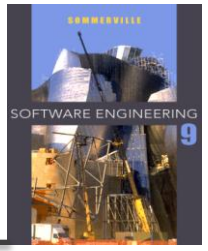
Chapter 3 – Agile Software Development

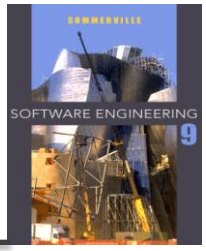
Chapter 3 – Agile Software Development

Topics covered



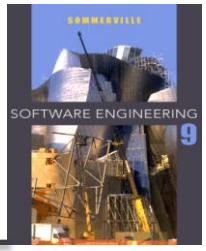
- ✧ Intro
- ✧ 3.1 Agile methods
- ✧ 3.2 Plan-driven and agile development
- ✧ 3.3 Extreme programming
- ✧ 3.4 Agile project management
- ✧ 3.5 Scaling agile methods

Agile software development - Intro



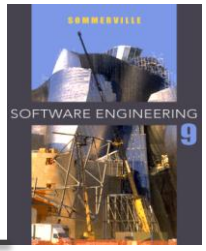
- Rapid development and delivery is now often the most important requirement for software systems
- Plan-driven development slow to adapt to changing requirements (rework)
- 80s IBM introduced incremental development recognizing need for rapid system development
- Late 90s: agile approach (Scrum, XP, ..)

Rapid Development



- Designed to produce useful software quickly
- Specification, design, and implementation interleaved
- System developed in series of increments / version (typically every 2 – 3 weeks)
- No detailed requirement specification, design document, or documentation
- Informal communication

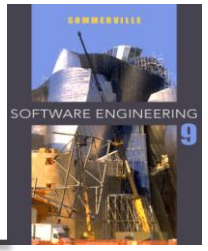
3.1 Agile methods



✧ 80s 90s: widespread view that best way to improve software was controlled and rigorous software development process

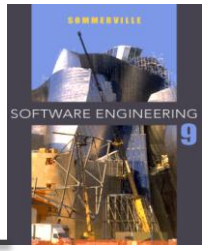
large distributed projects (e.g. safety critical, many rules and regulations) vs small and medium sized business software

Agile manifesto (2001 Snowbird)



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

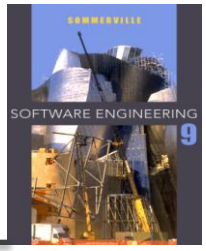
Agile manifesto



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*

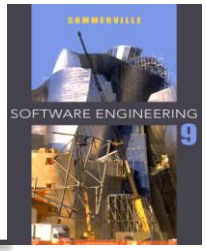
Agile manifesto



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*

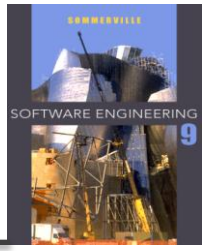
Agile manifesto



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*

Agile manifesto

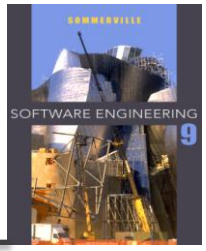


We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and Interactions	over	*Processes and Tools*
Working Software	over	*Comprehensive Documentation*
Customer Collaboration	over	*Contract Negotiation*
Responding to Change	over	*Following a Plan*

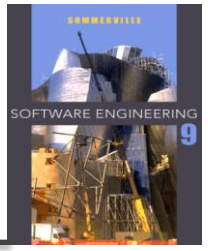
While there is value in the items on the right, we value the items on the left more.

Agile methods have been very successful for …



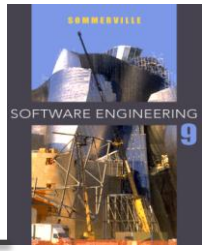
- Development of small or medium-sized product
- Custom system development with clear commitment from customer to become involved
- Few external rules and regulations

The principles of agile methods (Fig 3.1)



✧ All agile methods (XP, Scrum, Crystal, . . .) share principles based on the agile manifesto

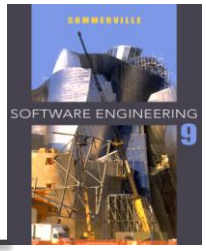
The principles of agile methods



✧ All agile methods (XP, Scrum, Crystal, . . .) share principles based on the agile manifesto

Customer involvement

The principles of agile methods

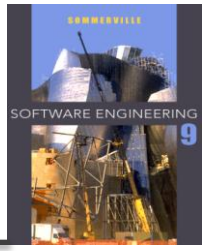


✧ All agile methods (XP, Scrum, Crystal, . . .) share principles based on the agile manifesto

Customer involvement

Incremental delivery

The principles of agile methods



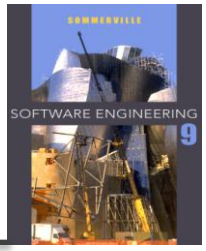
✧ All agile methods (XP, Scrum, Crystal, . . .) share principles based on the agile manifesto

Customer involvement

Incremental delivery

People not process

The principles of agile methods



✧ All agile methods (XP, Scrum, Crystal, . . .) share principles based on the agile manifesto

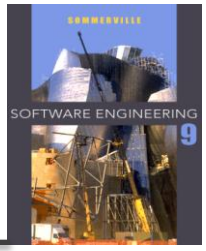
Customer involvement

Incremental delivery

People not process

Maintain simplicity

The principles of agile methods



✧ All agile methods (XP, Scrum, Crystal, . . .) share principles based on the agile manifesto

Customer involvement

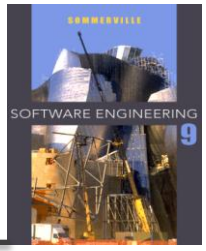
Incremental delivery

People not process

Maintain simplicity

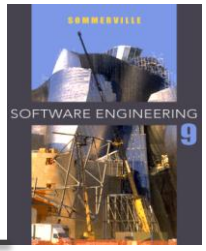
Embrace change

Problems with agile methods



- ✧ Keep customers' interest
- ✧ Team members unsuited for intense involvement
- ✧ Prioritizing changes with multiple stakeholders.
- ✧ Maintaining simplicity (extra work)
- ✧ Contract (based on time)
- ✧ Company culture
- ✧ Doesn't scale well

Agile methods and maintenance / evolution



✧ Two key issues:

- Are systems maintainable?

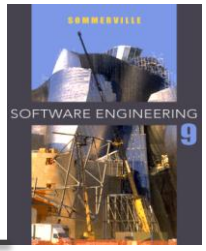
  Problem: system requirements document needed

- Can agile methods be effective for evolving a system?

  Problem: difficult to keep customer involved

✧ What if original development team dissolves?

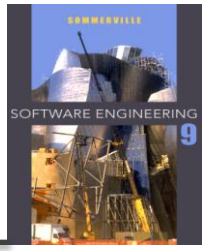
3.2 Plan-driven and agile development



✧ Plan-driven development

- separate development stages
- outputs from one stage used for planning next process activity
- Incremental development is possible (Iteration within activities)
- Formal documents used to communicate between stages

3.2 Plan-driven and agile development



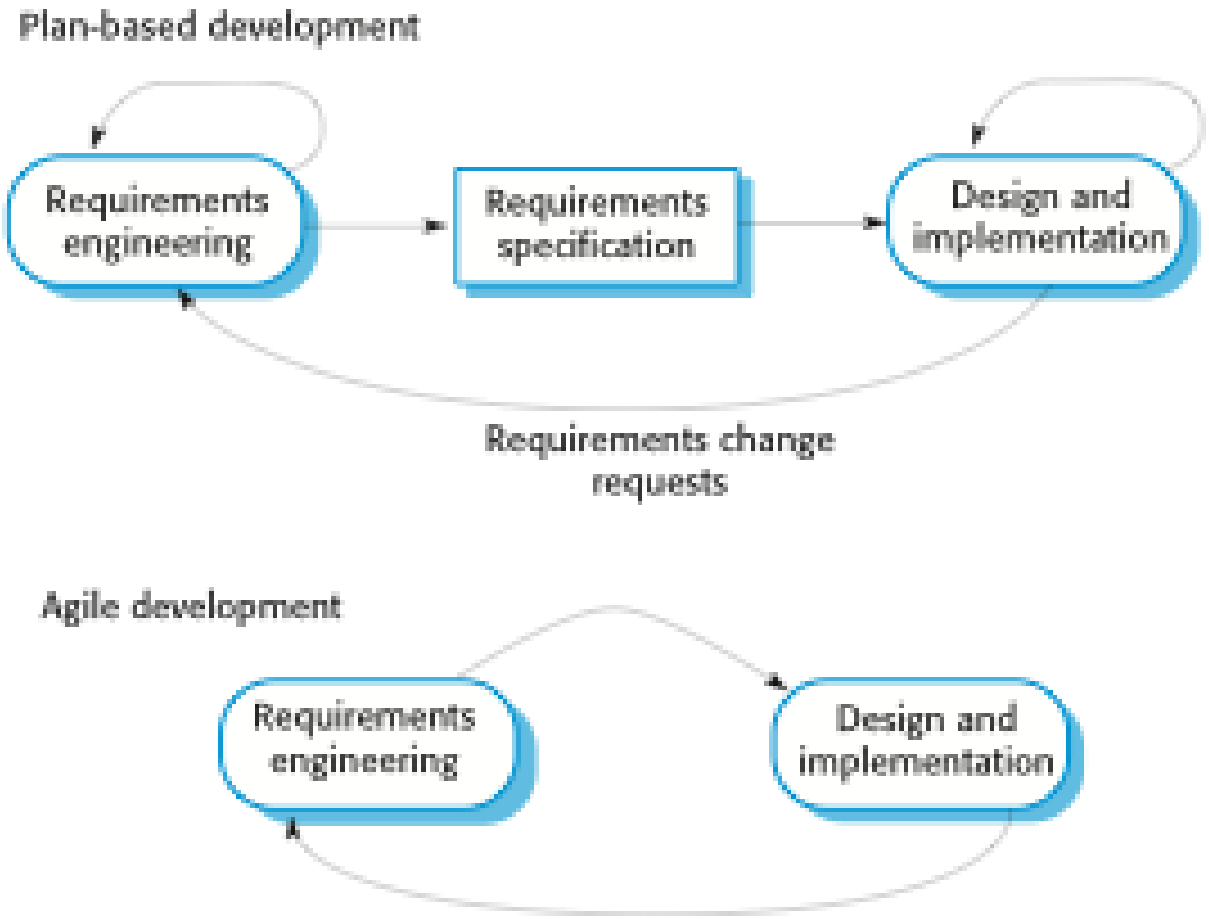
✧ Plan-driven development

- separate development stages
- outputs from one stage used for planning next process activity
- Incremental development is possible (Iteration within activities)
- Formal documents used to communicate between stages

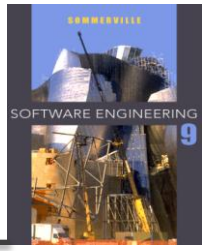
✧ Agile development

- Specification, design, implementation and testing are inter-leaved (iterations across activities)

Plan-driven and agile specification (Fig. 3.2)

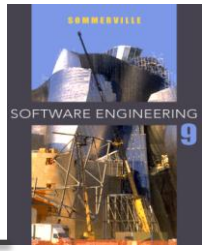


Technical, human, organizational issues



- Most projects include elements of plan-driven and agile processes. Deciding on the balance depends on:
  - How important is detailed specification and design before implementation?
  - Is an incremental delivery strategy realistic?
  - How large is the system that is being developed?
  - What type of system is being developed?
  - What is the expected lifetime of the system?

Technical, human, organizational issues



- What technologies are available to support system development?
- How is the development team organized? (outsourced?)
- Are there cultural / organizational issues?
- How good are the designers and programmers?
- Is the system subject to external regulation?

[Exercise Agile Manifesto]