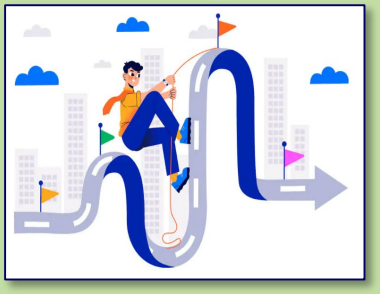


Software Process & Its Models

Fall, 2024

jehong@chungbuk.ac.kr

Topics Covered

Software Process Models

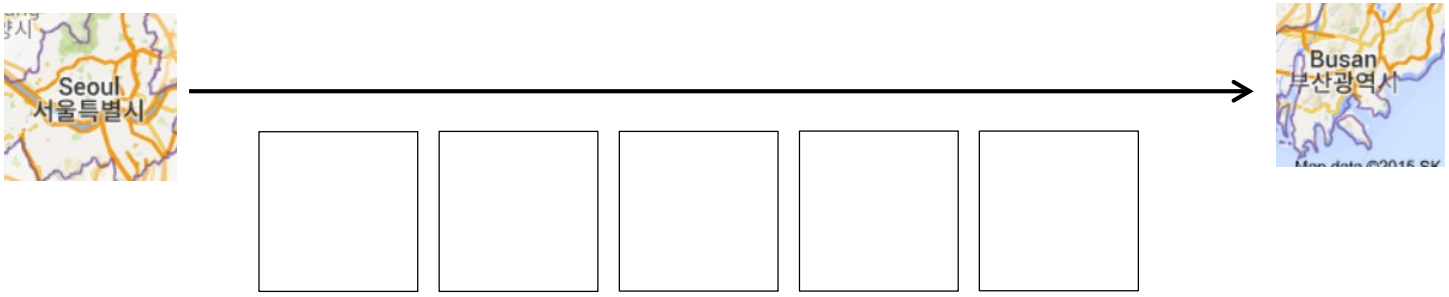
- Waterfall
- Evolutionary (Prototyping, Incremental)
- Agile Model (XP, Scrum)
- Transformation
- Spiral Model
- V Model
- DevOps Model, ...

SDLC by Project Characteristics

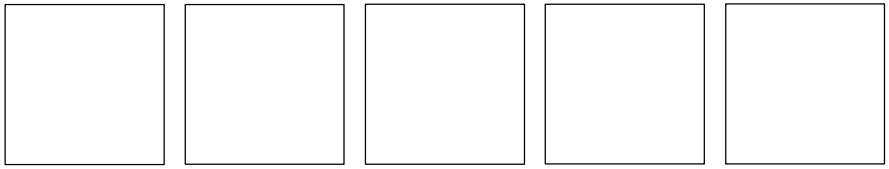
Software Process Standard : ISO 12207

여행가자

How to go final destination?



What something needed to go there?



How about Software Development?

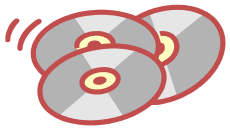
Software Development Process

Software development is Continuous Procedural Activities

What are those activities ?

Requirements

Final System



How to represent these several paths for developing software?

→ _____

What is Software Process Model

Software production process

- The process we follow to build, deliver, deploy, and evolve the software product
- From the inception of an idea all the way to the delivery and final retirement of the system

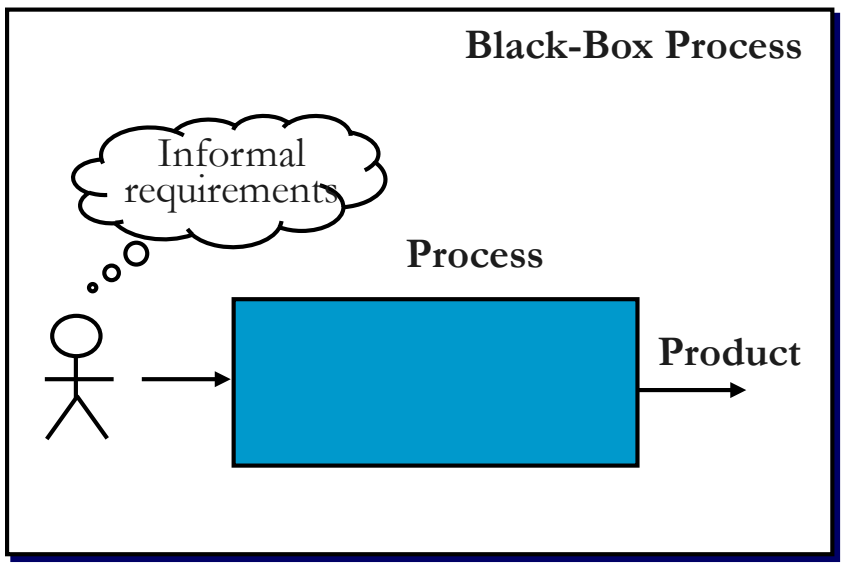
Goal of production process

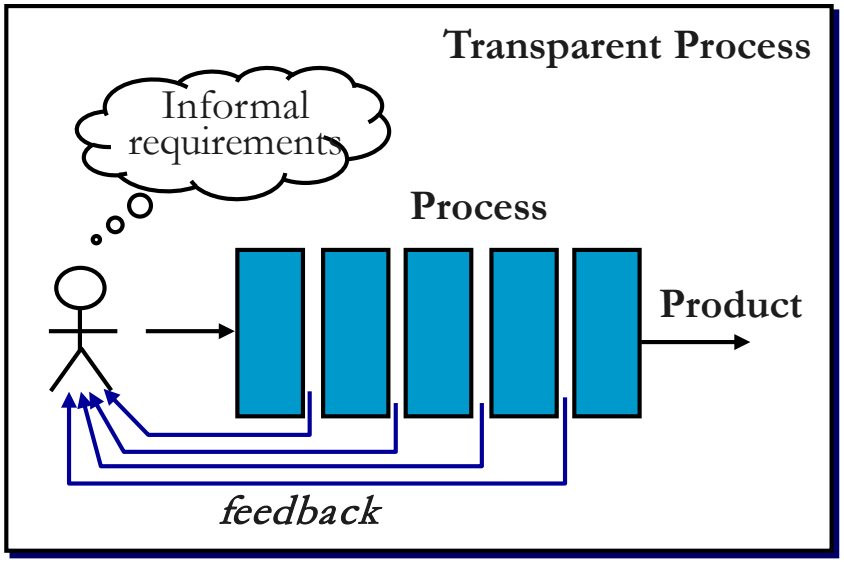
- Satisfy customers' expectations
  - by delivering quality products on time and within budget
  - by making products profitable,
  - by making production reliable, predictable, and efficient

Software life-cycle model

- Requirements → Analysis → Design → Coding → (Testing) → Delivery → Operations and Maintenance → Retirement

Why Process Models Important ?





Improving time to market and reducing production cost

Process has a decisive influence on the quality of products

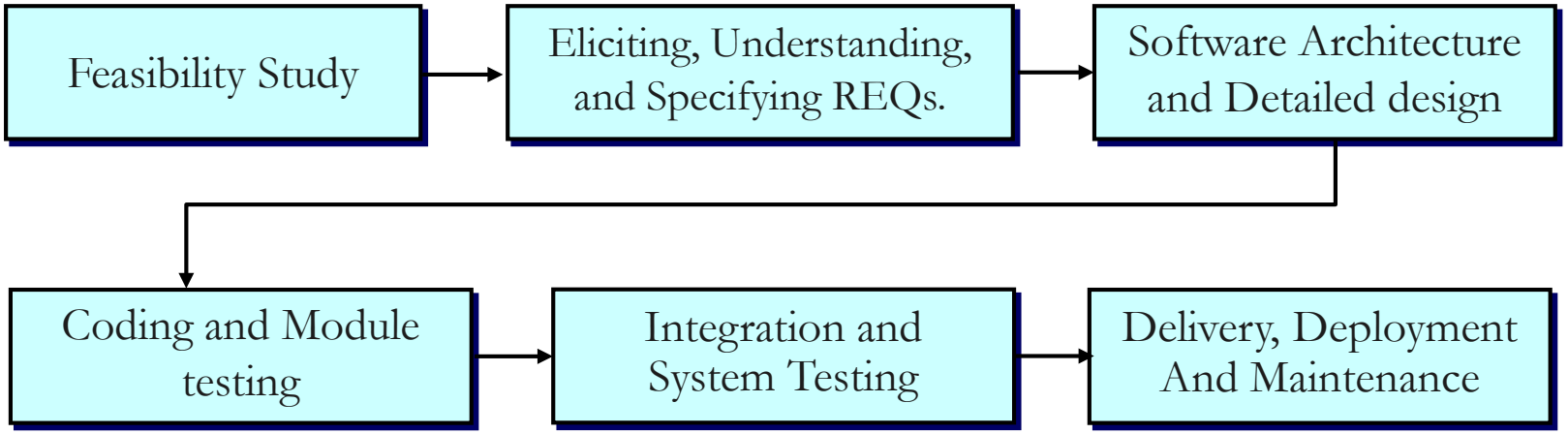
By controlling processes, achieved better control of the required qualities of products.

Activities in Software Production

Process

- A series of (related) steps for what to do

Main Activities



Software Process Models

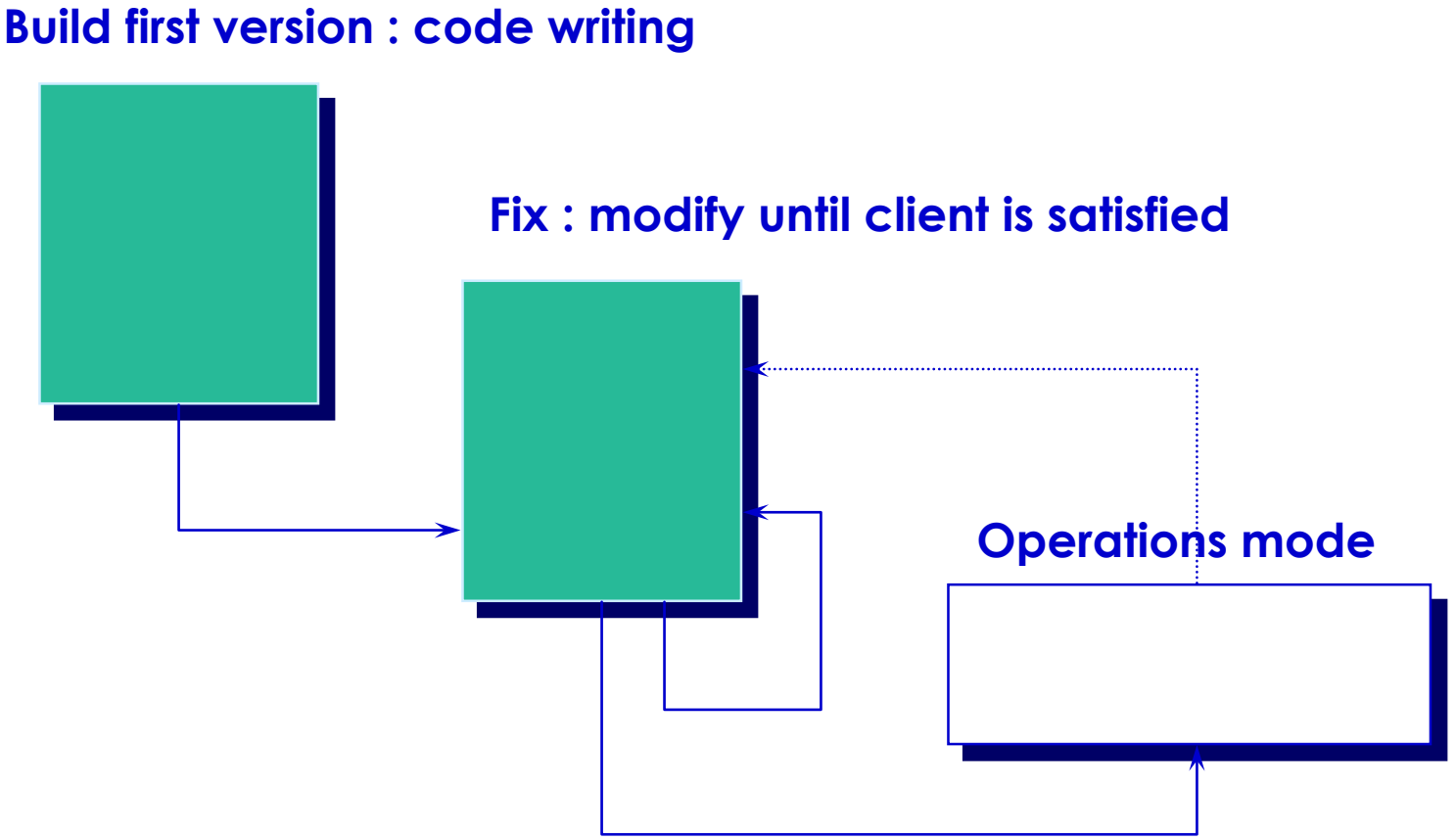
Software process model

- Representation to how the activities of software development can be organized in a process

Representative Process Models

- Build and fix (code and fix) Model
- Waterfall Model
- Evolutionary Model
  - Rapid prototyping
  - Incremental
- Transformation Model
- Spiral Model
- Others

Build-and-Fix Model



Build and Fix Model(Cont.)

Mainly a single-person task.

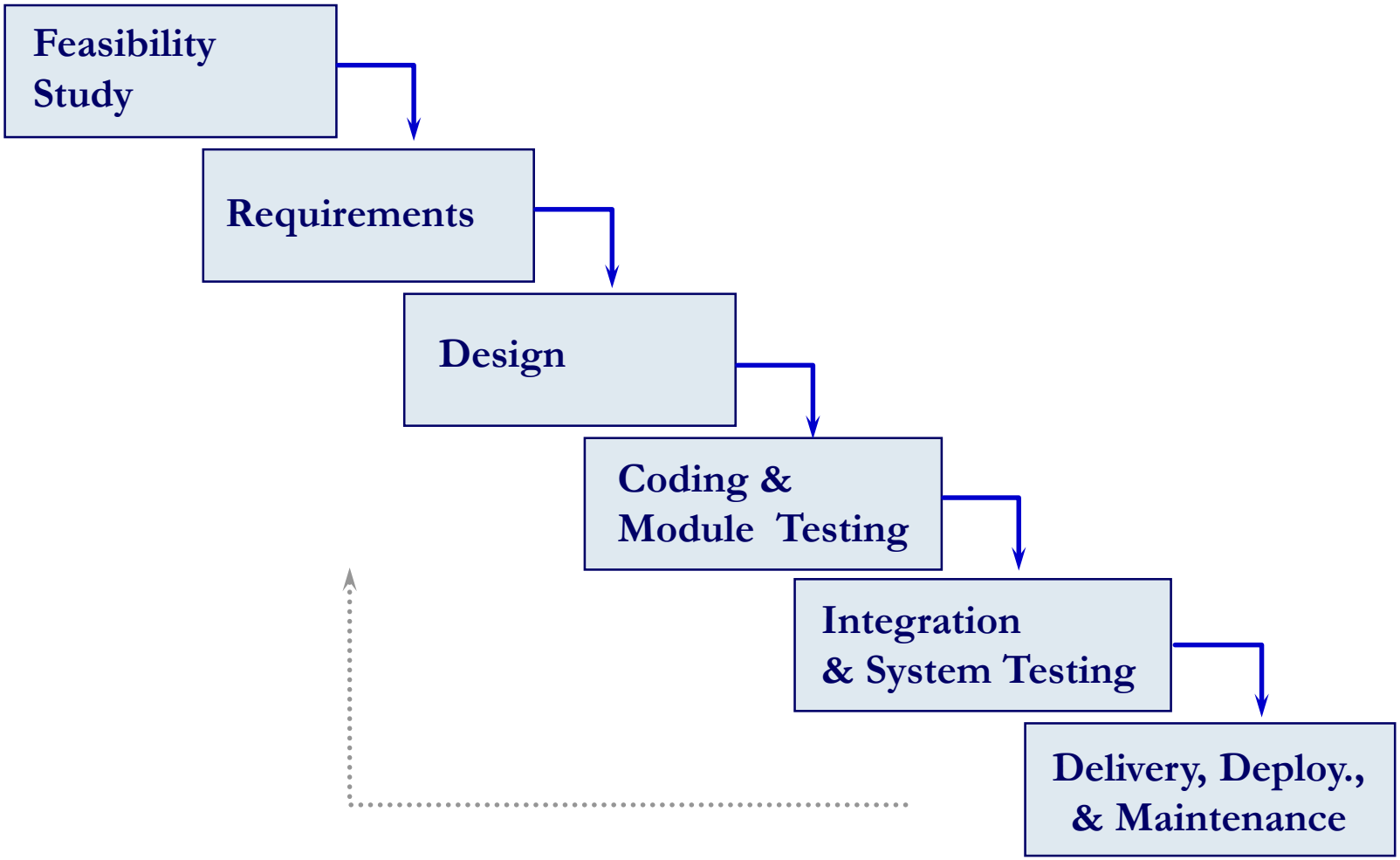
Constructed without specification.

Unsatisfactory for products of reasonable size.

Not suitable for today's environments where

- Developed for people with no computer background
- More stringent reliability requirement
- Group activity

Waterfall Model



Waterfall Model (Cont.)

Popularized industrial practices since 1970's.

Sequential, Phase-based, and Document-oriented

The output of one phase constitutes the input to next.

Contribution

- Enforced disciplined, planned and manageable approach.
- Implementing the product should be postponed until after the objectives of doing so are well understood.

What are the Problems ?

Exists many variants.

- Waterfall process with feedback loop
- Waterfall process with incremental builds
- And so on …

Evolutionary Model

Model whose stages consist of expanding the increments of operational software product

Development Strategy

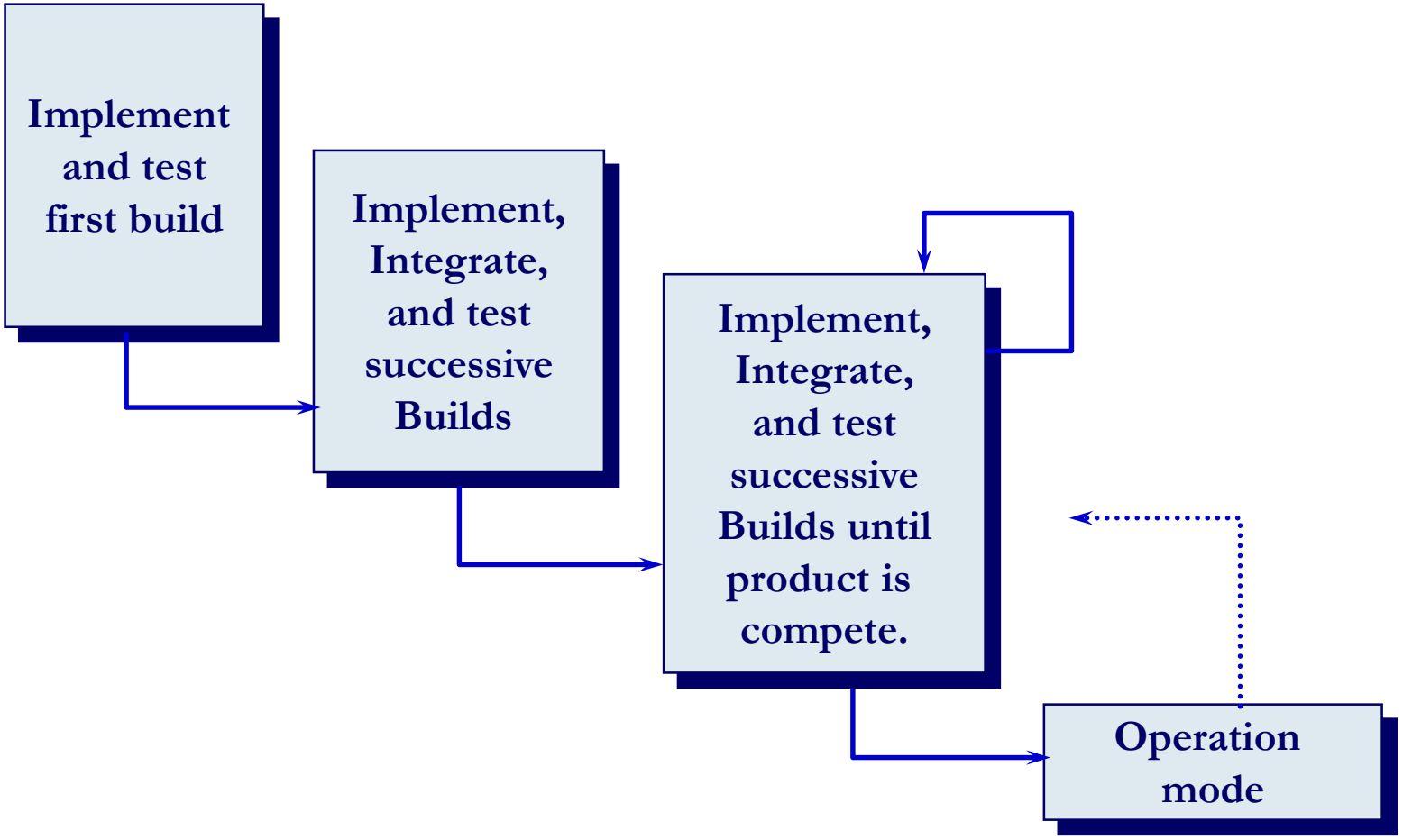
- Deliver (something to the real user)
- Measure (the added value to the user in all critical dimensions)
- Adjust (both the design and the objectives based on observed realities)

Maintenance disappears as a stage of the life cycle

Two types of evolutionary model

- Incremental approach
- Prototyping

Incremental Approach



Incremental Approach (Cont.)

Stepwise development

Must retain the discipline introduced by the waterfall model at each stage : a sequence of miniwaterfall model processes

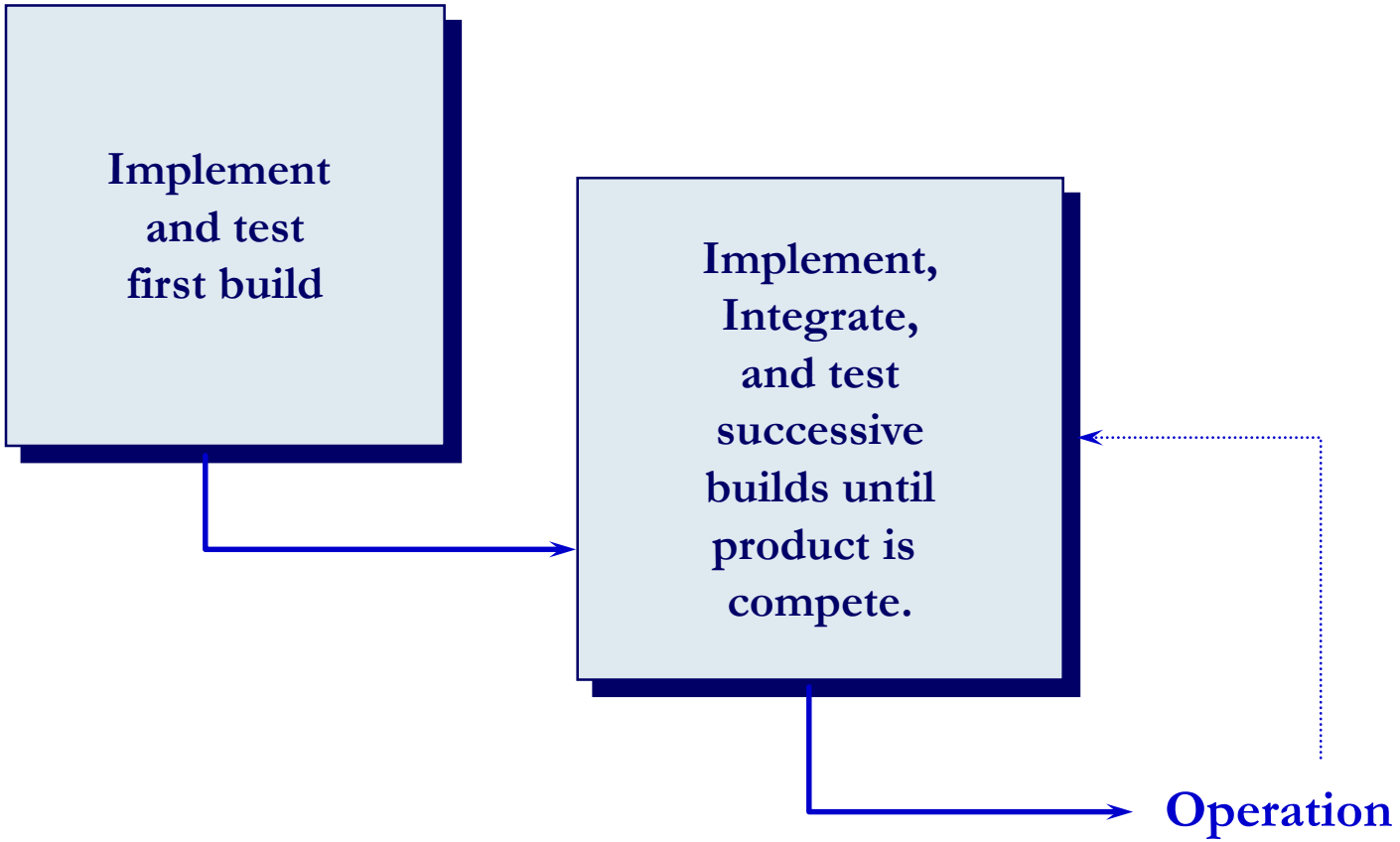
Benefits

- Provide the user with time to adjust to the new product
- Easy to accommodate changes
- Phased delivery does not require a large capital

Problems

- May similar with build-and-fix model
- Overhead at each integration and testing
- Partial system may be considered to final system by user

(Rapid) Prototyping



Prototyping (Cont.)

Do it twice

- First version
  - throwaway prototype to assess the feasibility of the product
  - to verify the requirements
- Second version
  - follow a waterfall model

Prototype can gradually evolves into the final system

Benefits

- Help to reduce cost & time
- Improves communications among developers and between developers and users
- Help to detect error early

Problem

- Does not stress the need for anticipating changes

Rapid Application Development

Q1 : Which ways can be used to quickly develop a software system ?

Q2 : Why we develop the software quickly?

Rapid Application Development

Rapid development and delivery is now often the most important requirement for software systems

- Businesses operate in a fast – changing requirement and it is practically impossible to produce a set of stable software requirements
- Software has to evolve quickly to reflect changing business needs.

In the rapid software development,

- Specification, design and implementation are inter-leaved
- System is developed as a series of versions with stakeholders involved in version evaluation
- User interfaces are often developed using an IDE and graphical toolset.

Rapid Development → Agile Development

Agile Methods

Representative Agile Methods aimed at the same problem: creating reliable software more quickly.

- **Dynamic System Development Method** (Dane Faulkner and others)
- **Adaptive Software Development** (Jim Highsmith)
- **Crystal Clear** (a family of methods, Alistair Cockburn)
- **XP (Kent Beck, Eric Gamma, and others)**
- **Scrum (Ken Schwaber, Jeff Sutherland, Mark Beedle)**
- **Lean** Software Development (Mary and Tom Poppendieck)
- **Feature-Driven Development** (Peter Coad and Jeff DeLuca)
- **Agile Unified Process** (Scott Ambler)

XP Process

eXtreme Programming approach by Kent Beck, 1990s

Dealing with requirement changes

Role & Responsibility

- Programmer: analysis, design, testing, coding, and integration
- Manager: control the progress of project processes
- Customer: requirements & their prioritization

Pursuing 4 values

- Communication
- Simplicity
- Feedback
- Courage

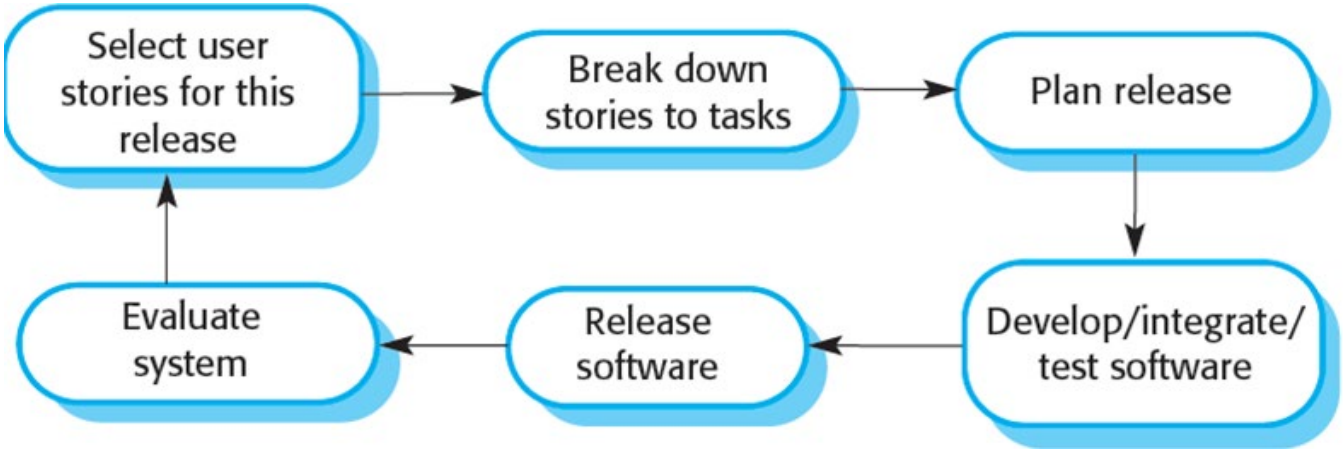
XP Process (Cont'd)

12 Practices in XP

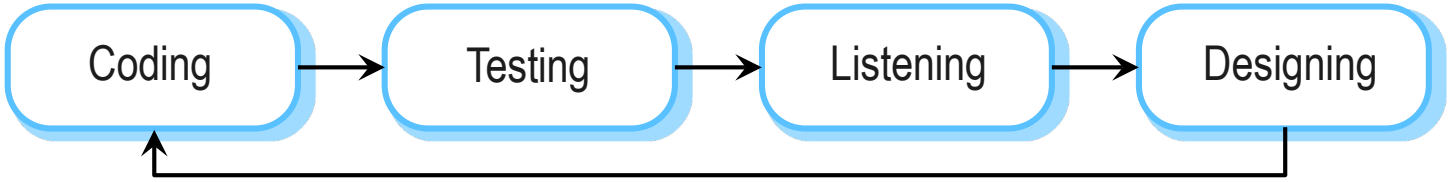
- Planning process
- Small release
- Metaphor
- Simple design
- Continuous testing
- Refactoring
- Pair programming
- Collective code ownership
- Continuous integration
- 40 hour week
- On-site customer
- Coding standard

XP Process – Development Cycles

Release Cycle



Development (Engineering) Cycle



Agile Method – Scrum

A general agile method but its focus is on managing iterative development rather than specific agile practices.

By Jeff Sutherland and Ken Schwaber [Schwaber & Beedle 2002]

Scrum method is …

- A feedback-driven empirical approach which is, like all empirical process control, underpinned by the three pillars of transparency, inspection, and adaptation.
- All work within the Scrum framework should be visible to those responsible for the outcome: the process, the workflow, progress, etc.
- In order to make these things visible, Scrum Teams need to frequently inspect the product being developed and how well the team is working.

Agile Method – Scrum

There are three phases in Scrum.

- The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
- This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
- The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

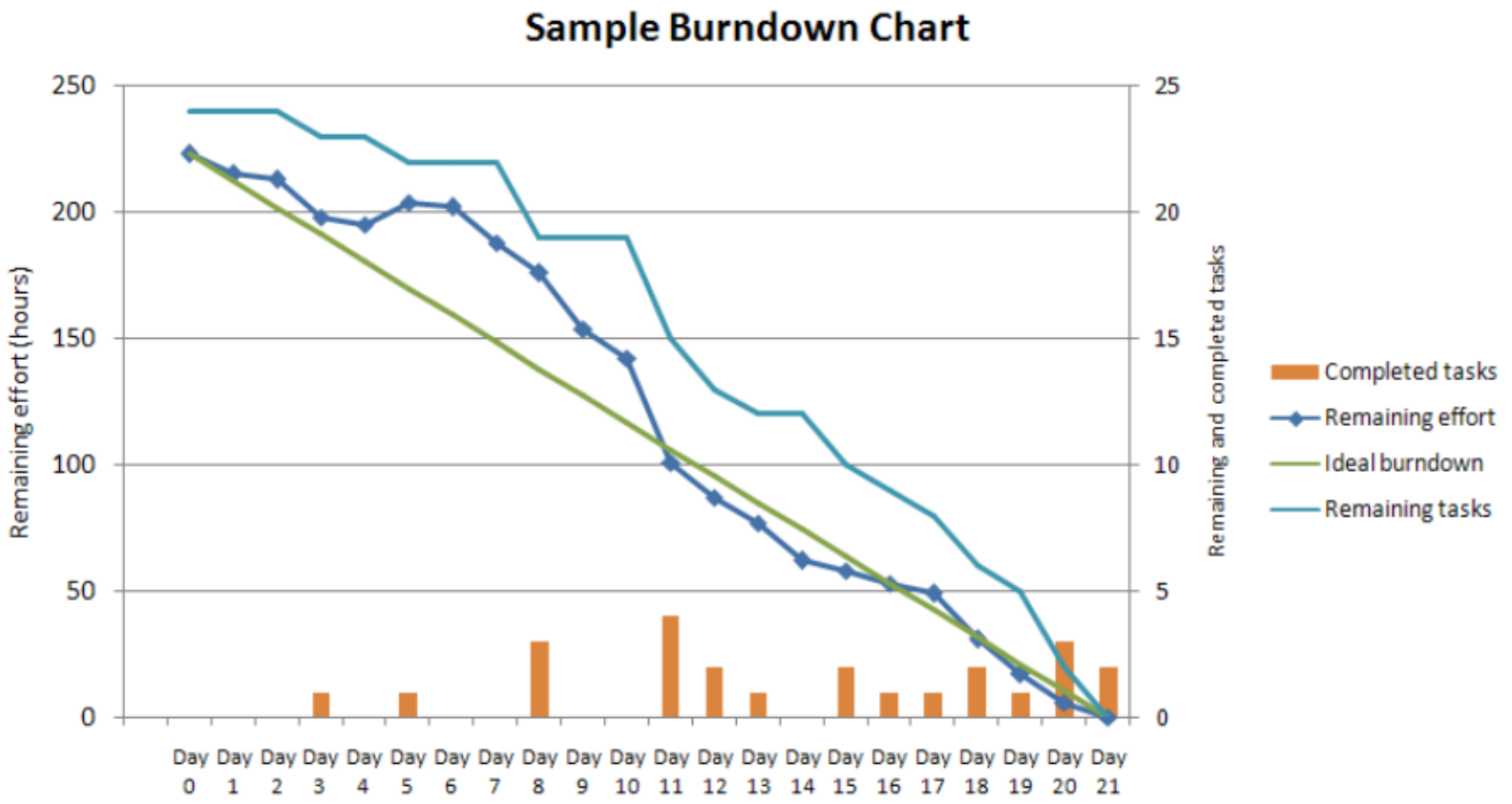
Agile Method – Scrum

Artifacts

- **Product Backlog**
  : an ordered list of requirements that a Scrum Team maintains for a product
- **Sprint Backlog**
  : the list of work the Development Team must address during the next Sprint.
- **Product Increment (or potentially shippable increment, PSI)**
  : the sum of all the Product Backlog Items completed during a Sprint, integrated with the work of all previous Sprints.
- **Burn-Down Chart**
  : the public displayed chart showing remaining work in the Sprint Backlog, updated every day.
  : Burn-Up Chart : a way to provide track progress toward a release

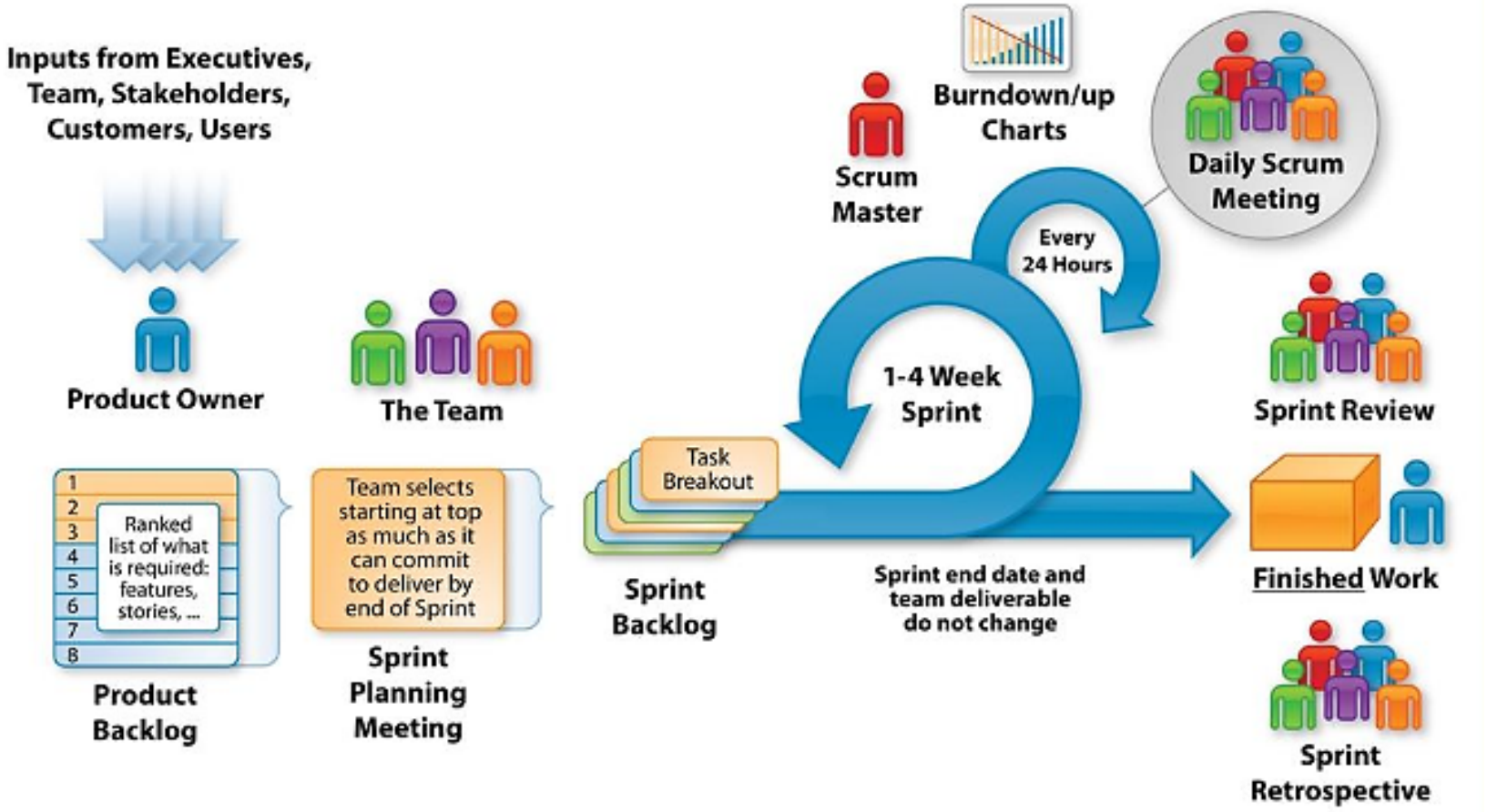
Agile Method – Scrum

Example of Burndown Chart

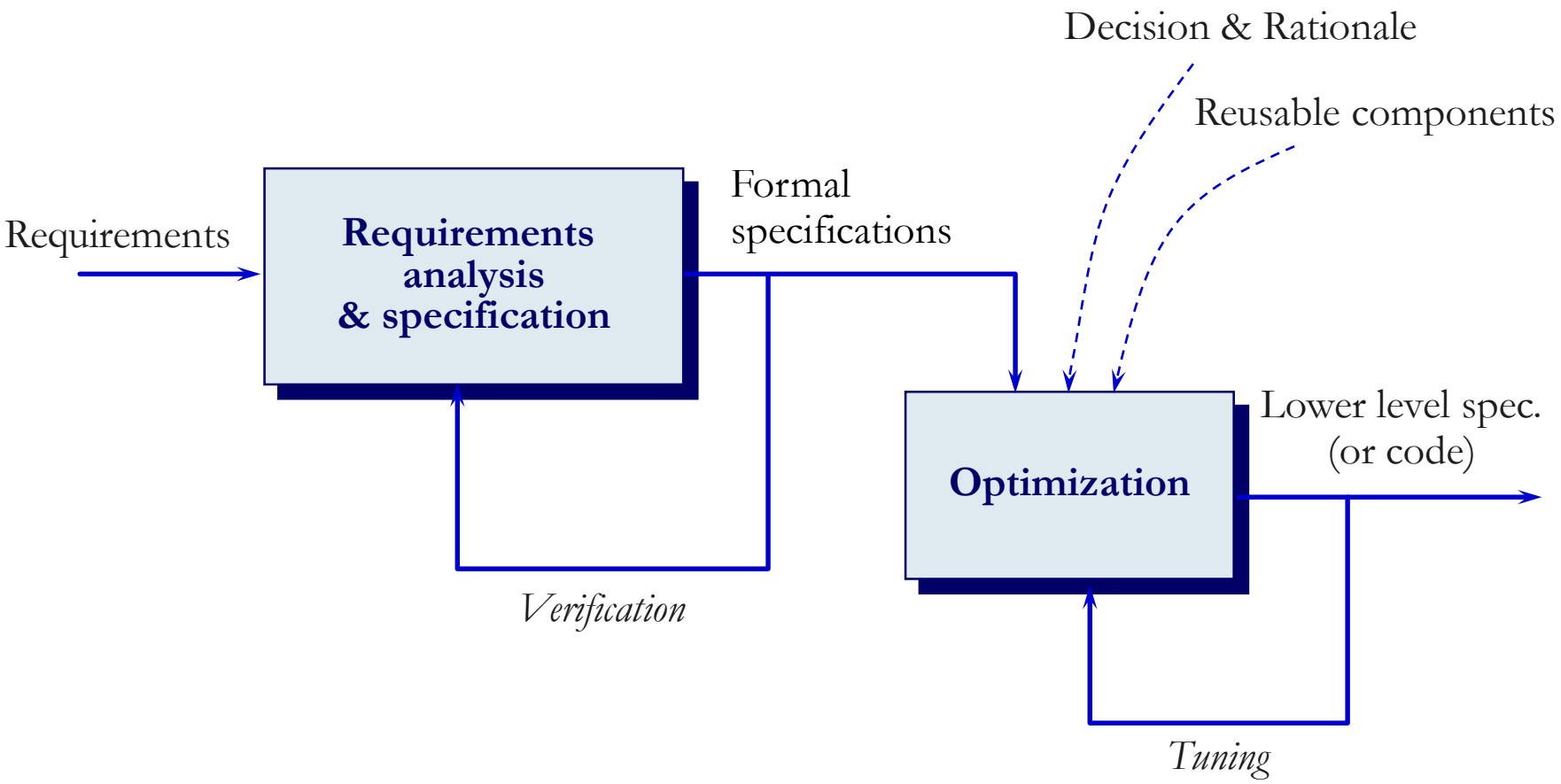


Agile Method – Scrum

Scrum Framework



Transformation Model



Transformation Model (Cont.)

Based on formal specification

- like Z, PetriNet, StateCharts, SDL …

Viewed as a sequence of steps that gradually transform a spec. into an implementation

Manually and automatically

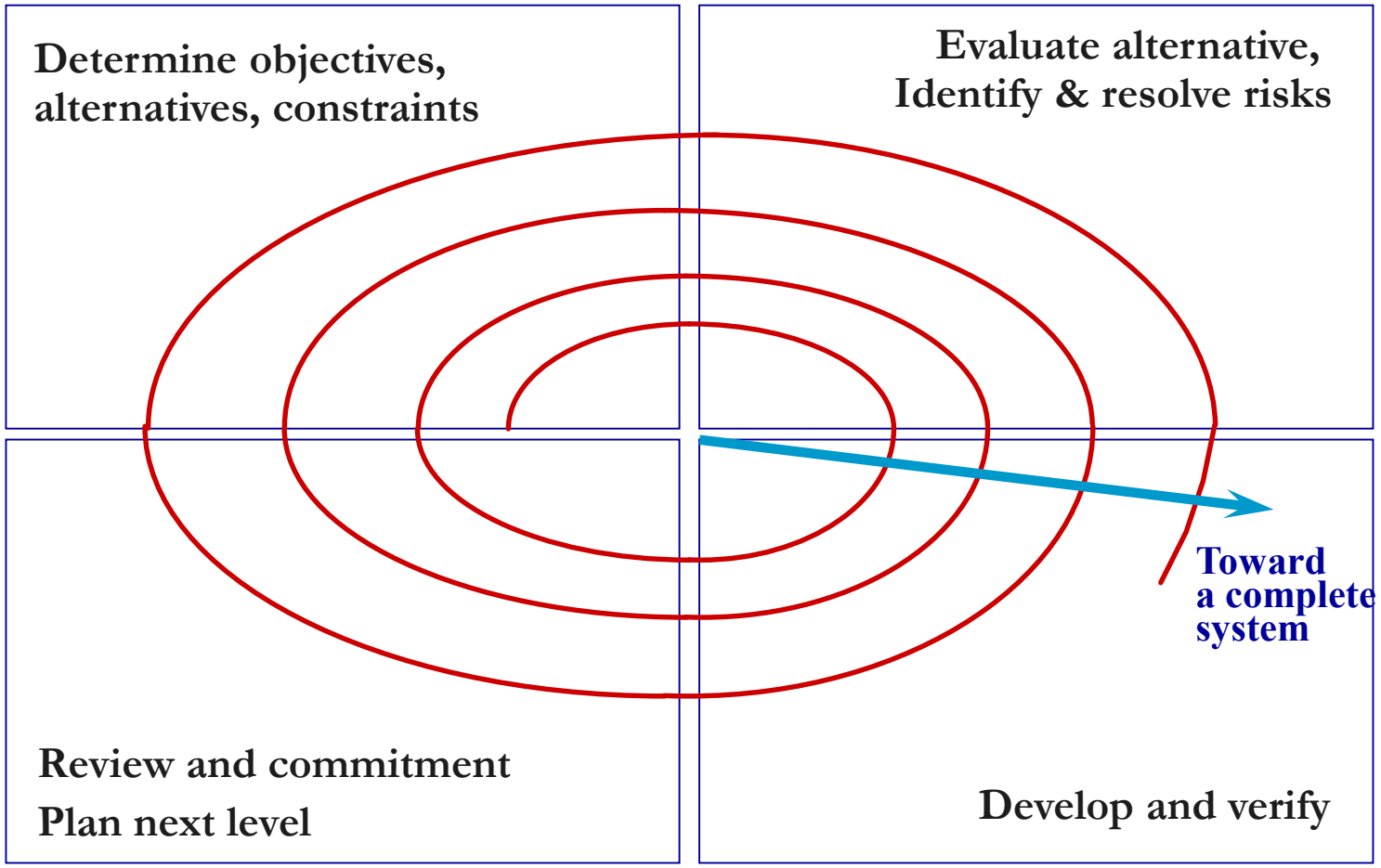
One of research-oriented approach (maybe)

Used for program correctness proof

Problems

- Need expert knowledge
- Narrow application in industrial use

Spiral Model



Spiral Model (Cont.)

By B. Boehm

Meta model

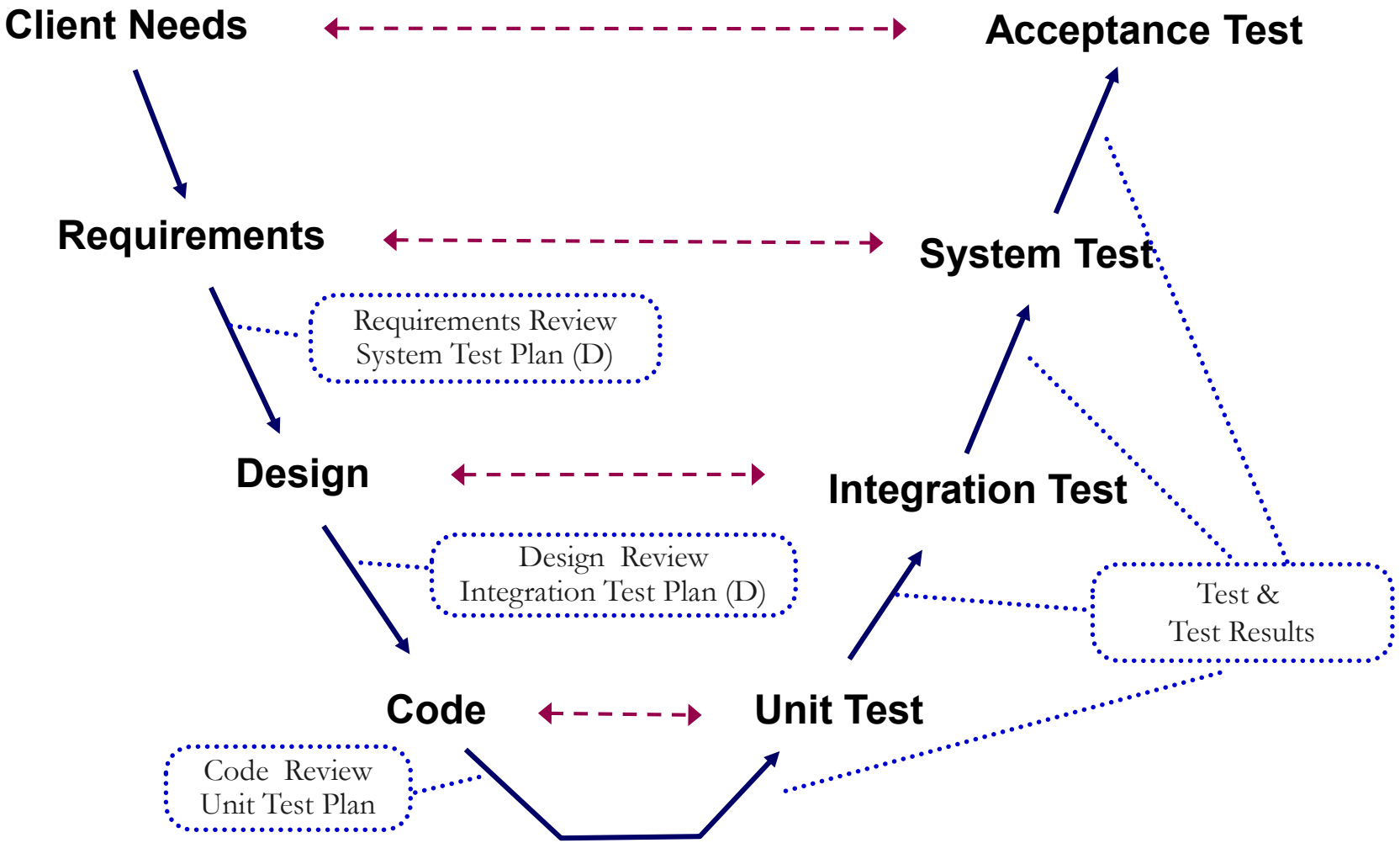
- Provide a framework for designing the software production process
- Guided by the risk levels in the project at hand

Focus on identifying and eliminating high-risk problems by careful process design

Problems

- May cost too much due to the risk analysis at each spiral
- Restriction on the range of its applicability
- Intended exclusively for development of large scale software

V Model


Client Needs
Acceptance Test
Requirements
System Test
Requirements Review
System Test Plan (D)
Design
Integration Test
Design Review
Integration Test Plan (D)
Test &
Test Results
Code
Unit Test
Code Review
Unit Test Plan

V Model (cont'd)

Known as Verification and Validation model

Extension of the waterfall model and is based on association of a testing phase for each corresponding development stage

Pros (+) and Cons (-)

- + : Highly disciplined model and Phases are completed one at a time.
- + : Works well for smaller projects where requirements are very well understood.
- + : Simple and easy to understand and use
- – : Not suitable for the projects where requirements are at a moderate to high risk of changing.
- – : Not a good model for complex and object-oriented projects.

CBSE Process

Component-Based Software Engineering Process

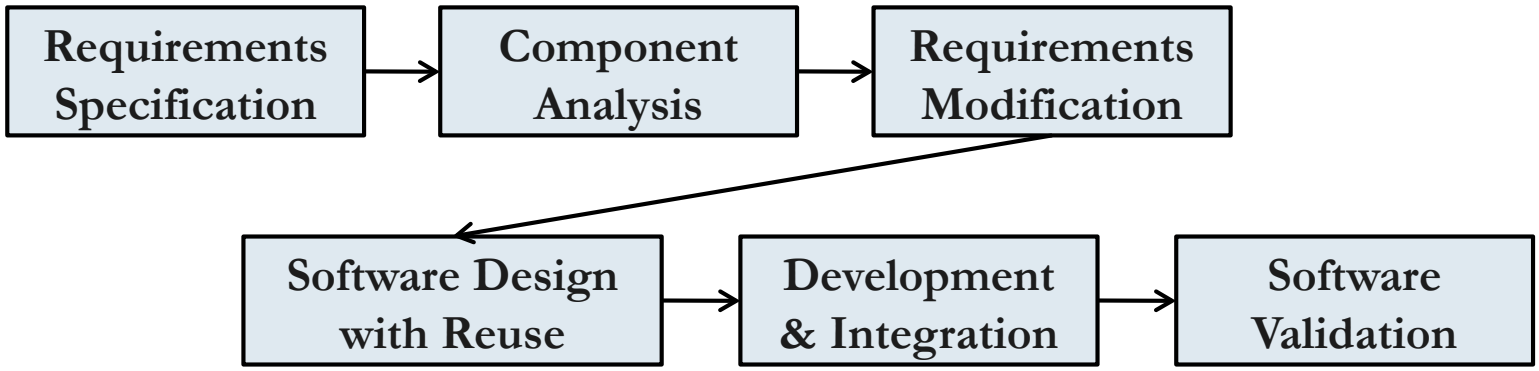
First prominent with Douglas Mcllroy, 1968

- Mass produced software components

Modern concept of a software component by Brad Cox, 1986

- Software ICs

Development with CBSE process



DevOps = Development + Operations

A philosophy and practice focused on agility, collaboration, and automation within IT and development team processes.

Traditionally, software development

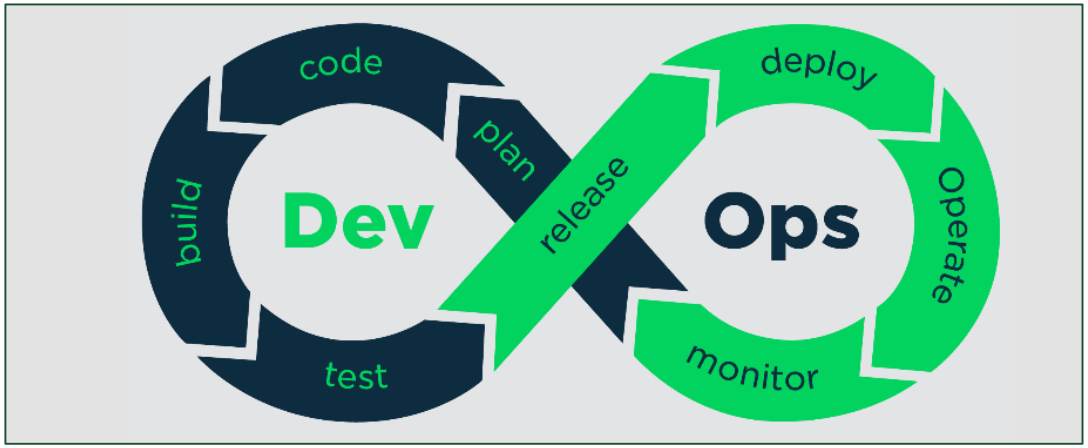
- Silos approach
- working independently within their own teams and processes.
- environment rife with miscommunication, poor alignment, and production delays ("War Room").

DevOps Goal

- bridge the gap between IT operations and development
- improve communication and collaboration
- more seamless processes, and align strategy and objectives
- faster and more efficient delivery.

What is DevOps?

Development-to-Operations Lifecycle of DevOps



By SUSE

- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Micro-Service Architecture
- Code-based Infrastructure
- Monitoring & Logging

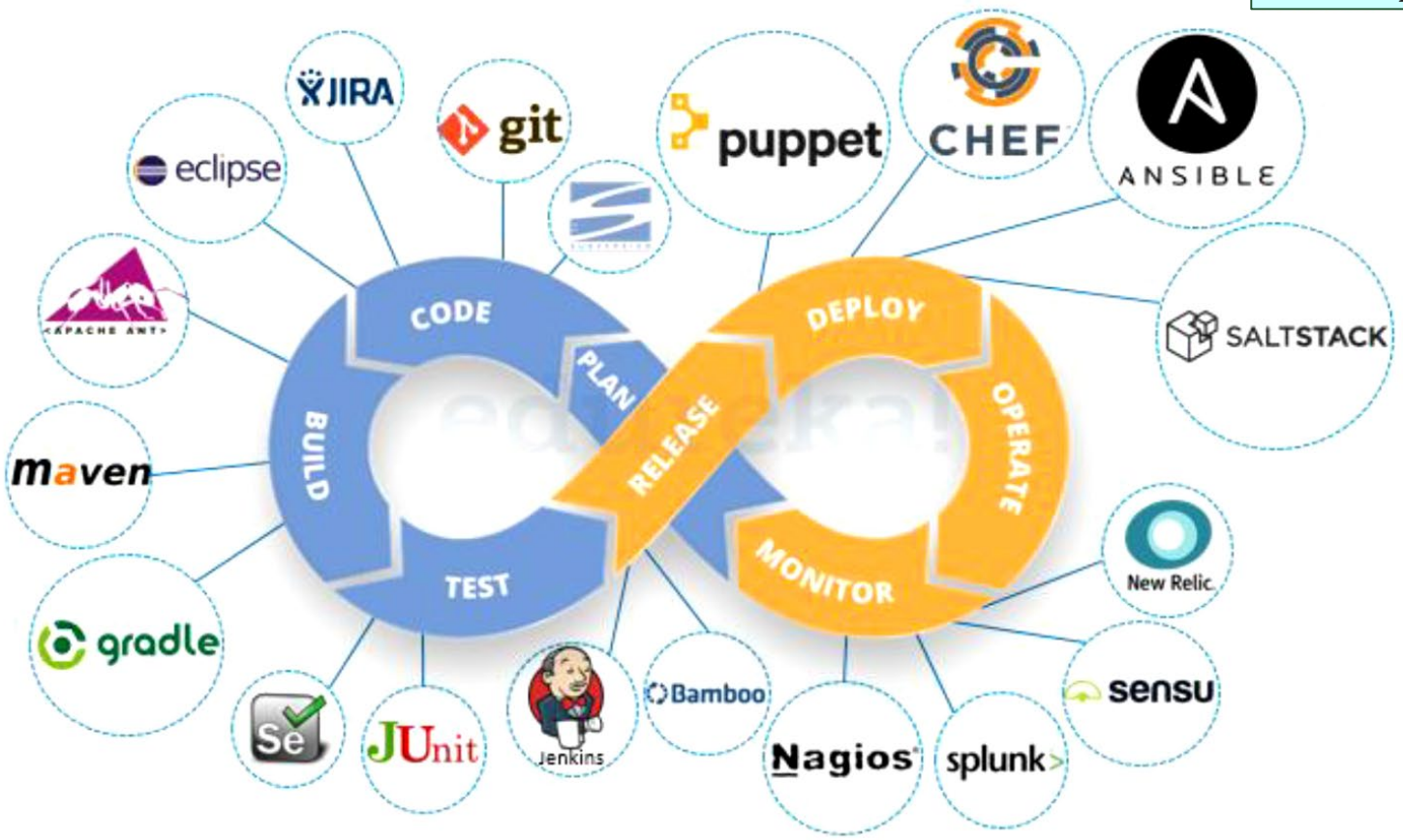
Benefits

- Fast Dev. & Fast Release
- Test Automation
- Fast and Easy Upgrade
- Strengthen Collaboration
- Secure Process

Toolchain in DevOps

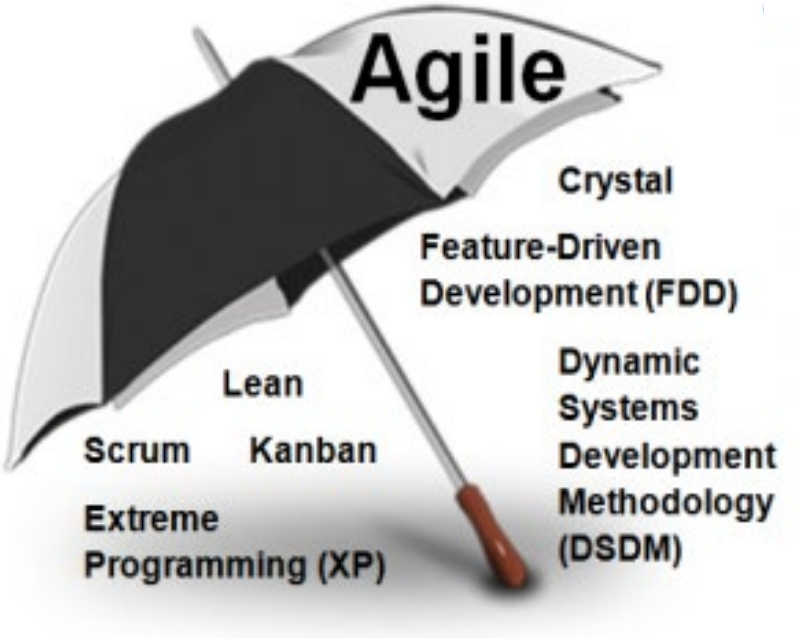
Automation is one of key...

Example



Other Process Models

and MORE … ?!



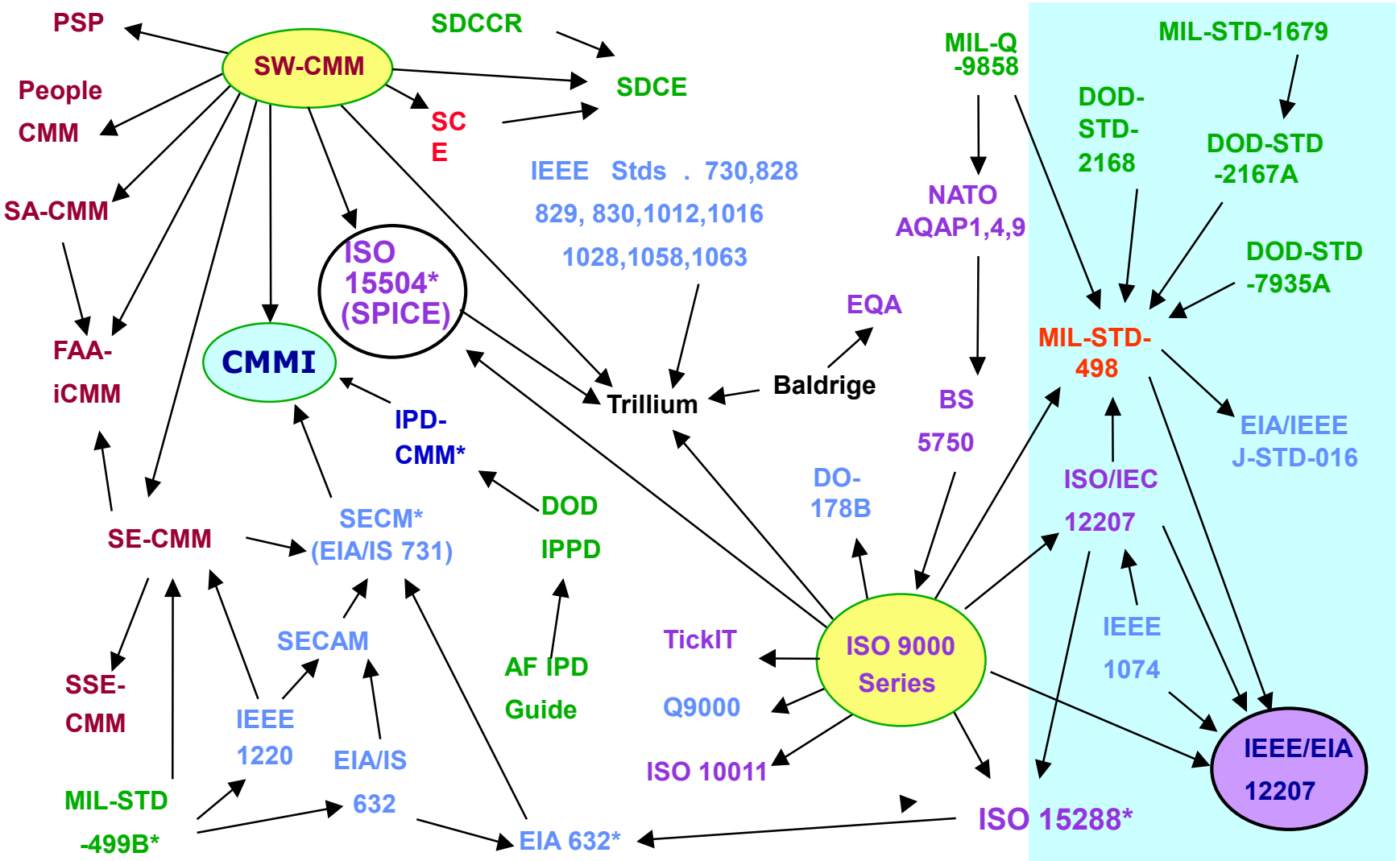
SDLC by Project Characteristics

Characteristics	Waterfall	Protyping	Spiral	Incremental	Iterative	Agile
Large scale			●	●	●	
Lots of Risks		●	●	●		
Ambiguous Reqts.		●		●		●
Long-Term			●	●	●	
Sufficient Budget			●	●		
Easy Technology	●					●
High Correctness		●	●	●		●
Customer Involvement						●

Major Output Documents

Phase	Documents	Components	Remarks
Project planning	**Project Management Plan**	- Management Issues (cost, schedule, resources, etc) - Quality Management Plan - Configuration management Plan	
Requirements Gathering	**RDD (REQD) (Requirements Description Doc.)**	- System description - Functional requirements - Non-functional Requirements	
Requirements Analysis	**SRS (Software Req. Specification)**	- Functional Analysis Model - Data Model	
Design	**SDD (Software Design Description)**	- Preliminary Design - Detailed Design (data, interface,..) - Deployment	Separated in some case.
Implementation	**Source Code List**	- Code	
Testing	**Test Plan** **Test Results**	- Test goal (Exit Criteria), Test Case - Report for Test run	

Standards for Software Process



www.software.org/quagmire

ISO/IEC/IEEE 12207

Purpose

- To establish a common framework for the life cycle of software:
  - To acquire, supply, develop, operate & maintain software
  - To manage, control, and improve the framework

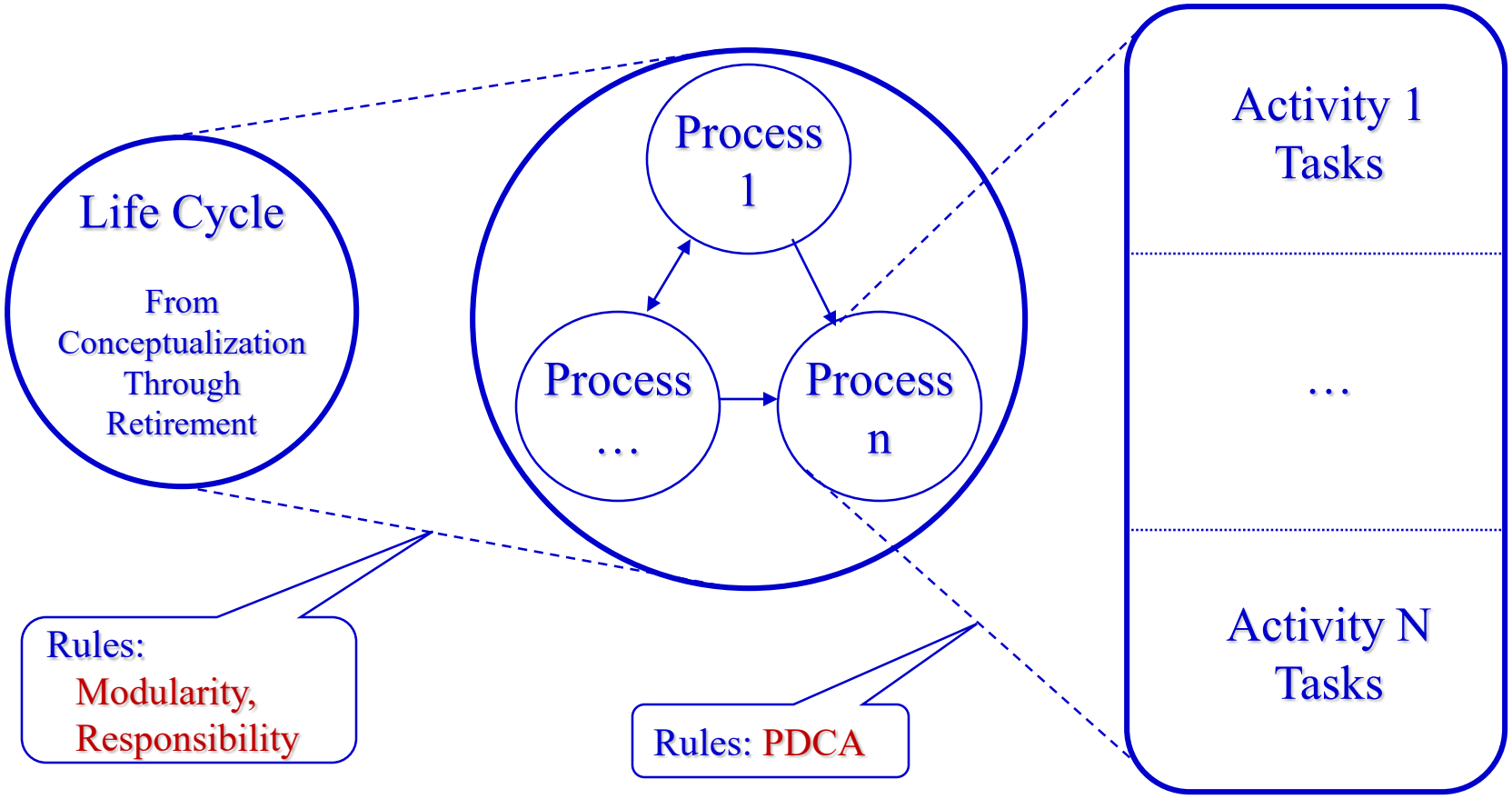
History

- Proposed in June 1988
- 4 Working Drafts; 2 Committee Drafts; 1 DIS
- Over 6 years and 17000 person-hours expended
- Published 1 August 1995
- New version in Nov. 2017

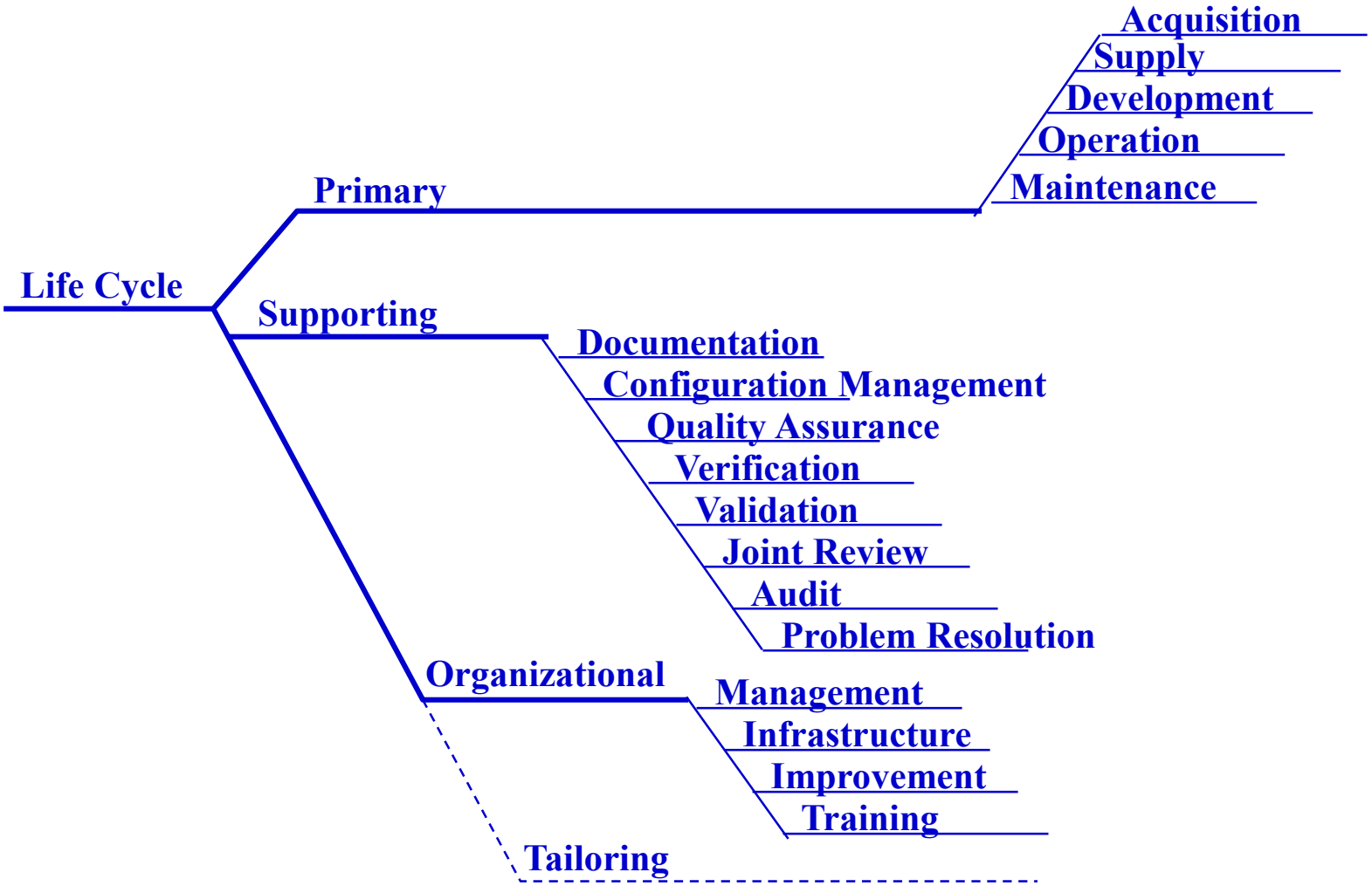
Participants

- Australia, Canada, Denmark, Finland, France, Germany, Ireland, Italy, Japan, Korea, Netherlands, Spain, Sweden, UK, USA

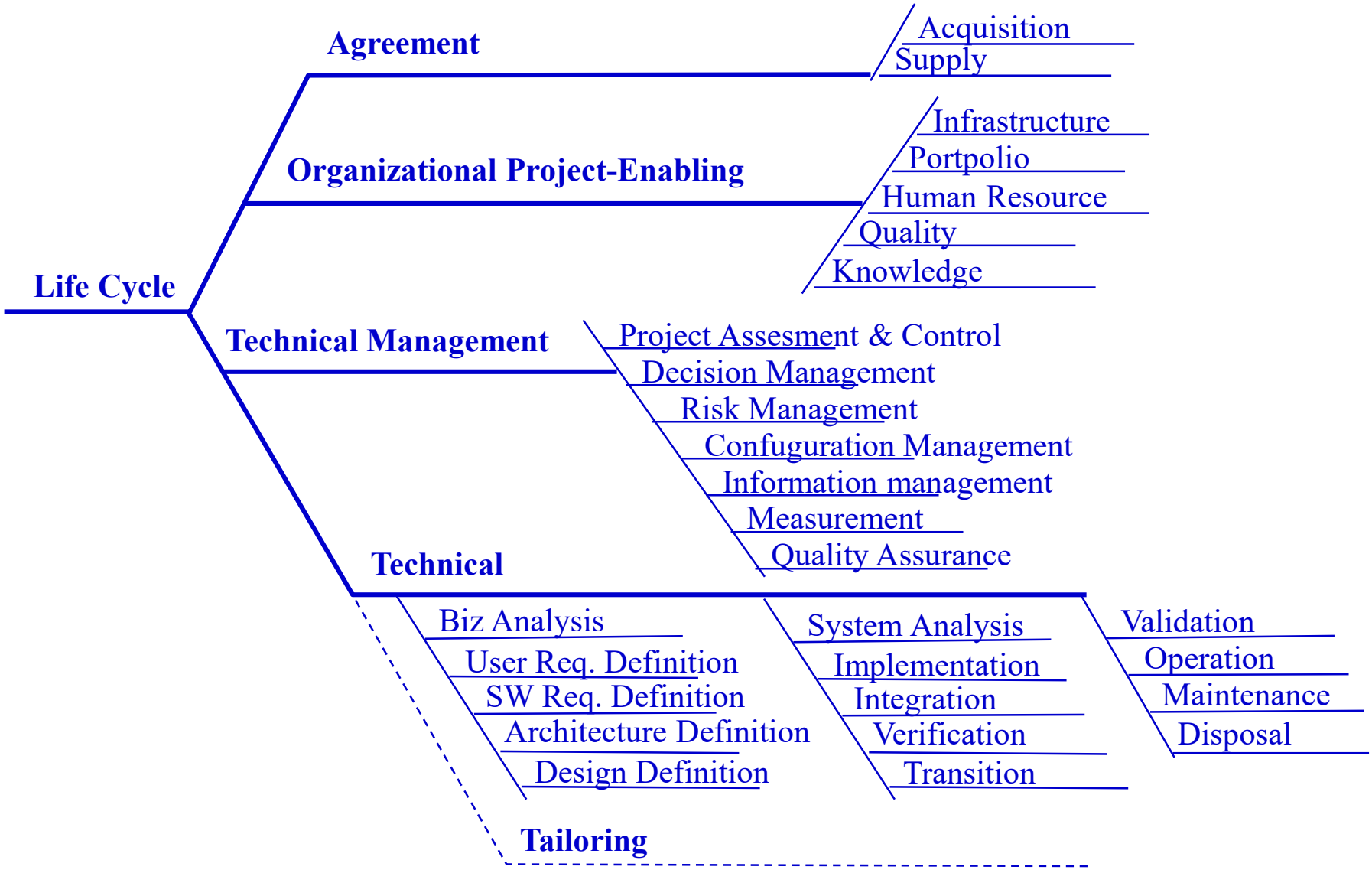
Basic Concepts - Architecture



Basic Concepts – Processes (1995)



Basic Concepts – Processes (2017)



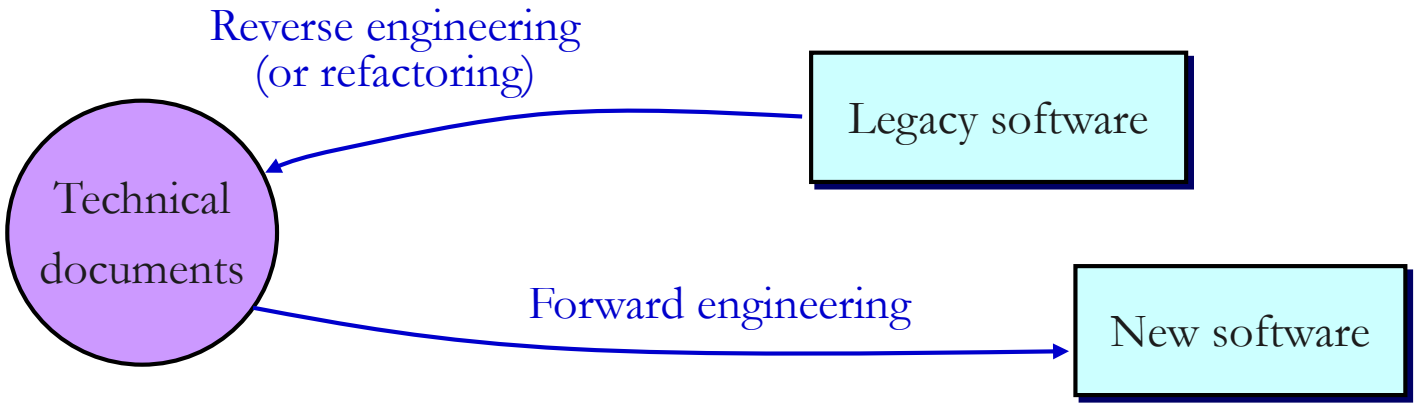
Dealing with Legacy Software

Motivation

- Not possible to develop new software from scratch
  - Huge investment in developing existing software
- Legacy : Asset to preserve very carefully before closing down

Reengineering

- Process through which an existing system undergoes an alternation, to be reconstituted in a new form



Summary and Discussion

Software (Production) Process Models

- Waterfall
- Evolutionary (Rapid prototyping, Incremental)
- Transformation
- Spiral Model
- Agile Model
- DevOps Model, ...

Why needed these process models ?

What is the difference between process models and methodology ?