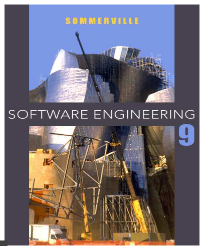
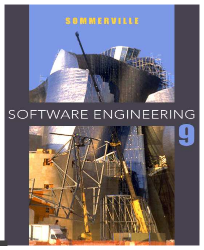


Chapter 2 – Software Processes

2.1 Software Process Models

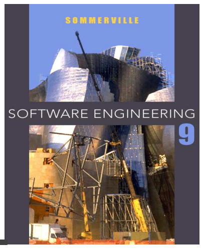
Overview:



✧ 2.1 Software Process Models

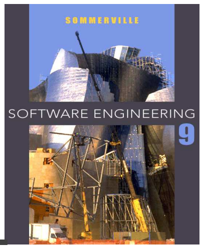
- Waterfall model
- Incremental development
- Reuse-oriented software engineering

The software process



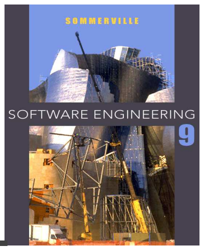
- A **software process** is a structured set of activities required to develop a software system.
- A **software process model** is an abstract representation of a process.
    - particular perspective
    - partial information

The software process



- ✧ A **software process** is a structured set of activities required to develop a software system.
- ✧ A **software process model** is an abstract representation of a process.
  - ▪ Describes process from a particular perspective
  - ▪ Provides only partial information about the process

There is no ‘ideal’ software process

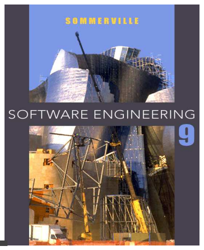


✧ Which process should we choose?

Consider type of **application**:

- Critical system vs business system

There is no 'ideal' software process



✧ Which process should we choose?

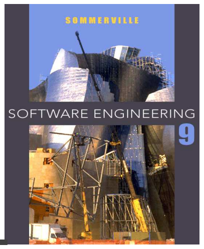
Consider type of **application**:

- Critical system vs business system

Consider **organization**:

- Process standardization

There is no ‘ideal’ software process



✧ Which process should we choose?

Consider type of **application**:

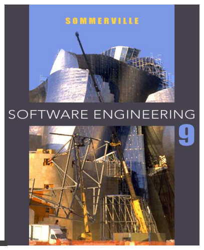
- Critical system vs business system

Consider **organization**:

- Process standardization

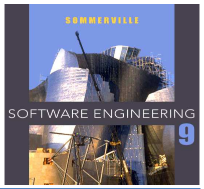
Consider **developer team**:

There is no ‘ideal’ software process

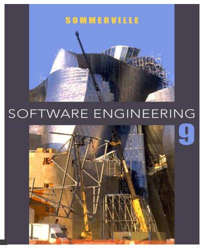


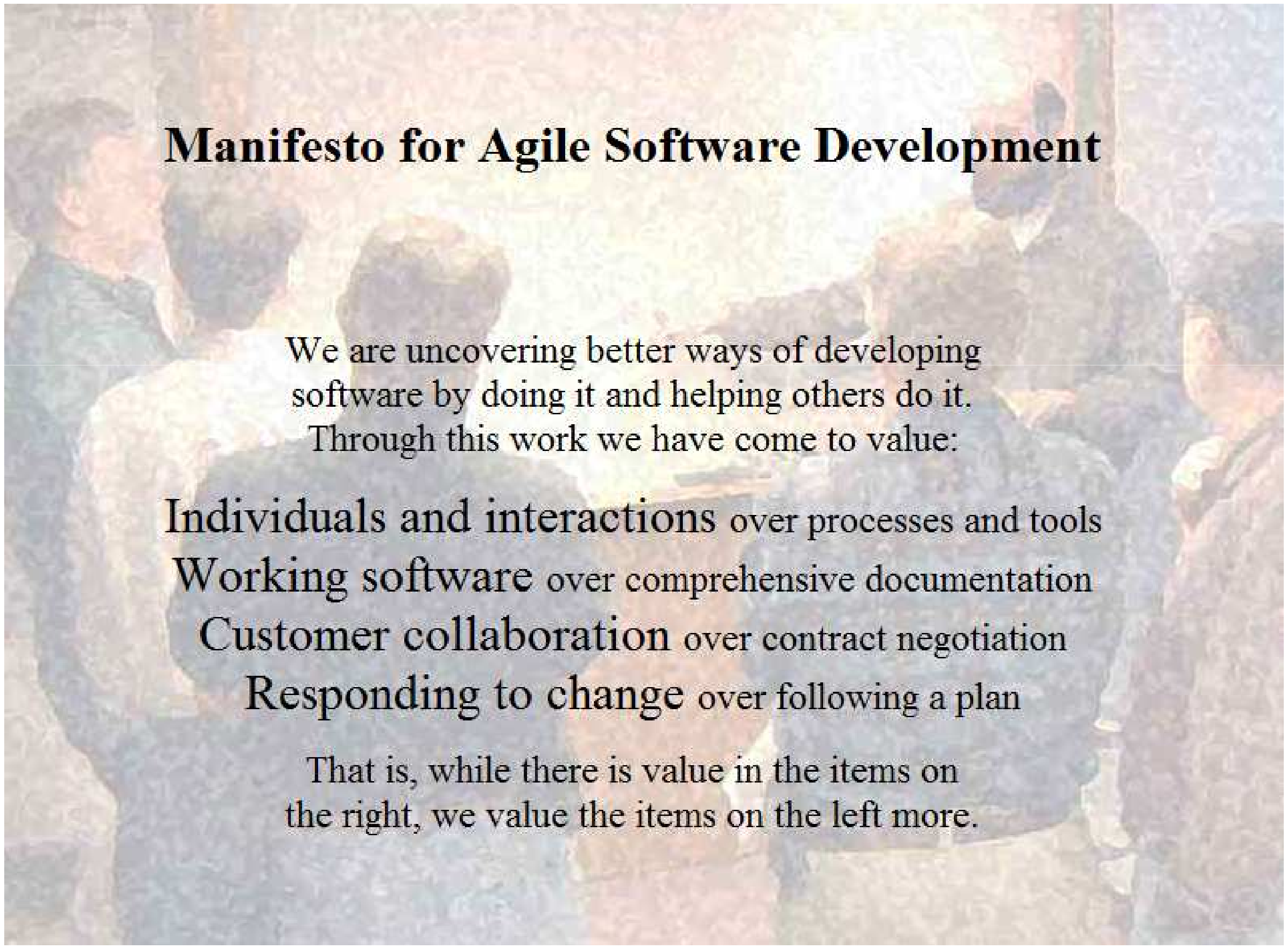
- ✧ We rely on people making decisions and judgments
- ✧ Consider Application:
  - Critical systems -> structured development process
  - Business systems -> more flexible process
- ✧ Consider Organization:
  - Software processes can be improved by process standardization
  - => better communication, reduced training times, …
- ✧ Consider Developer Team:

2 types of processes



Plan-driven process	Agile Process
Process activities are planned in advance	Planning is incremental
Progress is measured against this plan.	Easier to change
Process concludes with the delivery (except maintenance)	Delivery is incremental





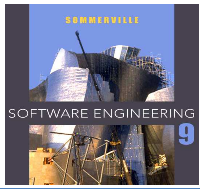
Manifesto for Agile Software Development

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.

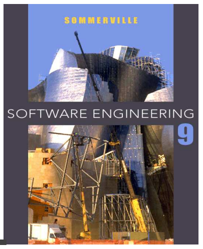
2 types of processes



Plan-driven process	Agile Process
Process activities are planned in advance	Planning is incremental
Progress is measured against this plan.	Easier to change
Process concludes with the delivery (except maintenance)	Delivery is incremental

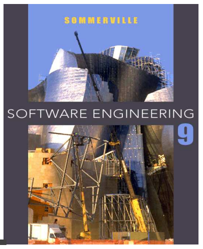
In practice, most practical processes include elements of **both** plan-driven and agile approaches.

2.1 Software process models



✧ 3 very general process models ("Process Paradigms")

2.1 Software process models

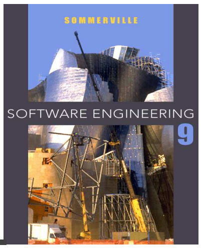


✧ 3 very general process models ("Process Paradigms")

a) The waterfall model

- Plan-driven and document-driven

2.1 Software process models

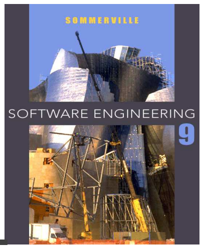


✧ 3 very general process models ("Process Paradigms")

a) The waterfall model
   - Plan-driven and document-driven

b) Incremental development
   - Specification, development, validation interleaved
   - Plan-driven or agile

2.1 Software process models

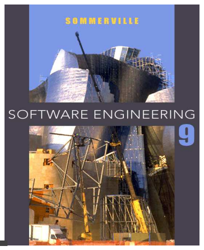


✧ 3 very general process models ("Process Paradigms")

a) The waterfall model
   - Plan-driven and document-driven

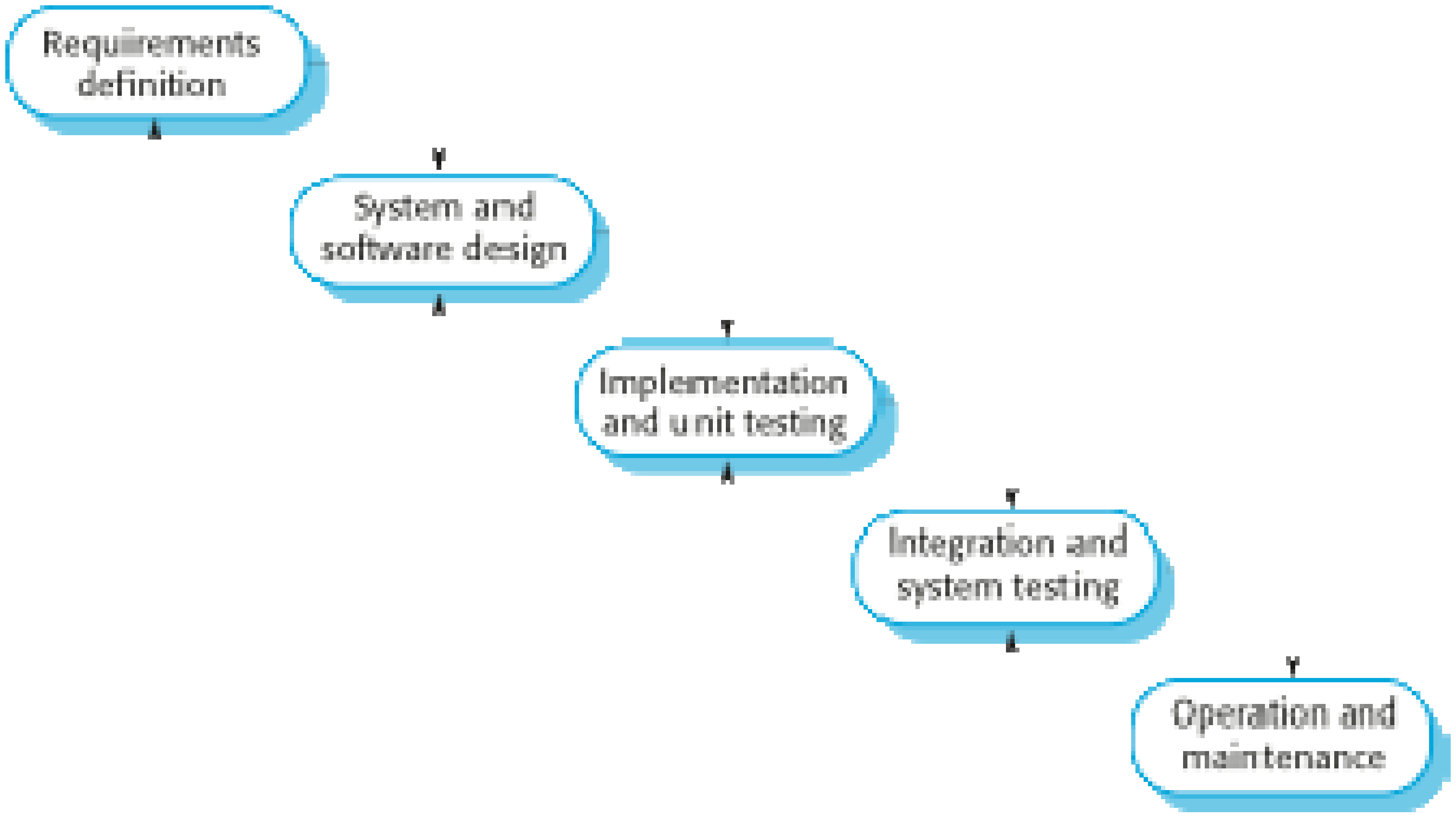
b) Incremental development
   - Specification, development, validation interleaved
   - Plan-driven or agile

c) Reuse-oriented software engineering
   - plan-driven or agile.

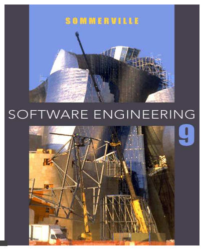


- ✧ Process paradigms described not mutual exclusive.
- ✧ Often used together – especially in large systems

2.1.1 waterfall model

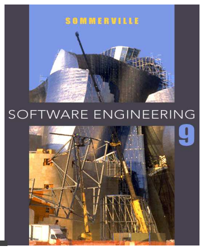

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Waterfall model



Complete a phase > produce document(s) > next phase

Waterfall model

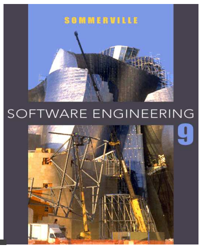


Complete a phase > produce document(s) > next phase

✧ Pro:

- Managers can monitor process
- Structure and documents help coordinate work

TODO:

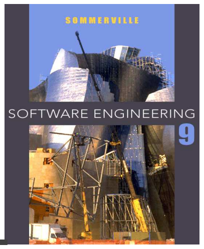


Complete a phase > produce document(s) > next phase

- Pro:

- Con:

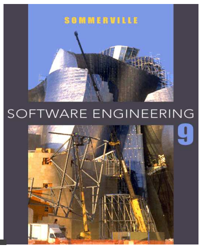
Waterfall model



Complete a phase > produce document(s) > next phase

- Pro:
  - Managers can monitor process
  - Structure and documents help coordinate work
- Con:
  - Inflexible
  - High cost of change

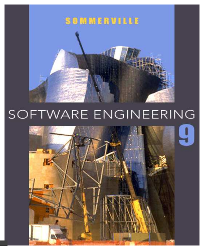
Waterfall model



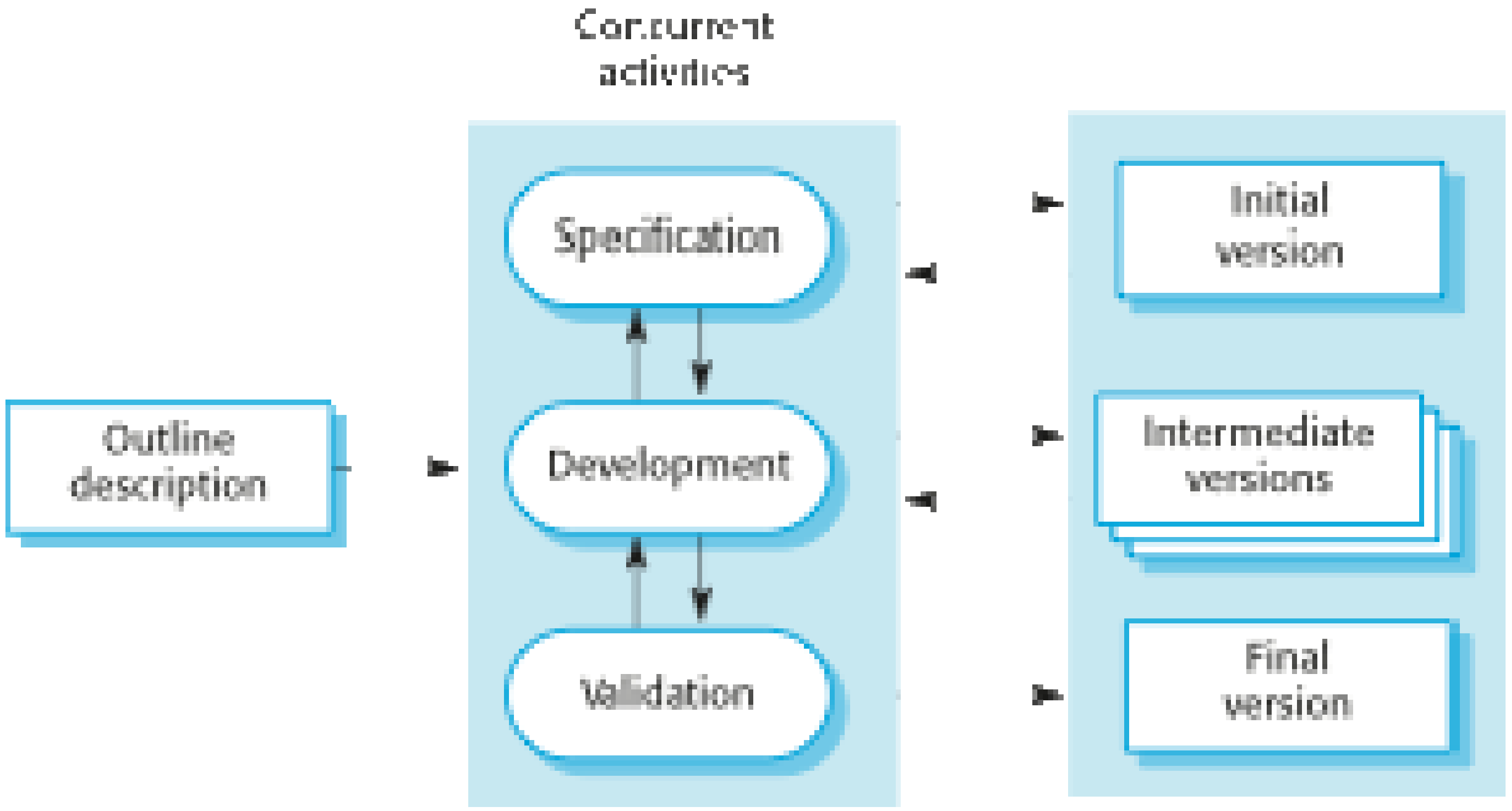
✧ Mostly used for

- Large distributed projects
- Safety and security critical systems
- Systems with many regulations

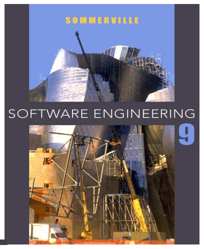
2.1.2 Incremental development



Develop initial implementation => expose to user comment => evolve through several versions



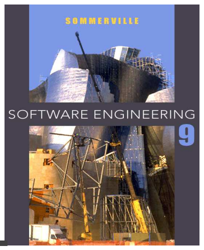
Incremental development



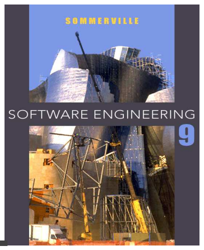
✧ Pro:

- More flexible, cost of change reduced
- Easier to get customer feedback
- More rapid delivery and deployment of useful software

Incremental development



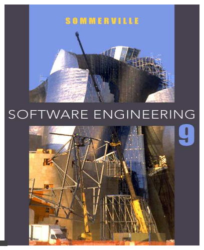
- ✧ Pro:
  - More flexible, cost of change reduced
  - Easier to get customer feedback
  - More rapid delivery and deployment of useful software
- ✧ Con:
  - Process not visible to managers
  - System structure tends to degrade (refactoring needed)
- ✧ Used for business, e-commerce, web applications, ..



What benefits can you see in incremental development?

What draw backs?

Incremental development



Can be used in plan-driven and agile processes

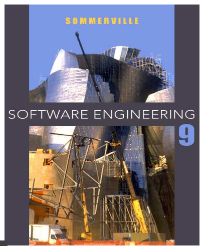
✧ Plan Driven:

- System increments are identified in advance

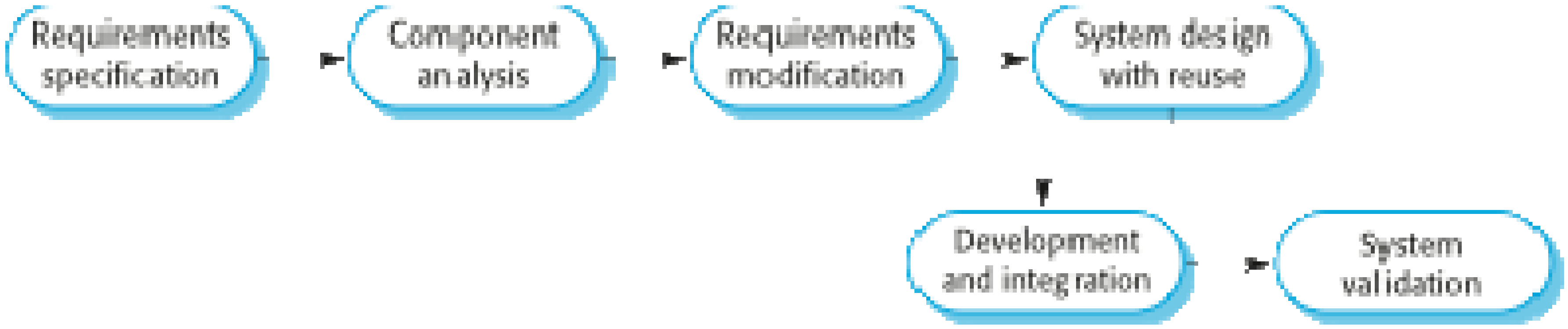
✧ Agile:

- Early increments are identified up front
- Later increments depend on progress and customer priorities

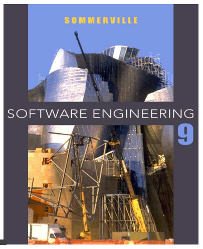
2.1.3 Reuse-oriented software engineering



✧ Relies on large base or reusable components and an integrating framework

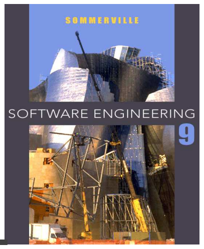

Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

3 types of software components



- ✧ **Web services** .. available for remote invocation.
- ✧ **Collections of objects** developed as a package to be integrated with component frameworks like .NET or J2EE.
- ✧ Stand-alone software systems (**COTS**) that are configured for use in a particular environment.

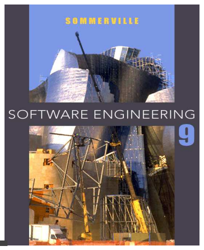
Pro / Con



✧ **Pro:**

- Faster development at lower cost
- Components are already tested

Pro / Con



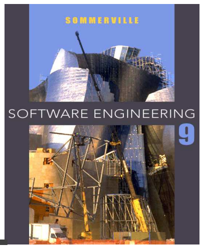
✧ **Pro:**

- Faster development at lower cost
- Components are already tested

✧ **Con:**

- Requirements compromises are inevitable
- Control over system evolution reduced (new versions)

Pro / Con



✧ **Pro:**

- Faster development at lower cost
- Components are already tested

✧ **Con:**

- Requirements compromises are inevitable
- Control over system evolution reduced (new versions)

✧ Reuse is now the standard approach for building many types of business system