AARHUS
UNIVERSITY
DEPARTMENT OF ENGINEERING

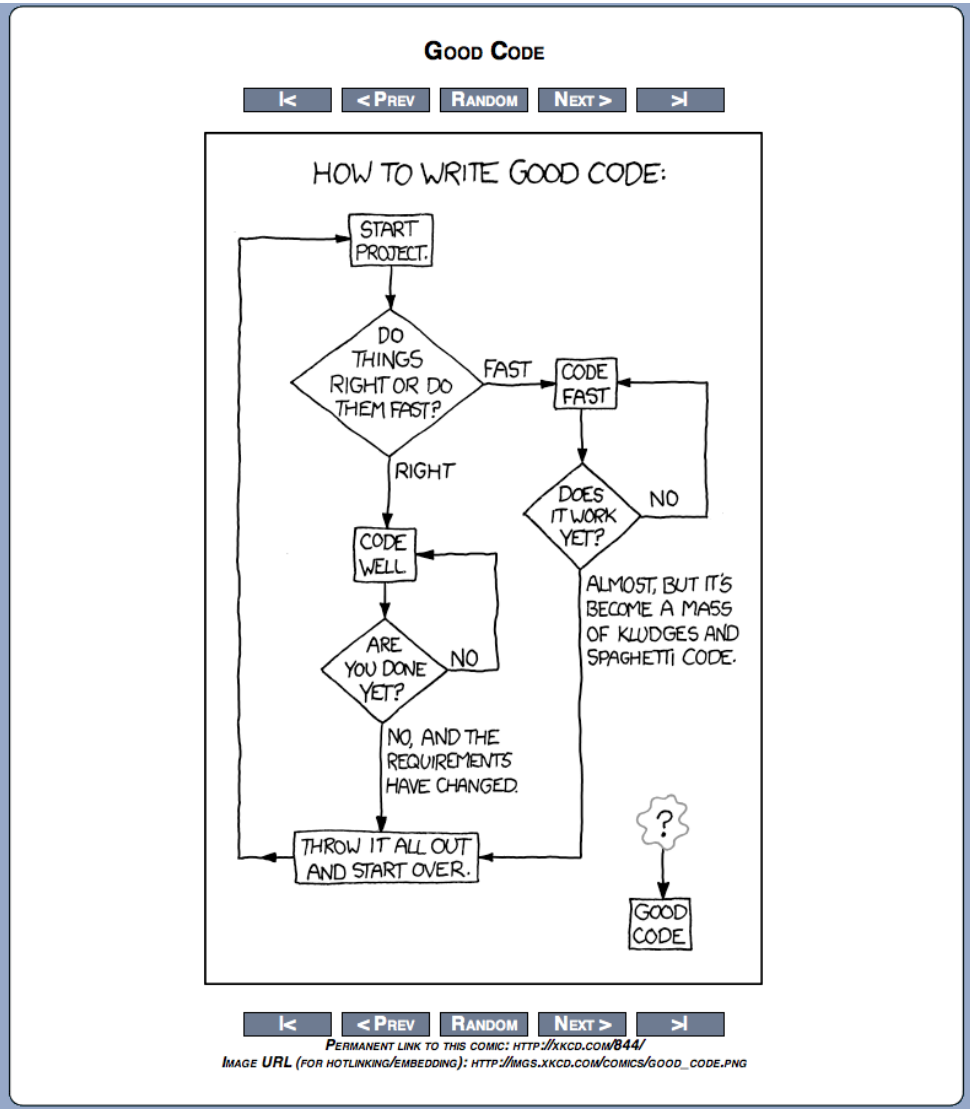
SOFTWARE ENGINEERING PRINCIPLES DEVELOPMENT PROCESSES

STEFAN HALLESTEDE
CARL SCHULTZ

FIRST ATTEMPT AT SW DEVELOPMENT

› Source: xkcd

› How can we know whether this ”works”?

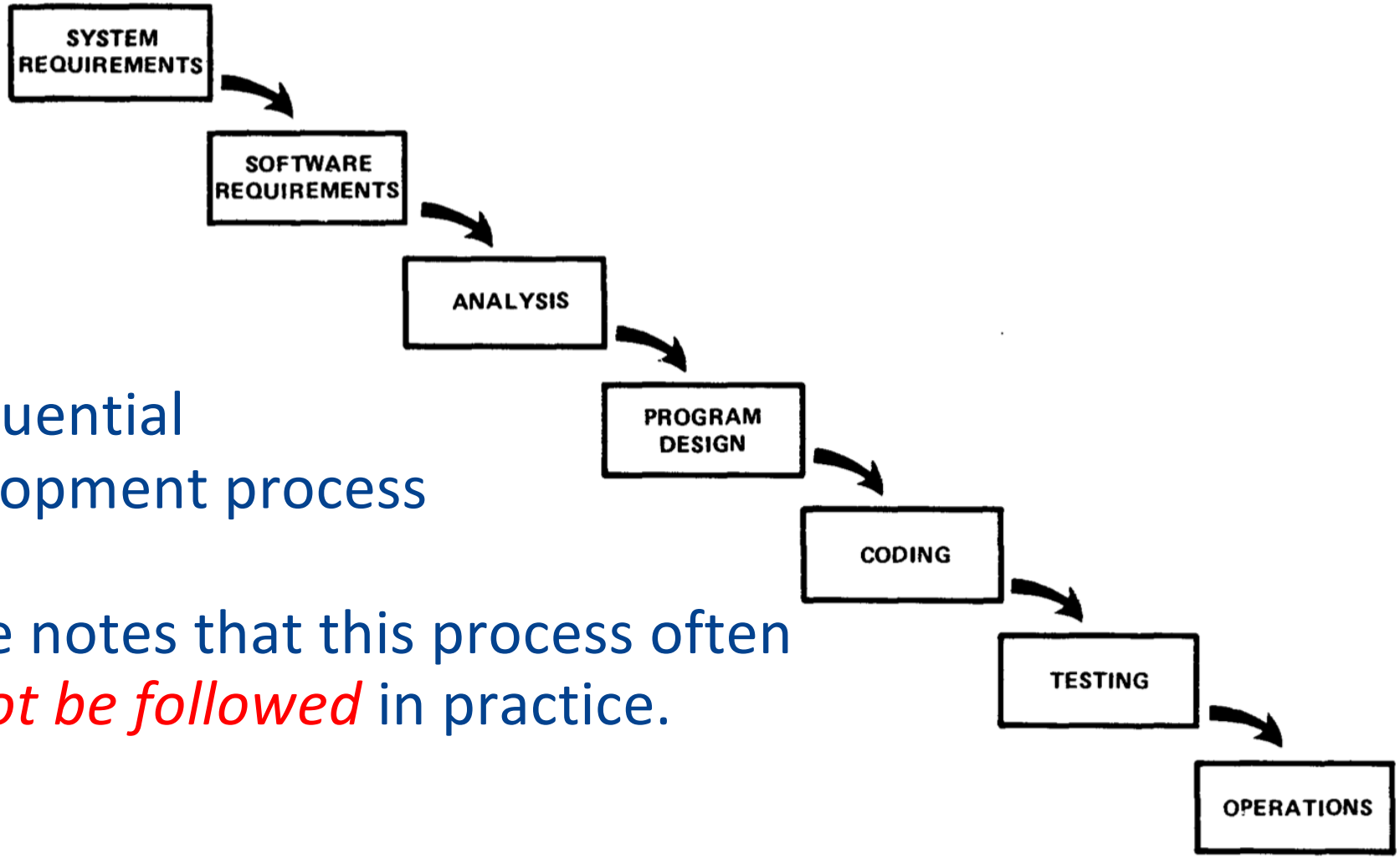


WHAT DO WE EXPECT OF A SOFTWARE PROCESS?

- What is the *aim* of having one?
- What is a *good* software process?
- What is *good* software?

Correctness Reliability
Efficiency Maintainability Portability
Integrity Testability Reusability
Usability Flexibility Interoperability

THE WATERFALL MODEL

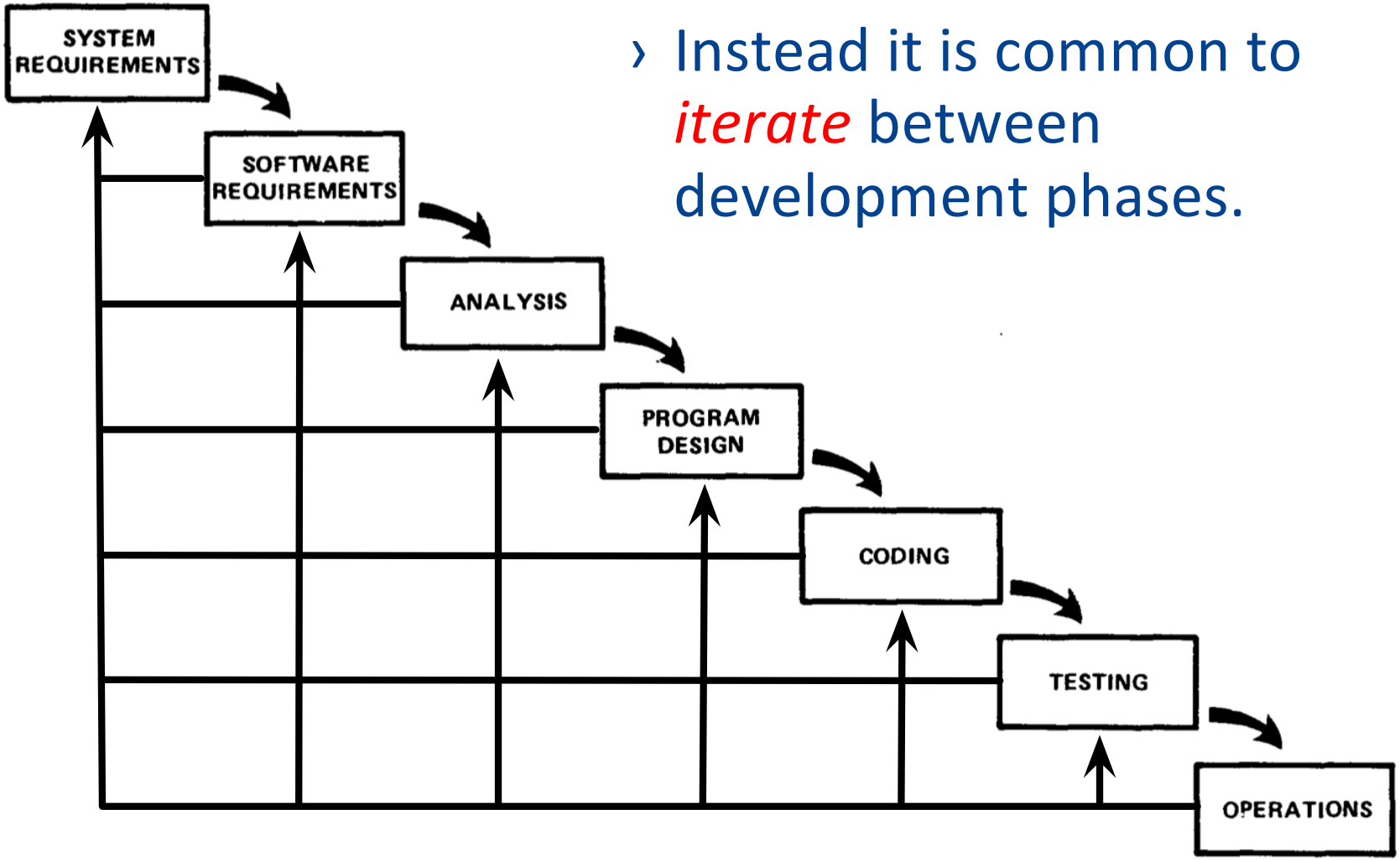


› A sequential development process

› Royce notes that this process often *cannot be followed* in practice.

Winston W. Royce (1970) *Managing the Development of Large Software Systems*. IEEE WESCON

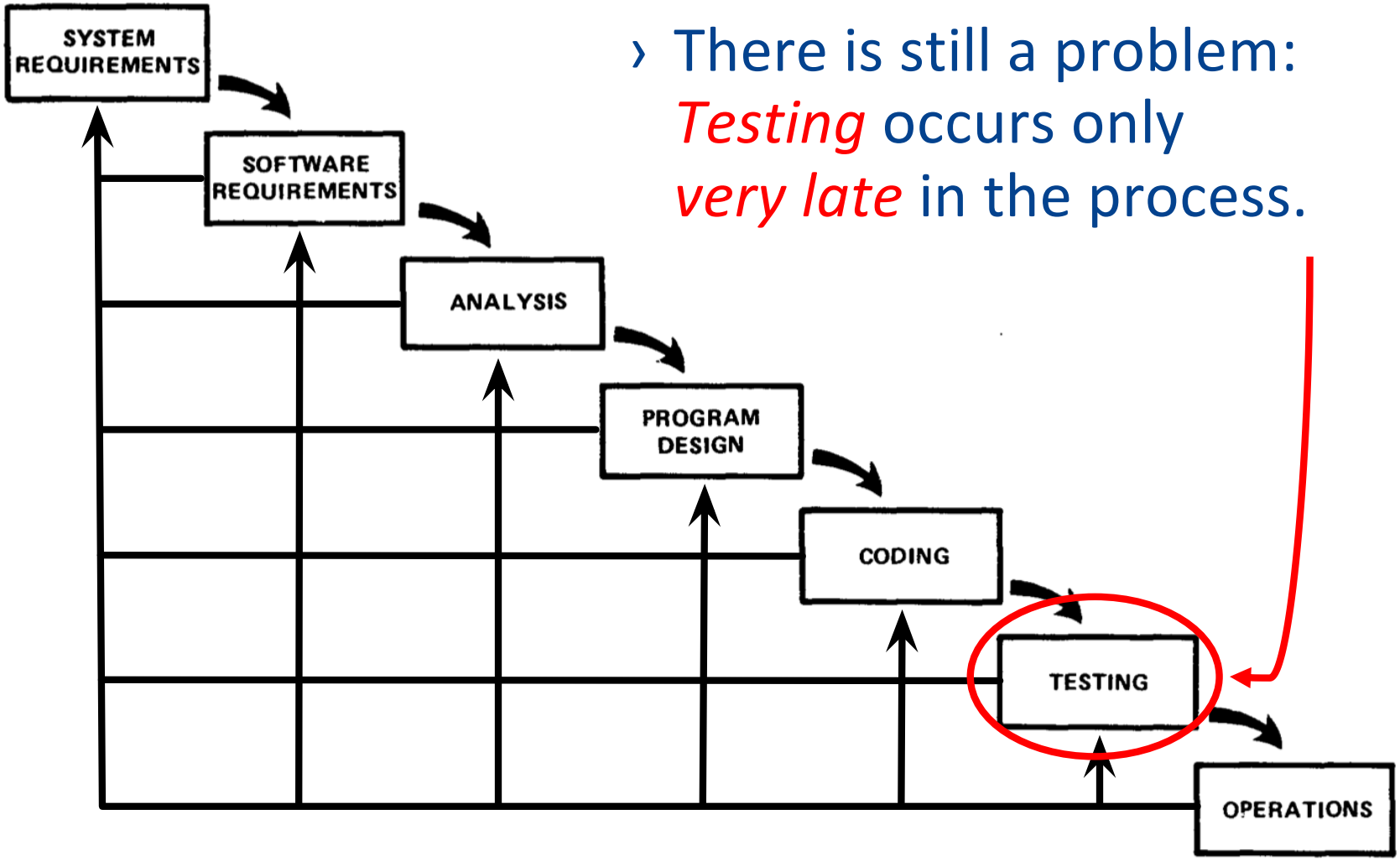
THE WATERFALL MODEL



› Instead it is common to *iterate* between development phases.

Winston W. Royce (1970) *Managing the Development of Large Software Systems*. IEEE WESCON

THE WATERFALL MODEL

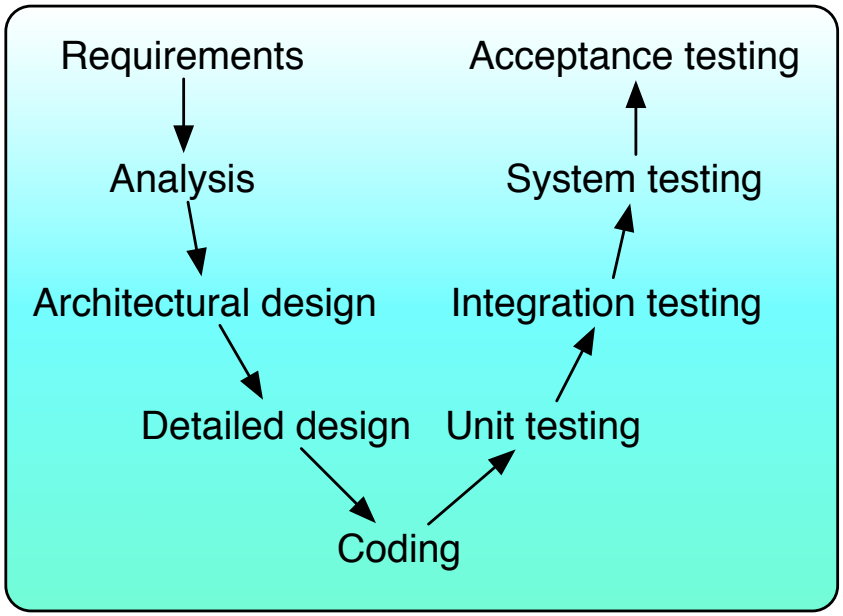


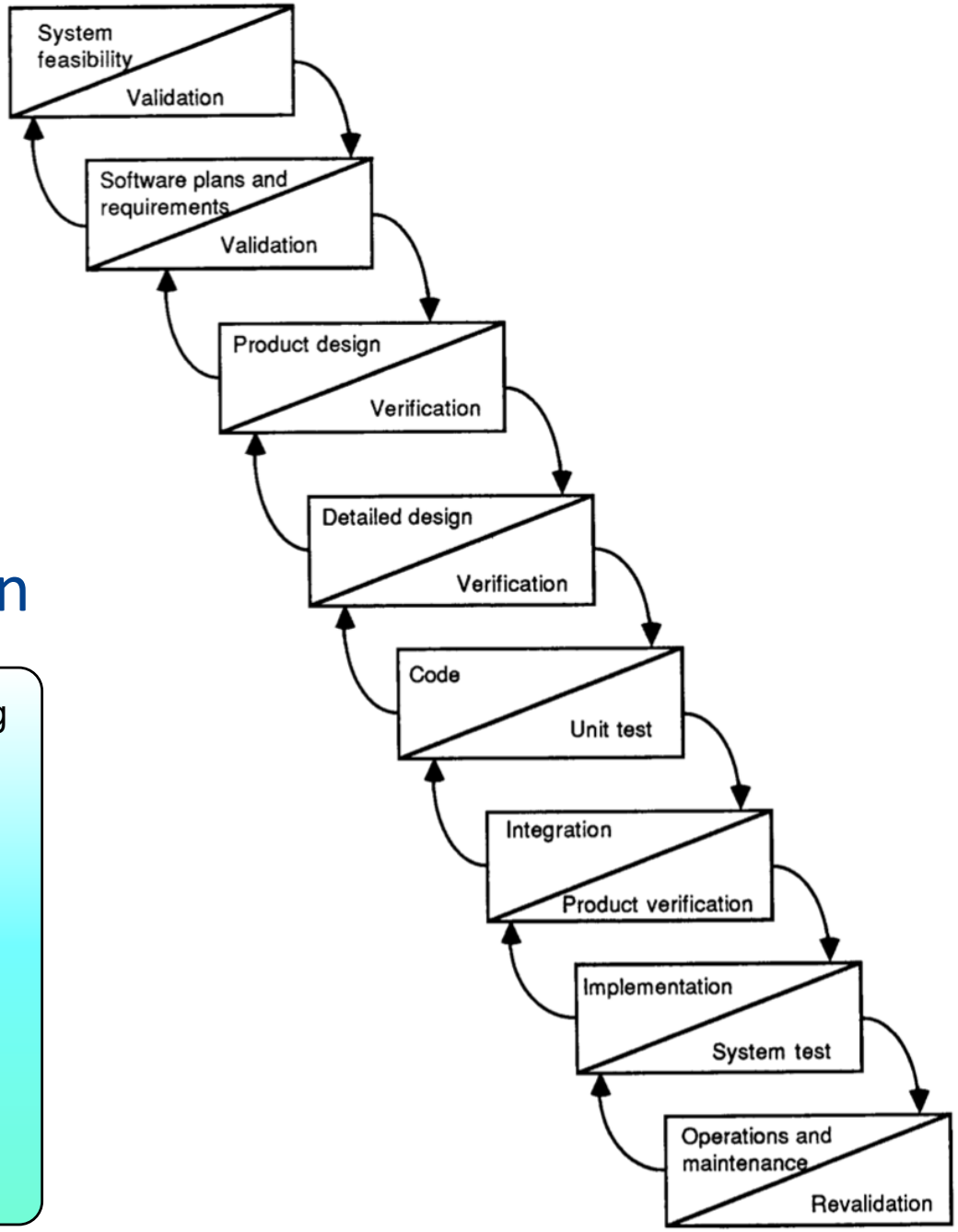
› There is still a problem: *Testing* occurs only *very late* in the process.

Winston W. Royce (1970) *Managing the Development of Large Software Systems*. IEEE WESCON

THE V-MODEL

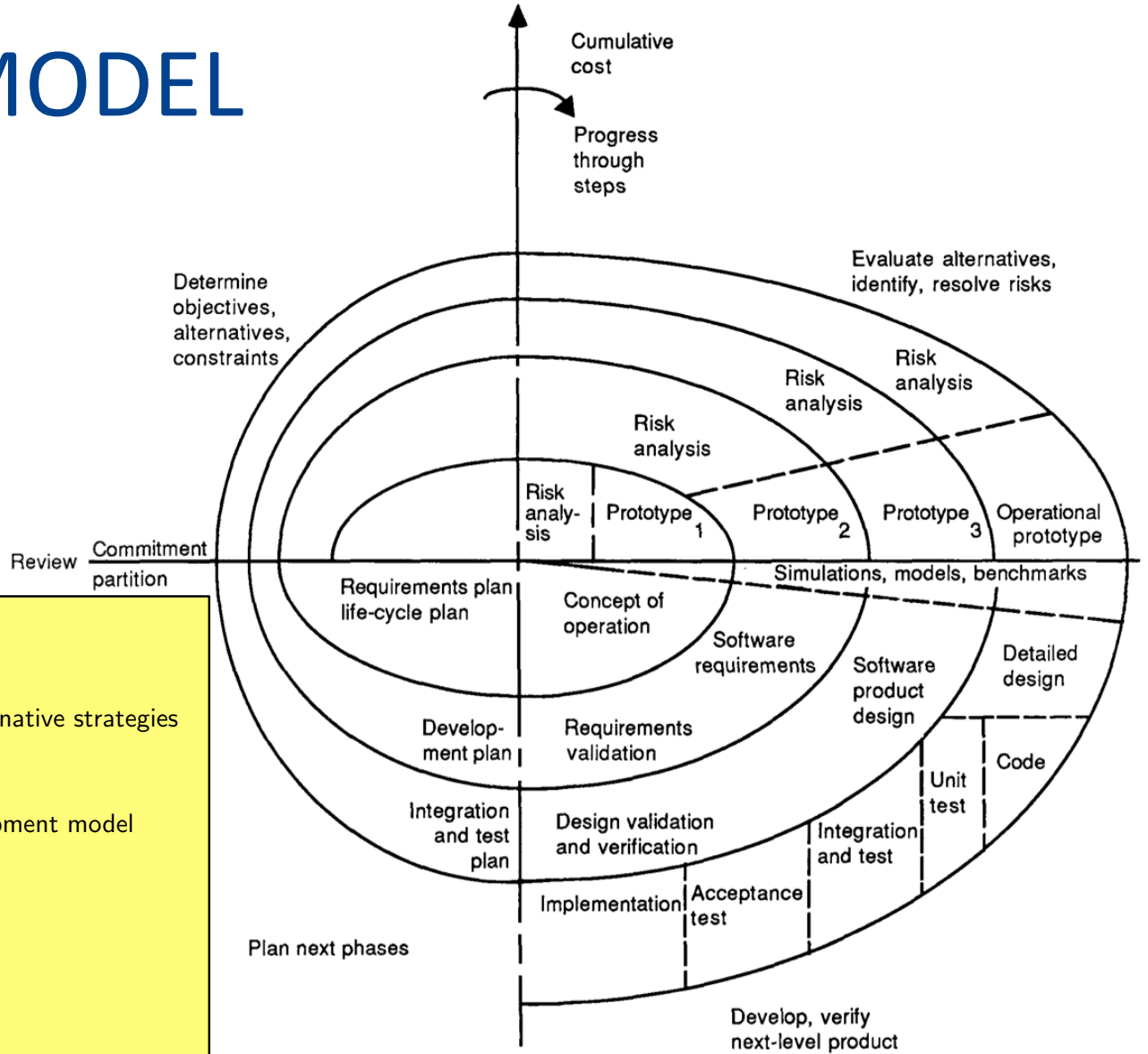
› A variant of the Waterfall Model that emphasises validation and verification





Barry W. Boehm (1988) *A Spiral Model of Software Development and Enhancement*. Computer 21(5), IEEE

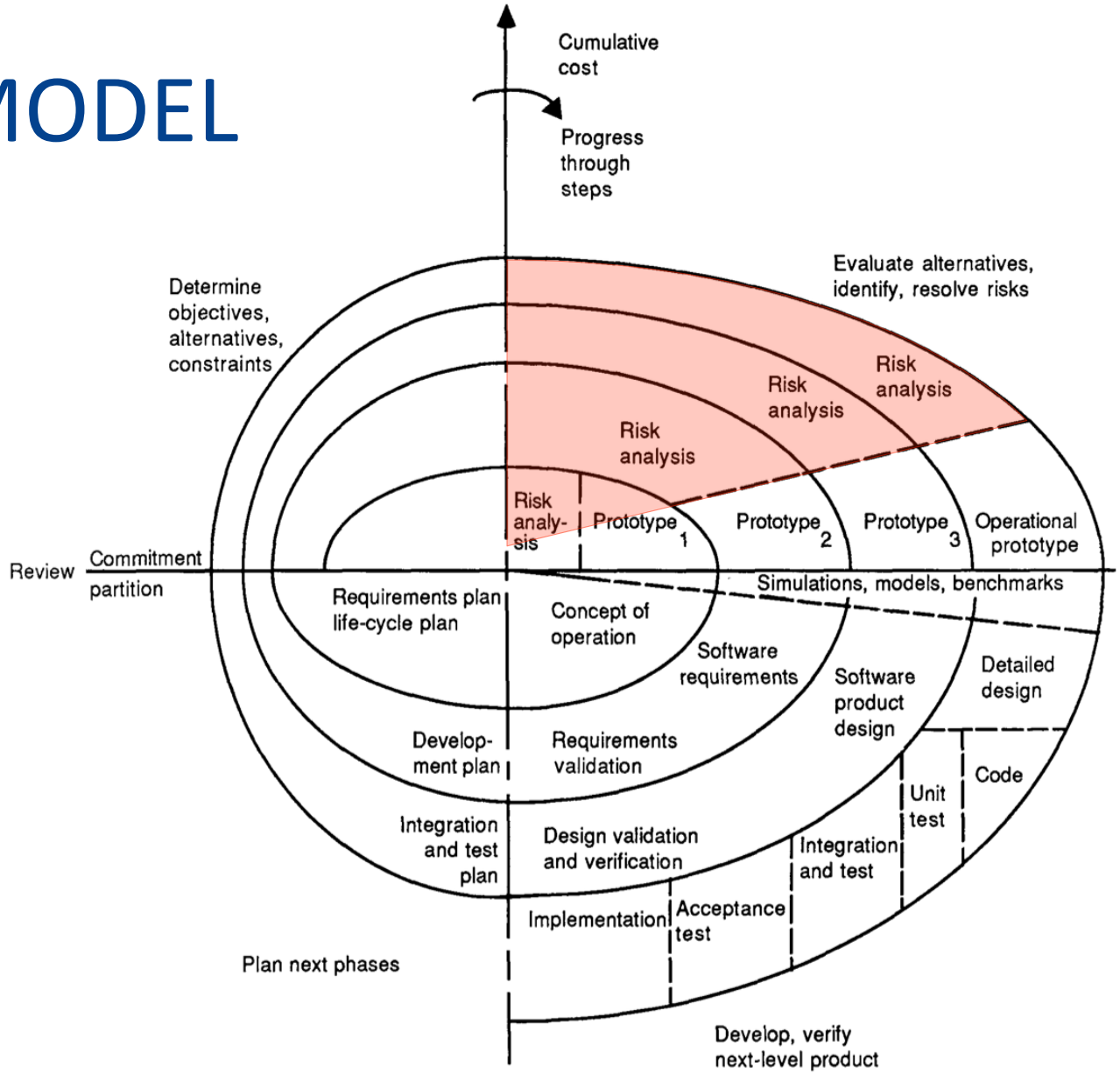
THE SPIRAL MODEL



1. Objective setting
   - Definition of objectives for the phase
   - Identification of risks and planning of alternative strategies
2. Risk assessment and reduction
   - Analyse risks, and choose or adapt development model
3. Development and validation
4. Planning
   - Review project
   - Decide whether to continue with spiral
   - Draw up plan for next phase

Barry W. Boehm (1988) *A Spiral Model of Software Development and Enhancement*. Computer 21(5), IEEE

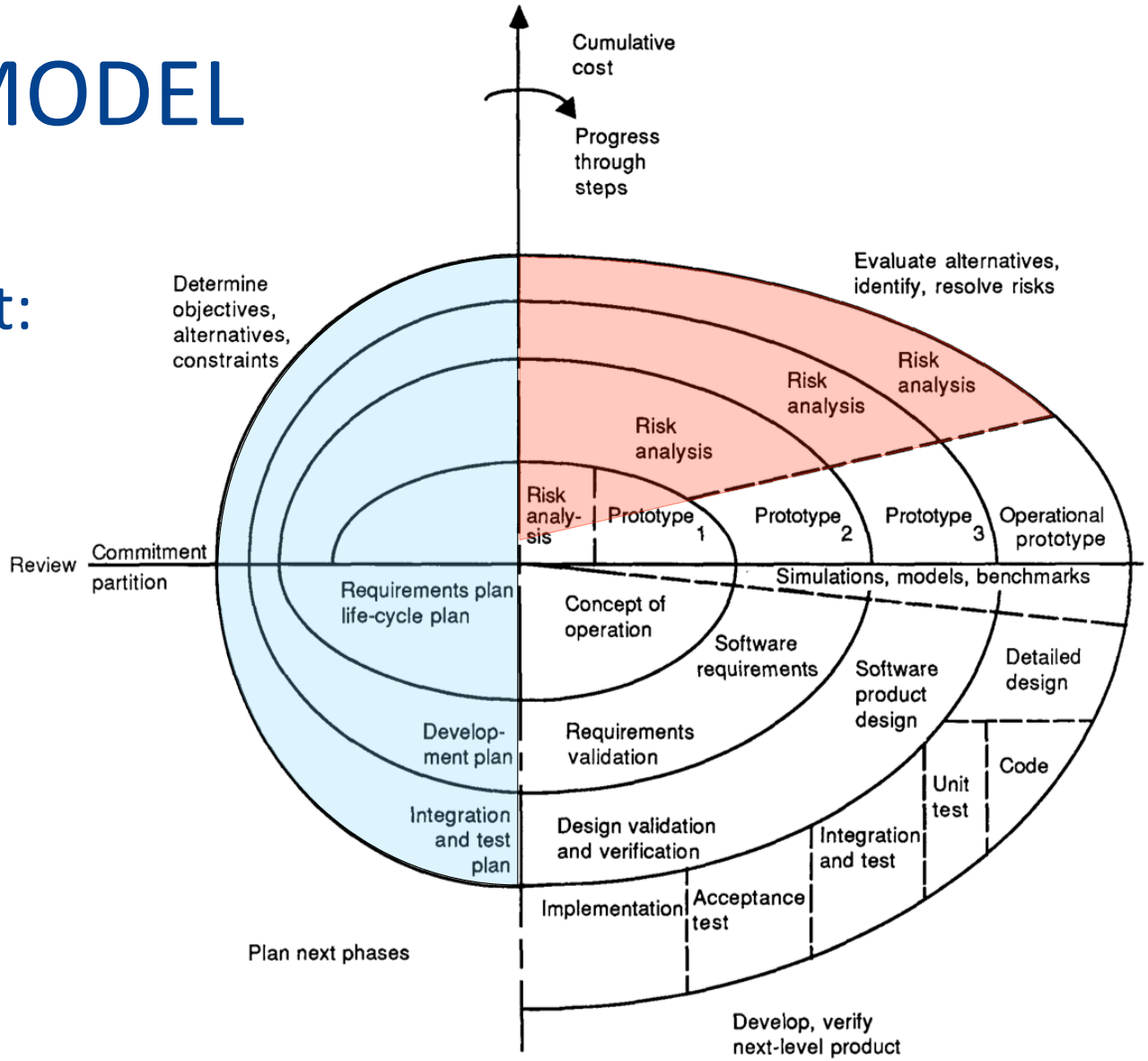
THE SPIRAL MODEL



Barry W. Boehm (1988) *A Spiral Model of Software Development and Enhancement*. Computer 21(5), IEEE

THE SPIRAL MODEL

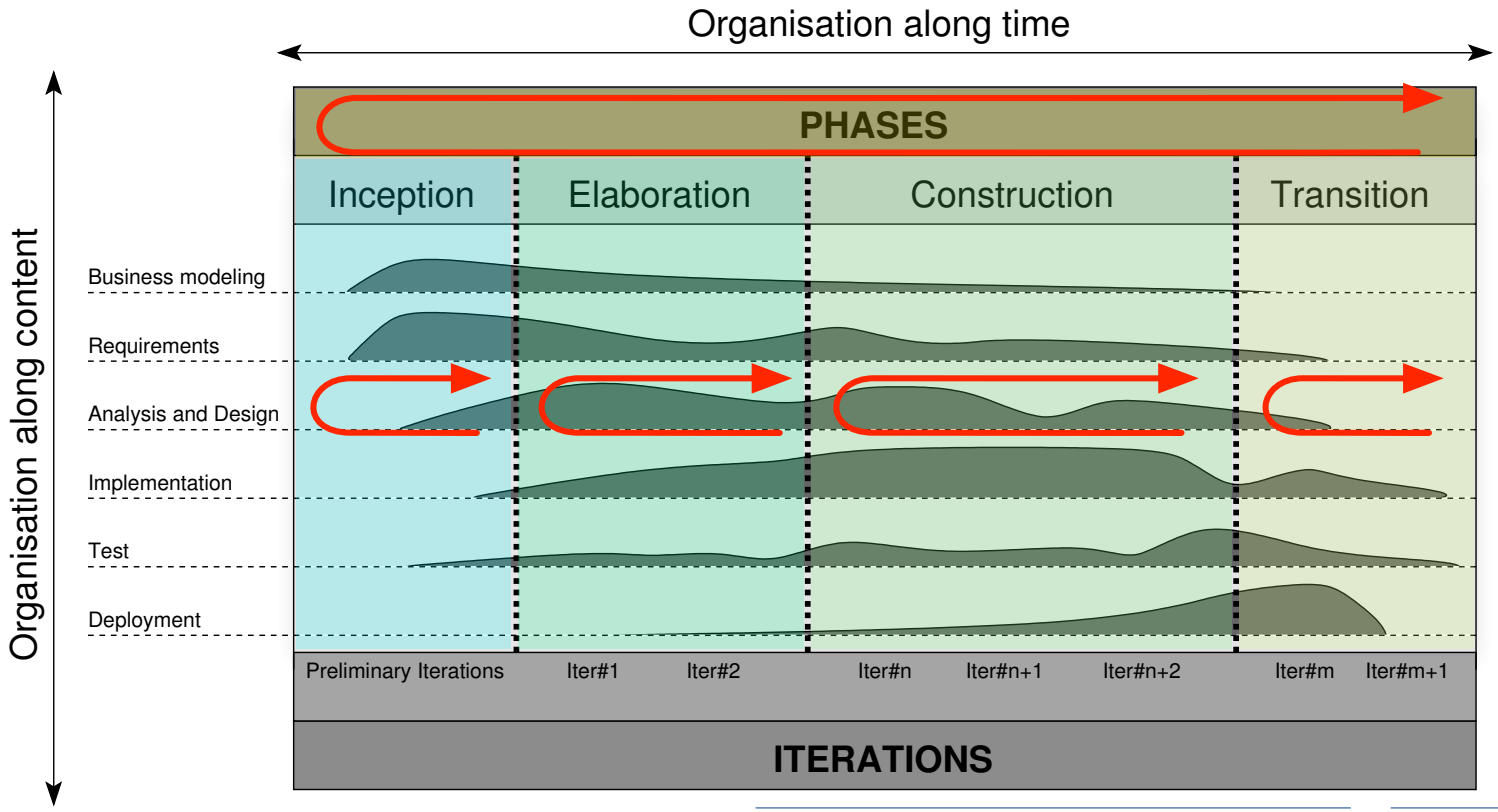
› The Spiral Model takes into account:
risk analysis
and
project management



Barry W. Boehm (1988) *A Spiral Model of Software Development and Enhancement*. Computer 21(5), IEEE

OVERLAPPING PHASES

› Unified Process (incomplete picture)

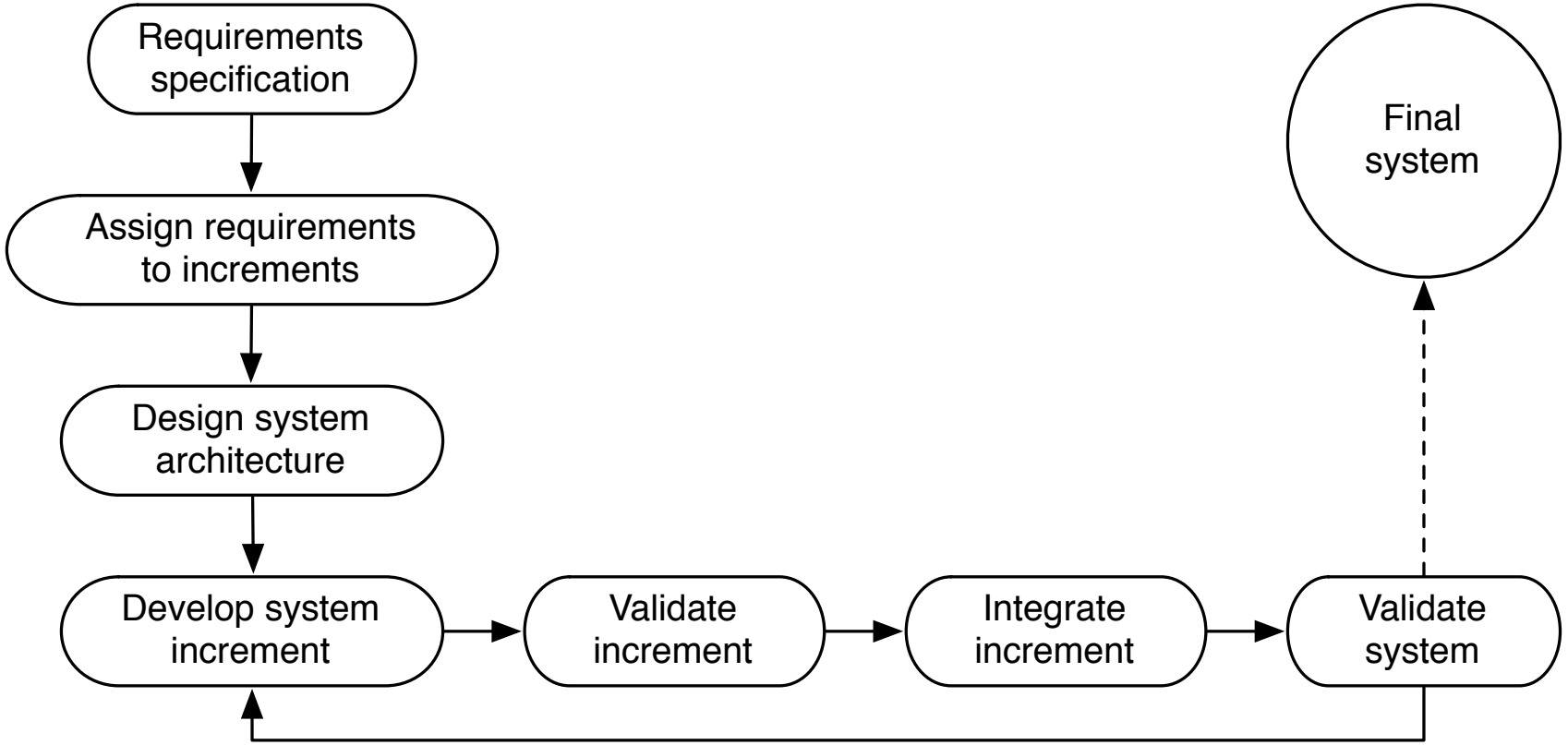


Philippe Kruchten (2004) The rational unified process: an introduction, 3rd ediiton. Addison-Wesley.

SPECIFIC TECHNIQUES

› Incremental Delivery:

› System is not delivered as a whole but in small increments



SPECIFIC TECHNIQUES

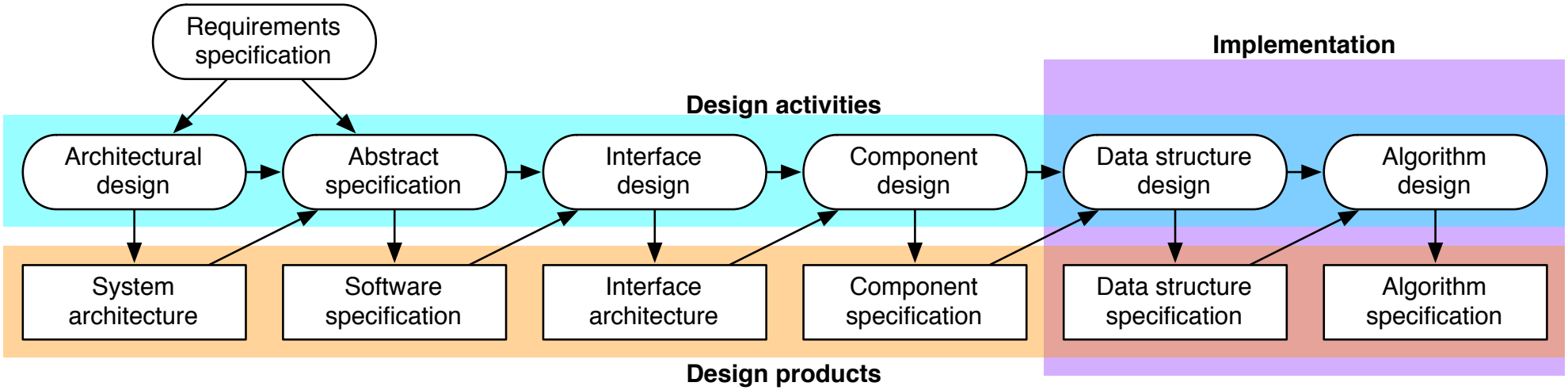
› Component-Based Software Engineering:
› Targeted at reuse
› Process stages adapted to this aim:
› Component analysis:
Search for component matching best the requirements
› Requirements modification:
Modify requirements to reflect available component
› System design with reuse:
Designers incorporate components to be reused,
development of new software only if reuse is not possible
› Development and integration:
Missing pieces of software are implemented, and
reused components integrated

SPECIFIC TECHNIQUES

› Software Design Process Without Reuse:

› Turn requirements specification into an executable system

› Interleaved model of design from architecture to code



› There are a lot of variations in software development.

› It's just like cooking!

AGILE METHODS

› Too much insistence on planning and organisation may impede smaller projects where more flexibility is needed and changes occur often.

› Agile methods attempt to remedy this

› They have strengths and weaknesses
› A host of specific techniques than can be combined with traditional processes

AGILE METHODS

The 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

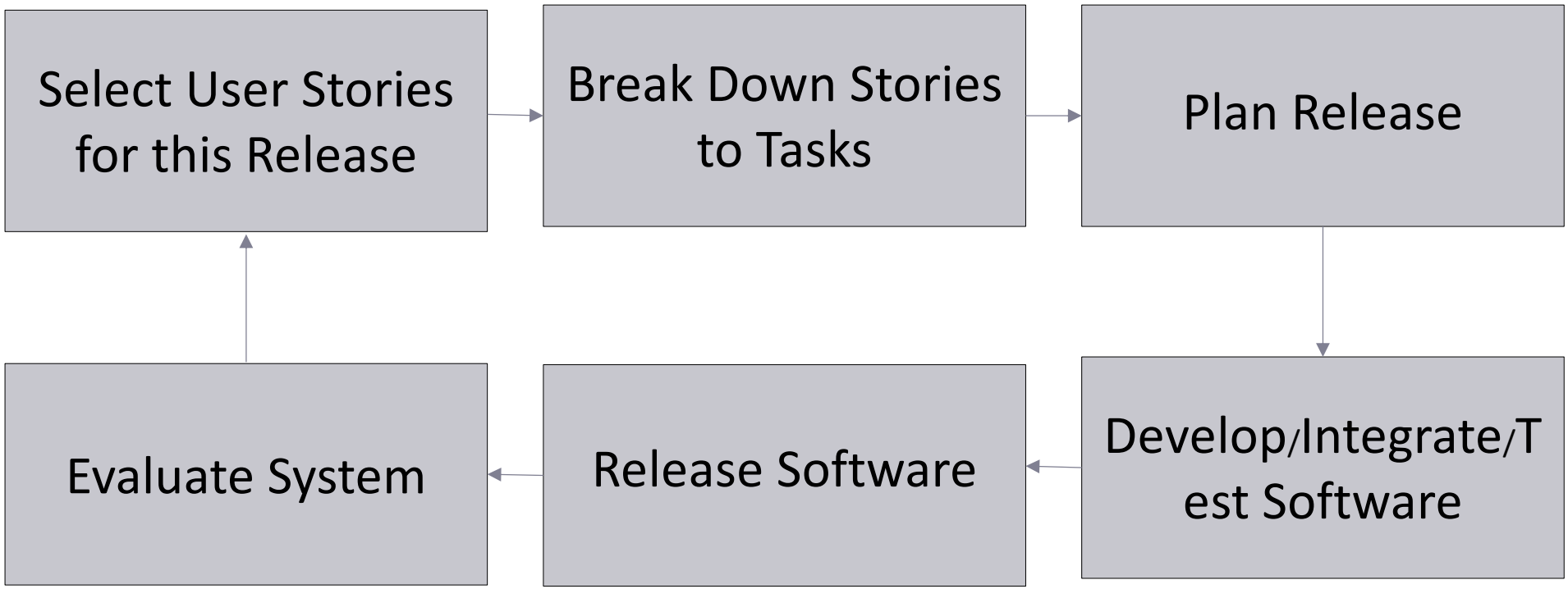
› That is, while there is value in the items on the right, we value the items on the left more.

PRINCIPLES OF AGILE METHODS

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is to provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

EXTREME PROGRAMMING (XP)

› The Extreme Programming Release Cycle



XP PRACTICES

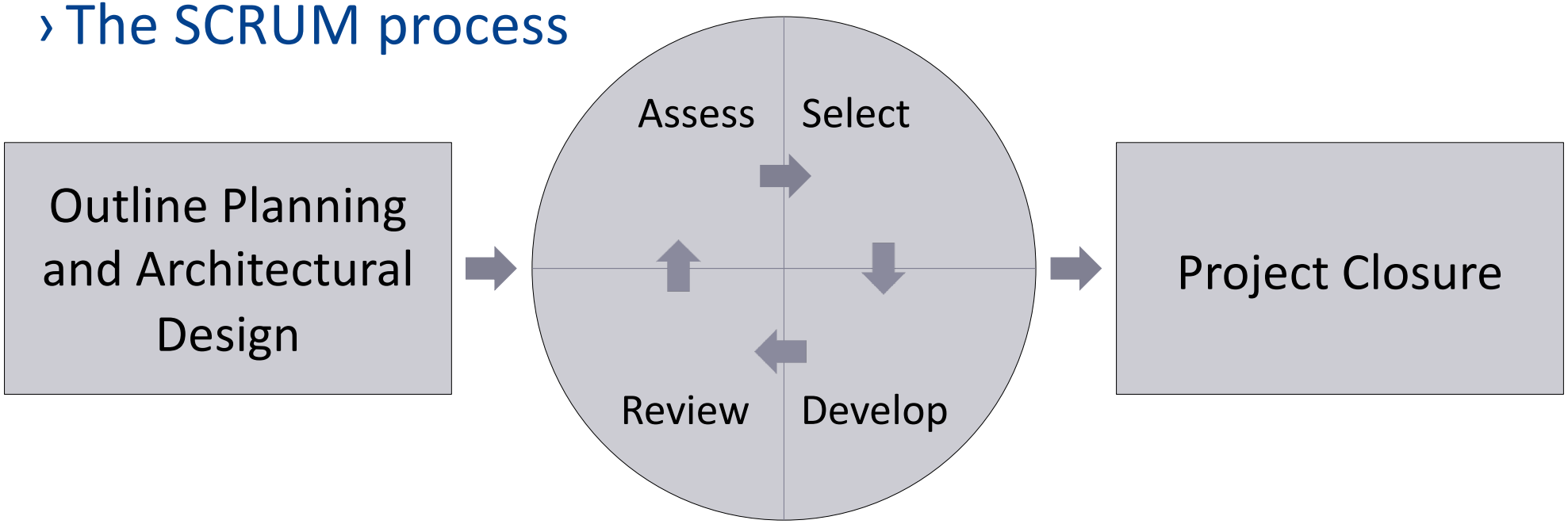
Principle or practice	Description
Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development 'Tasks'.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

XP PRACTICES

Principle or practice	Description
Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

SCRUM – AGILE PROJECT MANAGEMENT

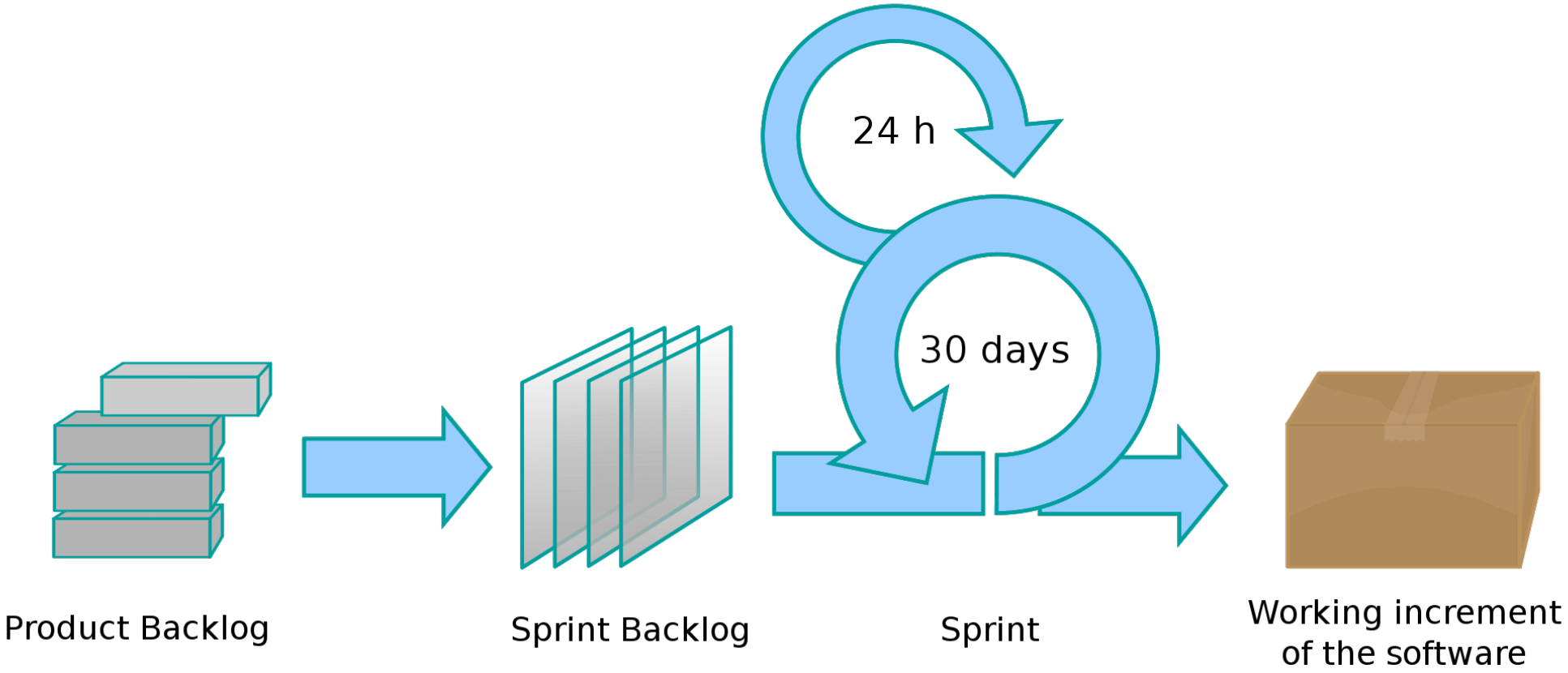
› The SCRUM process



› **Assess** work to be done; **Select** features/functionality; **Develop** features/functionality; **Review** work

SCRUM

› Implementation of an increment



COMPARISON

› How do we decide what method to choose,

› Waterfall
or
› Spiral
or
› Agile?

› Advantages And Disadvantages Of Different Software Processes
(Eric J. Braude, Michael E. Bernstein (2011) Software Engineering – Modern Approaches. Wiley.)

THE WATERFALL MODEL

› Advantages
› Simple and easy to use
› Easy to allocate resources
› Works well when requirements are well understood

› Disadvantages
› Requirements must be known upfront
› Late feedback by users
› Lack of parallelism
› Inefficient use of resources

THE SPIRAL MODEL

› Advantages
› Risks are managed early and throughout the process
› Software evolves as the project progresses
› Planning is built into the process

› Disadvantages
› Complicated to use
› May be overkill for small projects

THE AGILE APPROACH

› Advantages
› The project always has demonstrable results
› Developers tend to be more motivated
› Customers are able to provide better requirements because they can see the evolving product

› Disadvantages
› Problematical for large applications and large teams
› Documentation output is questionable

EXERCISES

- Draw a diagram that describes the process of making a cup of tea.
- What techniques do you use to develop software that needs to be maintainable and can be evolved over a long period of time?