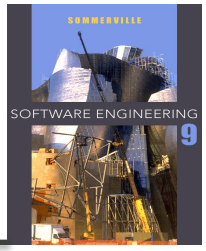


- Agile Unified Process (AUP) is a simplified version of the Rational Unified Process (RUP).

Phases of AUP

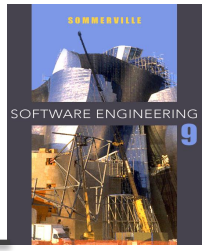
- Inception (scope of the project)
- Elaboration (basic architecture design)
- Construction (implementation)
- Transition (final product release)

Disciplines of AUP



- Business Modelling (requirements and design)
- Implementation
- Test
- Deployment
- Configuration and Change Management
- Project Management
- Environment

Scrum

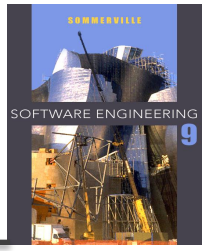


- It is an Agile S/W development method for project management

Characteristics:

- Prioritized work is done
- Completion of backlog items
- Progress is explained
- Agile Software Development

Scrum Framework



Roles

- Product owner
- Scrum Master
- Team

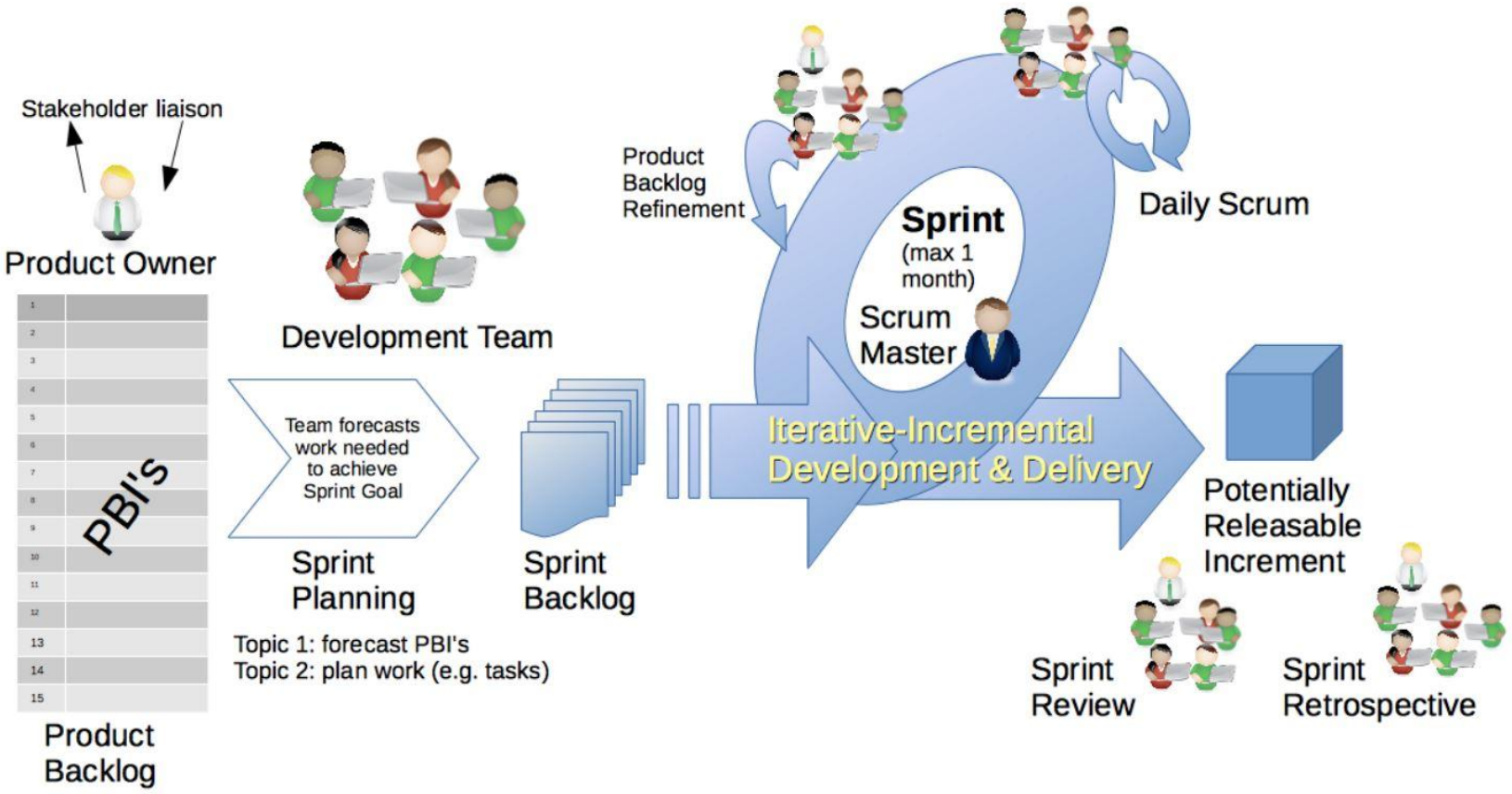
Ceremonies

- Sprint planning
- Sprint review and Sprint retrospective
- Daily scrum meeting

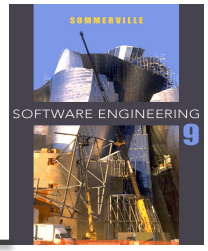
Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

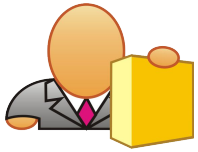
Scrum Framework (cont.)



Scrum Roles



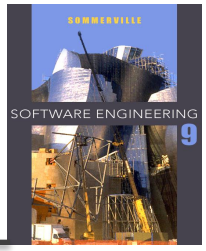
- Product Owner
  - Possibly a Product Manager or Project Sponsor
  - Decides features, release date, prioritization, $$$



- Scrum Master
  - Typically a Project Manager or Team Leader
  - Responsible for enacting Scrum values and practices
  - Remove impediments / politics, keeps everyone productive

- Project Team
  - 5-10 members; Teams are self-organizing
  - Cross-functional: QA, Programmers, UI Designers, etc.
  - Membership should change only between sprints

Sprint Planning Mtg.



Team capacity

Product backlog

Business conditions

Current product

Technology

Sprint planning meeting

Sprint prioritization

- Analyze/evaluate product backlog
- Select sprint goal

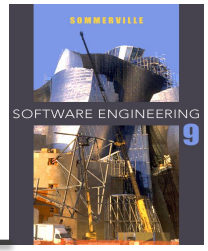
Sprint planning

- Decide how to achieve sprint goal (design)
- Create sprint backlog (tasks) from product backlog items (user stories / features)
- Estimate sprint backlog in hours

Sprint goal

Sprint backlog

Daily Scrum Meeting



- Parameters

  Daily, ~15 minutes, Stand-up

  Anyone late pays a $1 fee

- Not for problem solving

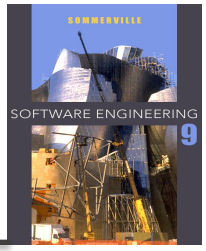
  Whole world is invited

  Only team members, Scrum Master, product owner, can talk

  Helps avoid other unnecessary meetings

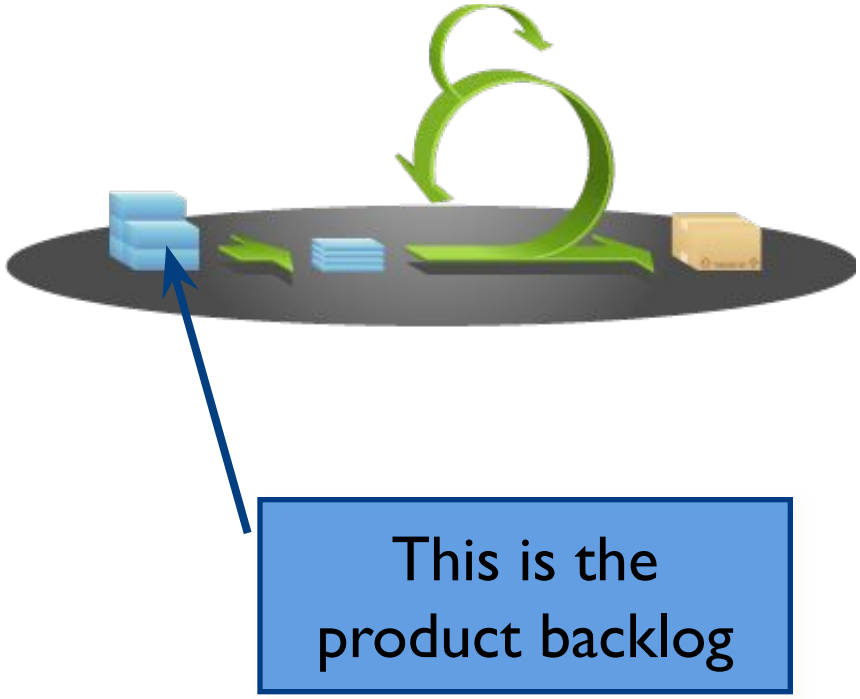
- Three questions answered by each team member:
  1. What did you do yesterday?
  2. What will you do today?
  3. What obstacles are in your way?

Scrum's Artifacts



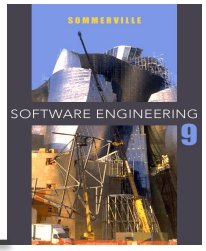
- Scrum has remarkably few artifacts
  - Product Backlog
  - Sprint Backlog
  - Burndown Charts
- Can be managed using just an Excel spreadsheet
  - More advanced / complicated tools exist:
    - Expensive
    - Web-based – no good for Scrum Master/project manager who travels
    - Still under development

Product Backlog



- The requirements
- A list of all desired work on project
- Ideally expressed as a list of user stories along with "story points", such that each item has value to users or customers of the product
- Prioritized by the product owner
- Reprioritized at start of each sprint

User Stories



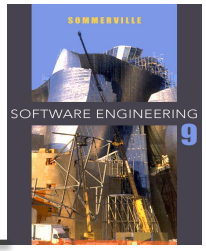
- Instead of Use Cases, Agile project owners do "user stories"
  - **Who** (user role) – Is this a customer, employee, admin, etc.?
  - **What** (goal) – What functionality must be achieved/developed?
  - **Why** (reason) – Why does user want to accomplish this goal?

As a [user role], I want to [goal], so I can [reason].

- Example:
  - "As a user, I want to log in, so I can access subscriber content."

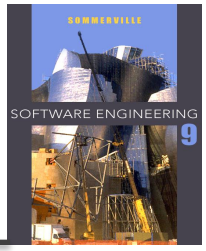
- **story points**: Rating of effort needed to implement this story
  - common scales: 1-10, shirt sizes (XS, S, M, L, XL), etc.

Sample Product Backlog



Backlog item	Estimate
Allow a guest to make a reservation	3 (story points)
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
…	30
…	50

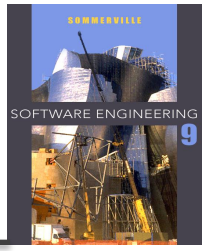
Sprint Backlog



- Individuals sign up for work of their own choosing
  - Work is never assigned
- Estimated work remaining is updated daily

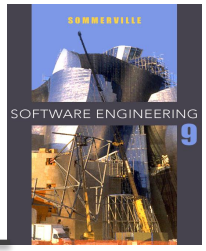
- Any team member can add, delete change sprint backlog
- Work for the sprint emerges
- If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
- Update work remaining as more becomes known

Sample Sprint backlog



Tasks	Mon	Tue	Wed	Thu	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	4	
Test the middle tier	8	16	16	11	8
Write online help	12				
Write the Foo class	8	8	8	8	8
Add error logging			8	4	

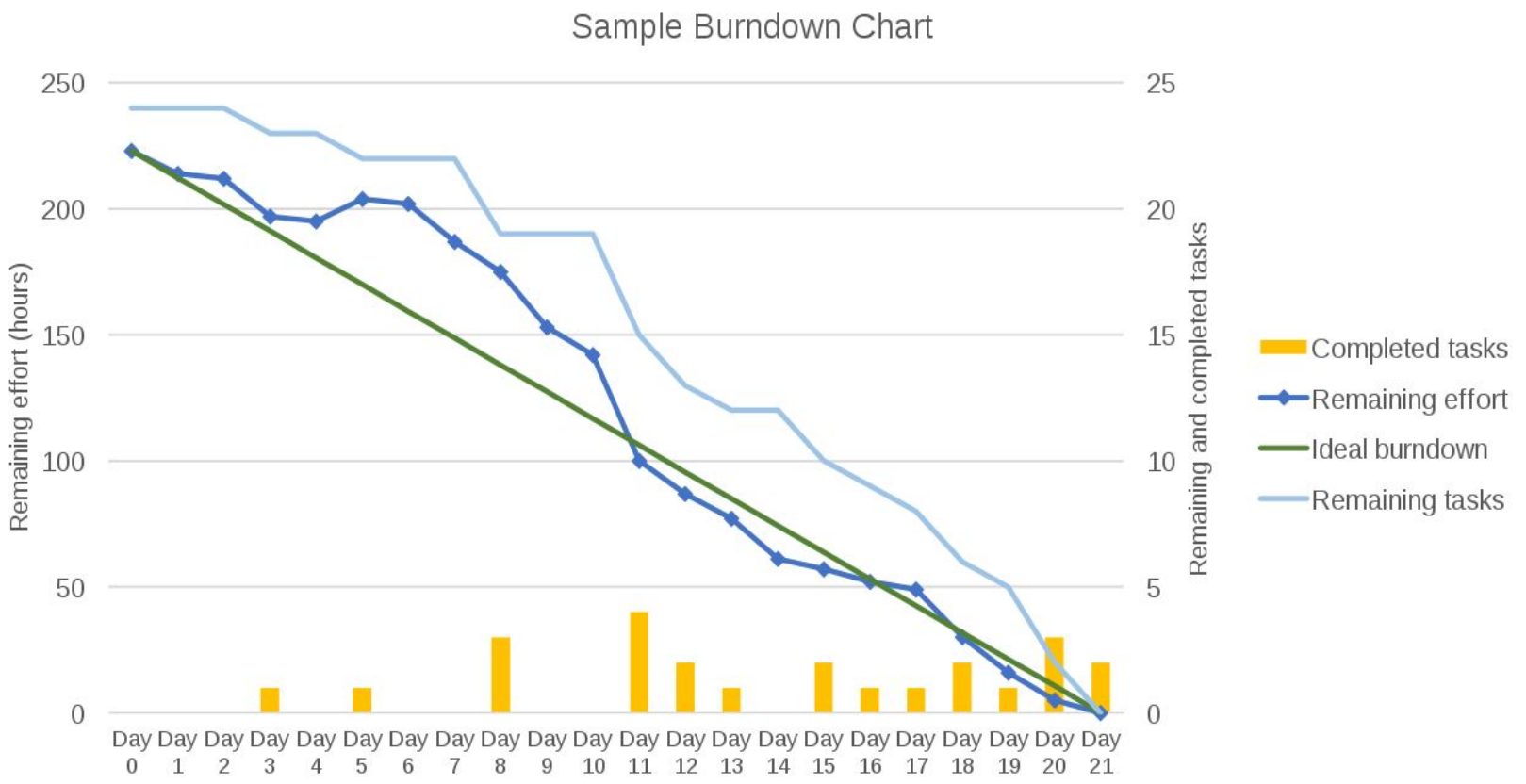
Sprint Burndown Chart



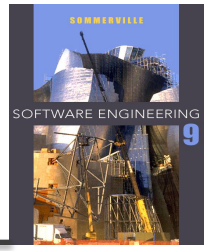
- A display of what work has been completed and what is left to complete
  - one for each developer or work item
  - updated every day
  - (make best guess about hours/points completed each day)

- *variation:* Release burndown chart
  - shows overall progress
  - updated at end of each sprint

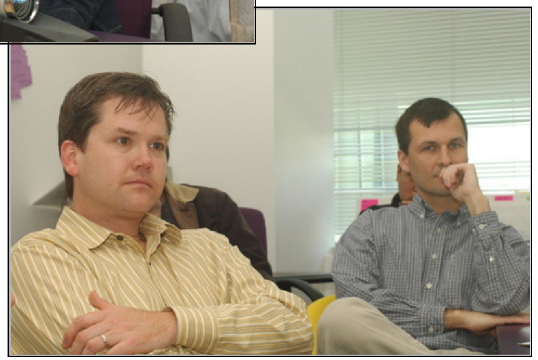
Sprint Burndown Chart



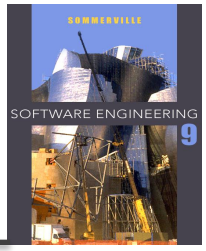
The Sprint Review



- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world



Credits, References



- Mike Cohn, Mountain Goat Software
  www.mountaingoatsoftware.com
- *Scrum and The Enterprise* by Ken Schwaber
- *Succeeding with Agile* by Mike Cohn
- *Agile Software Development Ecosystems* by Jim Highsmith
- *Agile Software Development with Scrum* by K. Schwaber and M. Beedle
- *User Stories Applied for Agile Software Development* by Mike Cohn

- www.agilescrum.com/
- www.objectmentor.com
- jeffsutherland.com/
- www.controlchaos.com/scrumwp.htm
- agilealliance.com/articles/articles/InventingScrum.pdf