Modellistica Medica

Maria Grazia Pia

INFN Genova

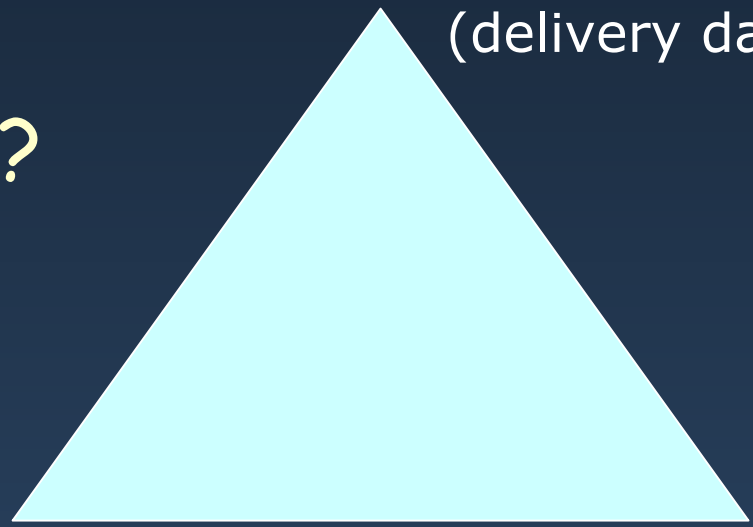
Scuola di Specializzazione in Fisica Sanitaria
Genova
Anno Accademico 2002-2003

Lezione 16-17

Introduction to software process

Software process models, standards

What do we want?



- A system that meets the user's needs
- Delivered on time
- Within budget
- Reliable, easily upgraded or enhanced with new features (maintainable)

No "silver bullet"…

- In other words, we will not improve software engineering an order of magnitude with new technologies
- Essentially, software is fundamentally difficult to develop, and that is unlikely to change
- The way to progress is to study and improve the way software is produced
  - better technology only helps once the organizational framework is set
  - there is evidence that going for new technology instead of improving the process can make things worst

So what *do* we do?

Improve the software process

The benefits of software process

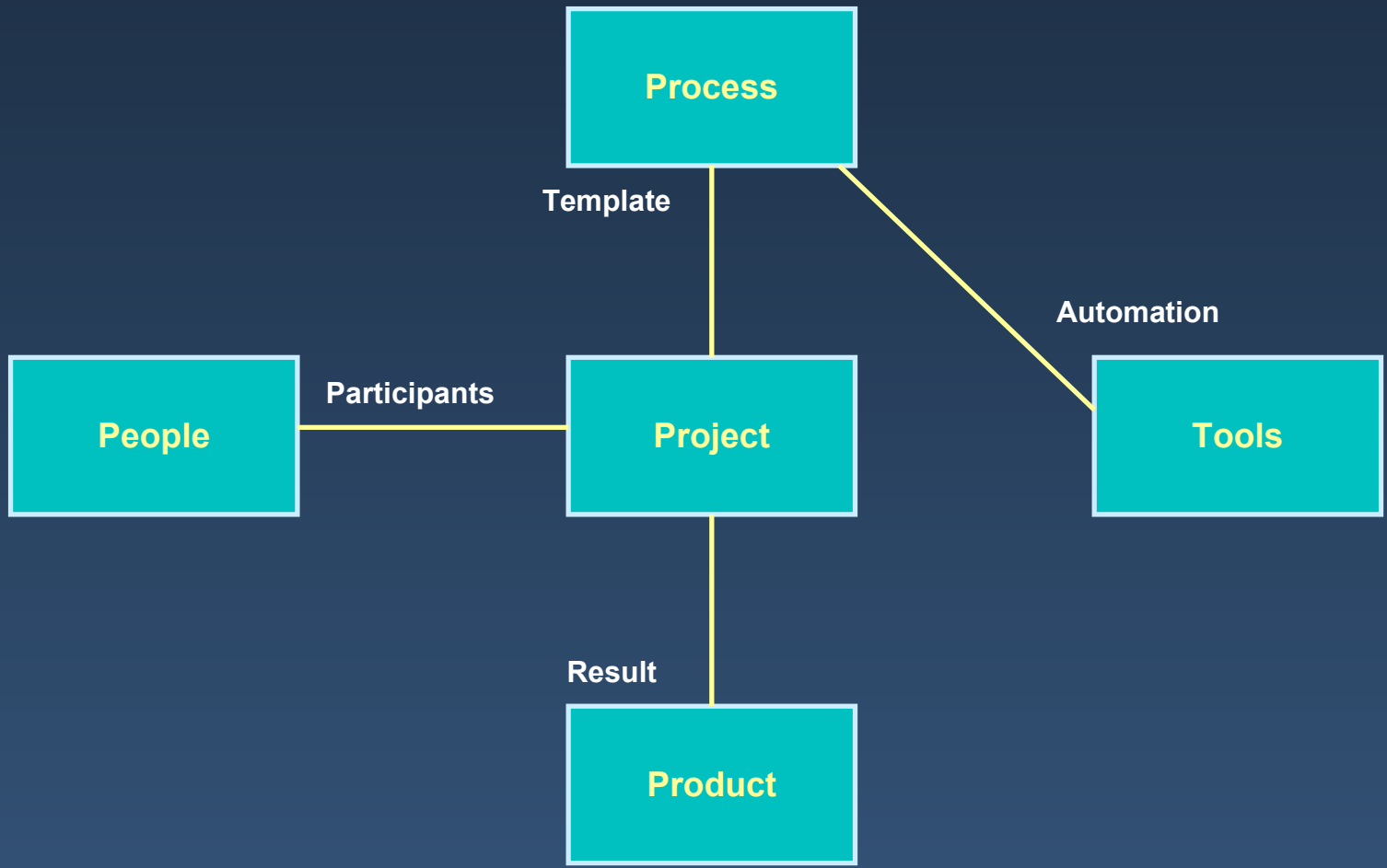
The goal: producing better software at lower cost, within predictable resource allocations and time estimates, and happier users of the software

Three key components:

- the **people** involved
- the organization of the development **process**
- the **technology** used

- The practices of SPI are well established, and have been applied in a large number of organizations for several years
  - *the results prove that the economical benefits are largely worth the investment*
  - *early defect detection, time to market, and quality also improve, to the point that the return on investment for SPI is about 500%*

Four P's of software development



Software life-cycle

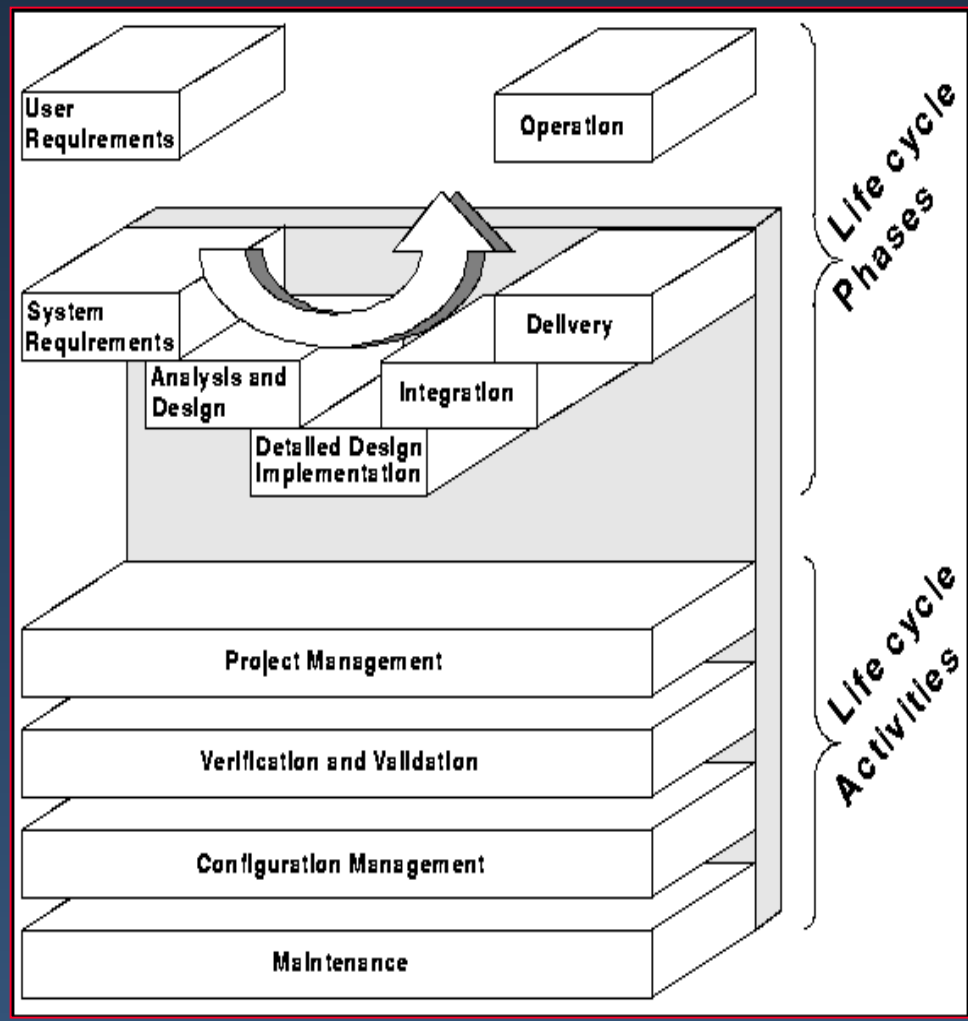
Various elements:

- User Requirements definition
- Software Requirements definition
- Architectural Design
- Detailed Design and construction
- Delivery to the user
- Operations

- Frequently the tasks of different life cycle phases are performed somewhat in parallel

  to consider them disjoint in time is a simplification

- It is however important
  - to distinguish them logically
  - to identify documents that are the outcome of the various phases



What is a Process Model?

- Simplified description of a set of activities/events/roles from a particular perspective
- It shows how something is done
- Could be a detailed schedule of events

- It does not refer to software only
  - building a car follows a process…

Process Principles

A software process

- provides guidance to a team's activities
- specifies which work products should be produced and when
- offers criteria for monitoring and measuring the project's products and activities

- Prescribes (guides) all major activities
- Uses resources, within a set of constraints, to produce intermediate and final products
- May be composed of sub-processes
- Each activity has entry and exit criteria
- Has a set of guiding principles to explain goals
- Constraints may apply to activity, resource or product

Software Lifecycle Models

- How software is created and evolves

- Software engineering standards (ISO/IEC, ESA…)
- Software process models (waterfall, spiral…)
- Software process frameworks
- Software capability evaluation models

Standards

Examples:

- ISO/IEC 15504: **I**nternational **S**tandardization **O**rganization
  - **I**nternational **E**lectrotechnical **C**ommission
- ESA PSS-05: Software Engineering Standards
  - **E**uropean **S**pace **A**gency

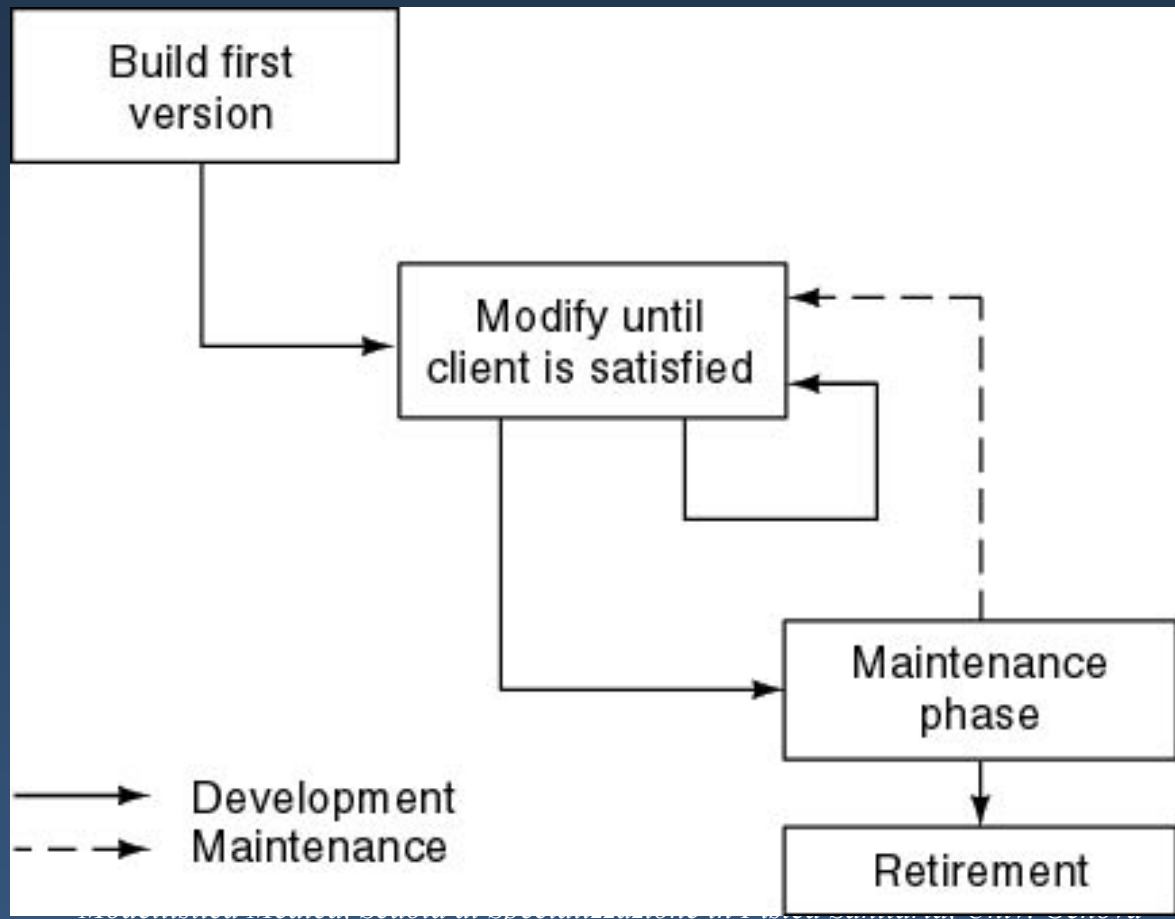
De facto "standards" (wide spread usage)

- CMM
- RUP

Overview of software process models

Build-and-fix model

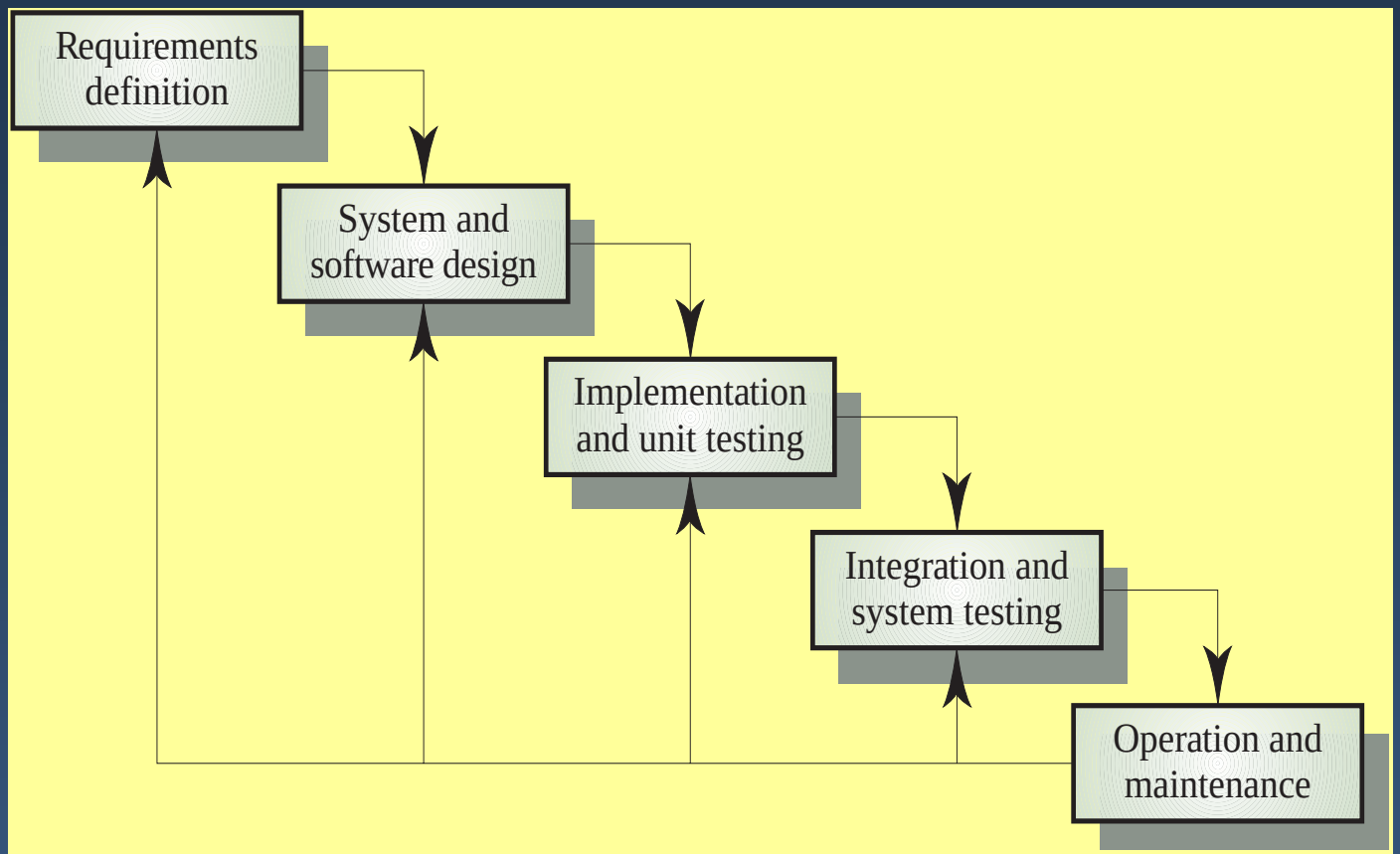
- The haphazard model; we all know it…
- OK on short programs
- Fails utterly on larger ones



The Waterfall Model

The **de-facto model** for many decades

- Earliest software process model
- Cascade of phases
  - the output of one is input to the next
- The choice of phases differs in various standards and organizations



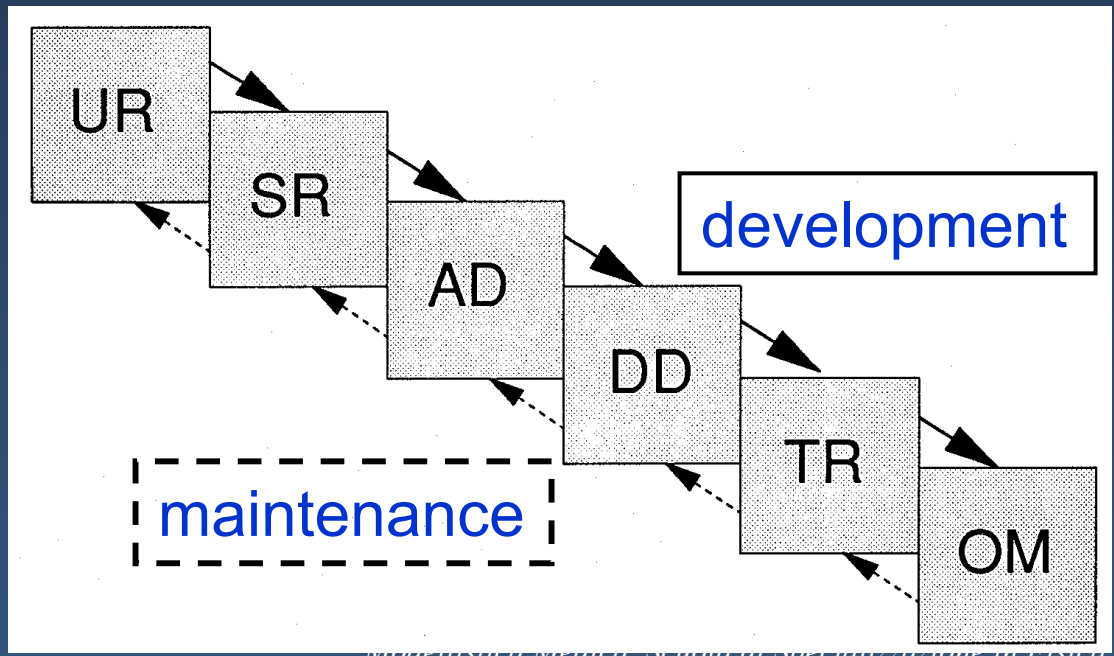
ESA PSS-05 Software Life Cycle Management

Activities:

- Software project management
- Software configuration management
- Software verification and validation
- Software quality assurance

Phases:

- UR - Definition of the user requirements
- SR - Definition of the software requirements
- AD - Definition of the architectural design
- DD - Detailed design and production of the code
- TR - Transfer of the software to operations
- OM - Operations and maintenance



Criticism

- Monolithic
- No backwards communication
- The drawback of the waterfall model is the difficulty of accommodating change after the process is underway
- It seems best-suited to solving well-understood problems
- Weak at addressing risks (at an early stage)

Plan to Throw One Away

"For most projects, the first system built is barely usable: too slow, too big, too hard to use, or all three.

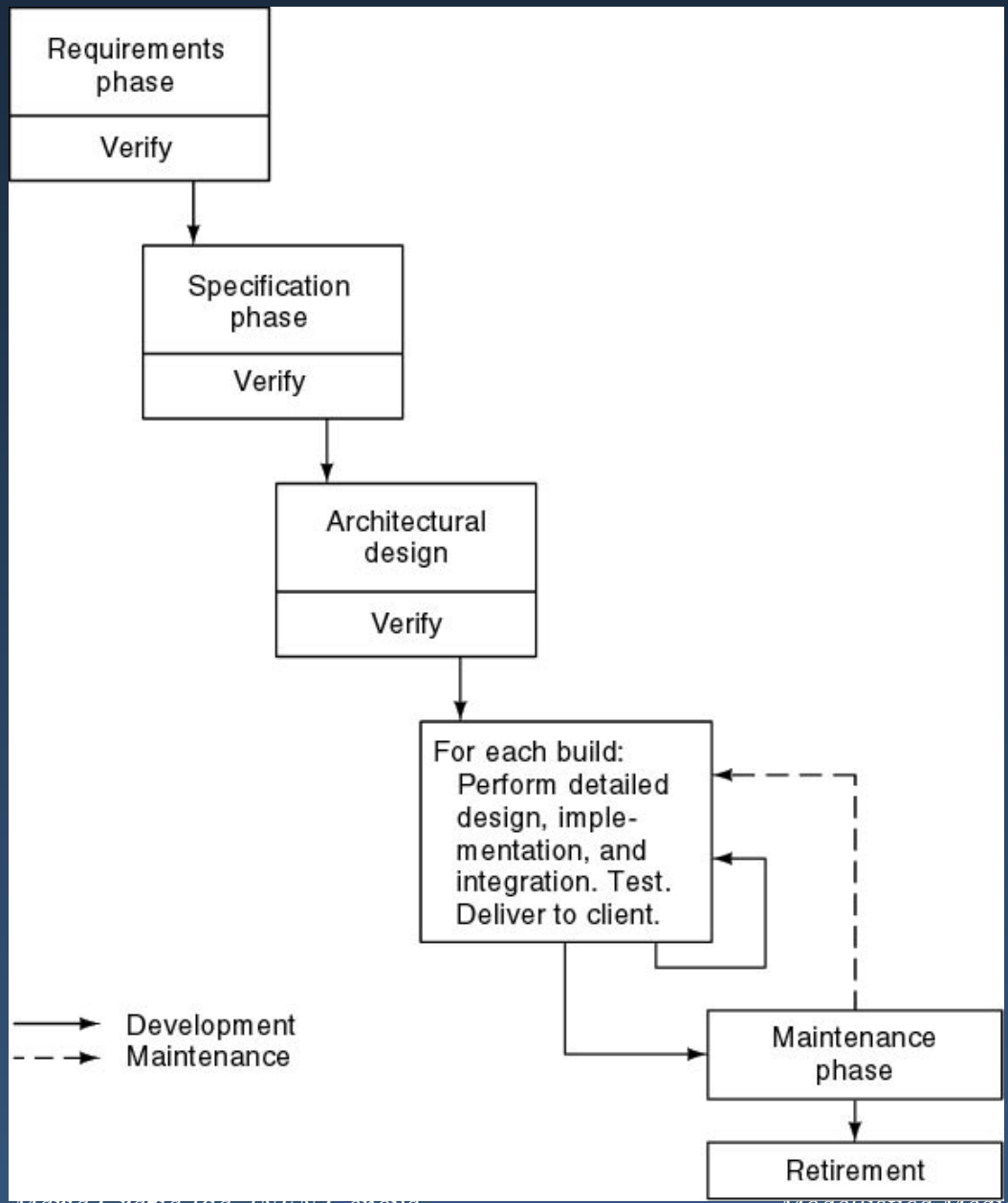
Plan to throw one away; you will, anyhow."

Fred Brooks The Mythical Man-Month

Alternatives

- **Incremental Model**
  - Staged delivery of components or functions
- **Evolutionary Model**
  - Multiple releases are provided *(well-suited to prototyping)*

Incremental DeliveryApproach



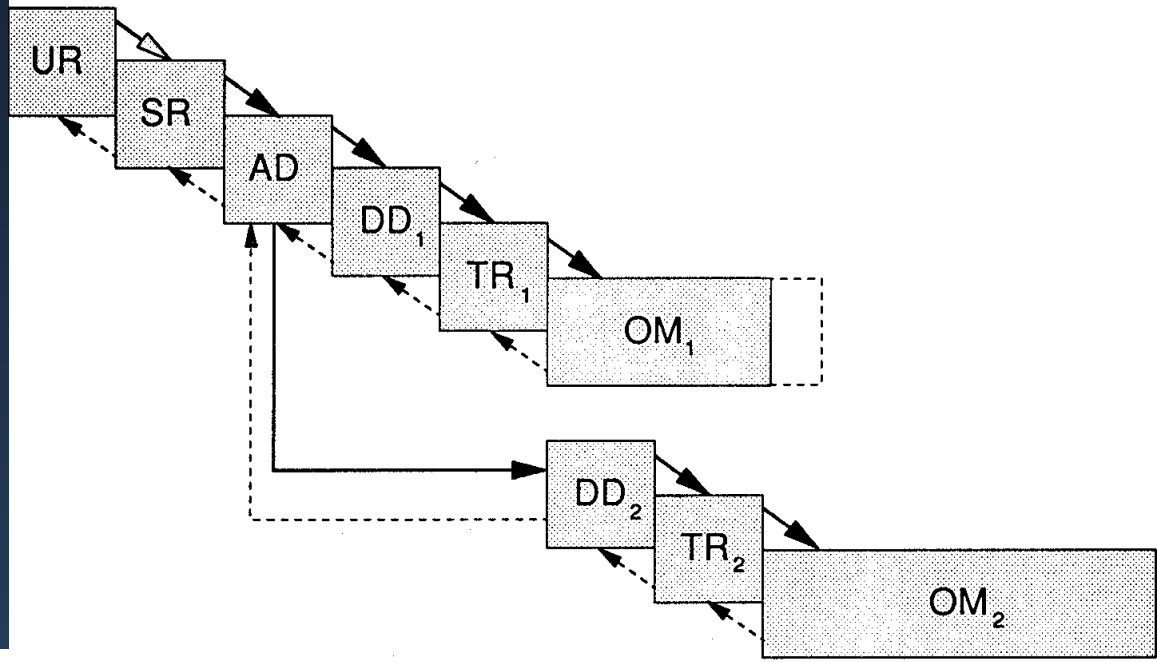
Deliver product piece-by-piece

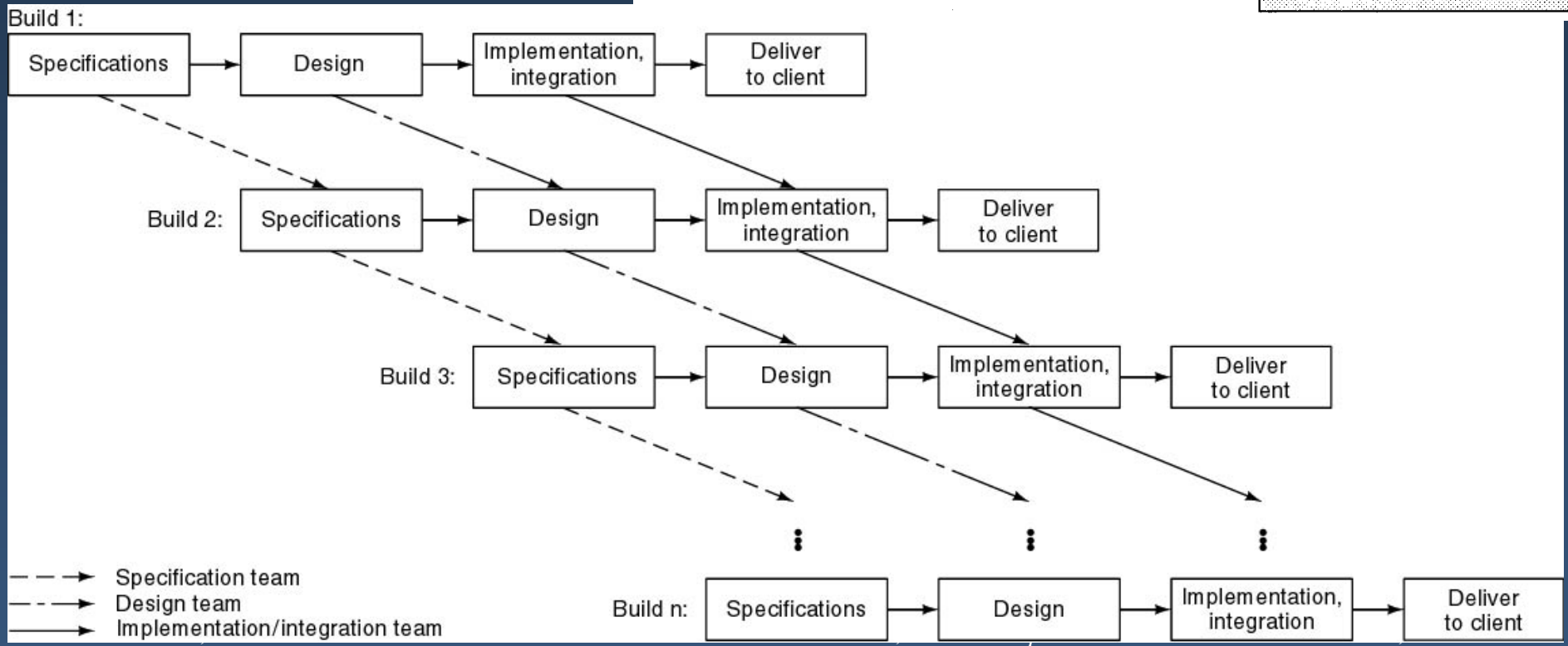
Happens often in small business

May easily degrade into build-and-fix

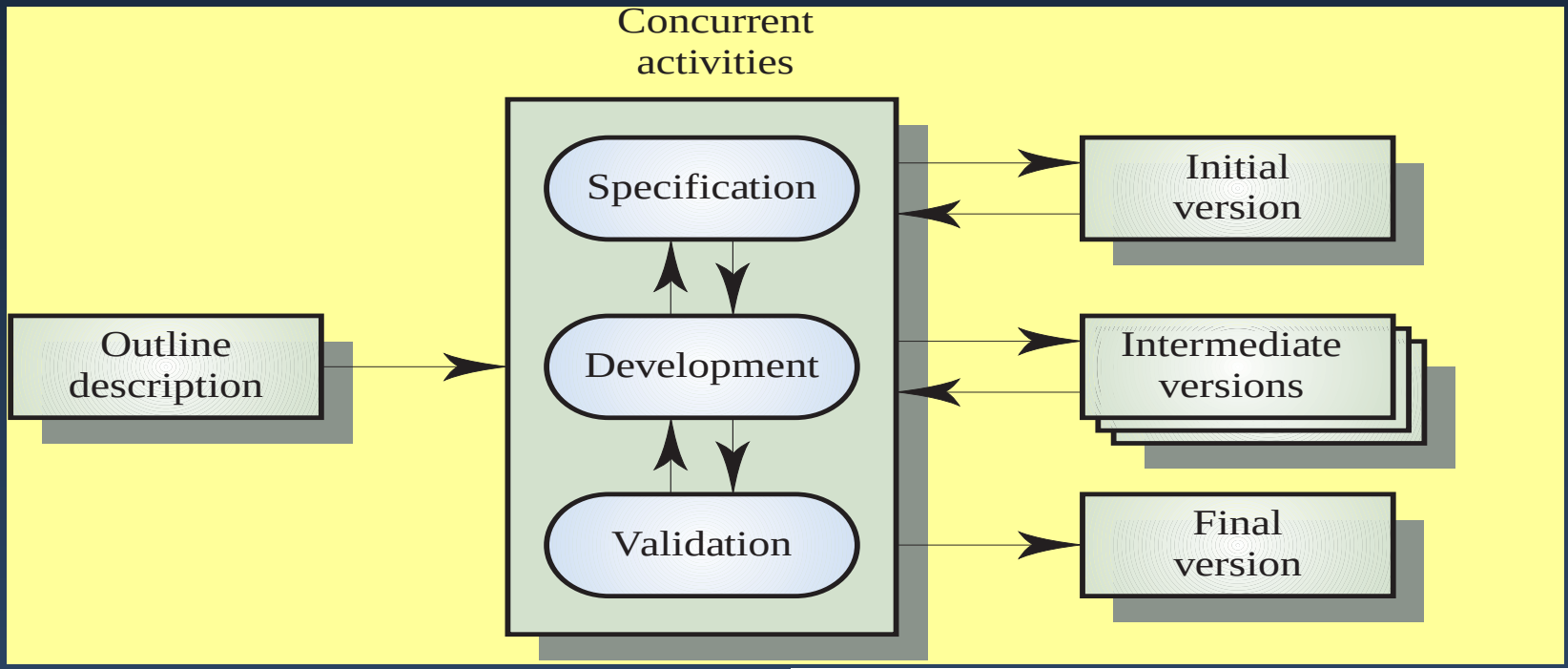
Incremental Approach

- Risky: CodeABitTestABit (CABTAB)
  - no guarantee builds will "fit together"

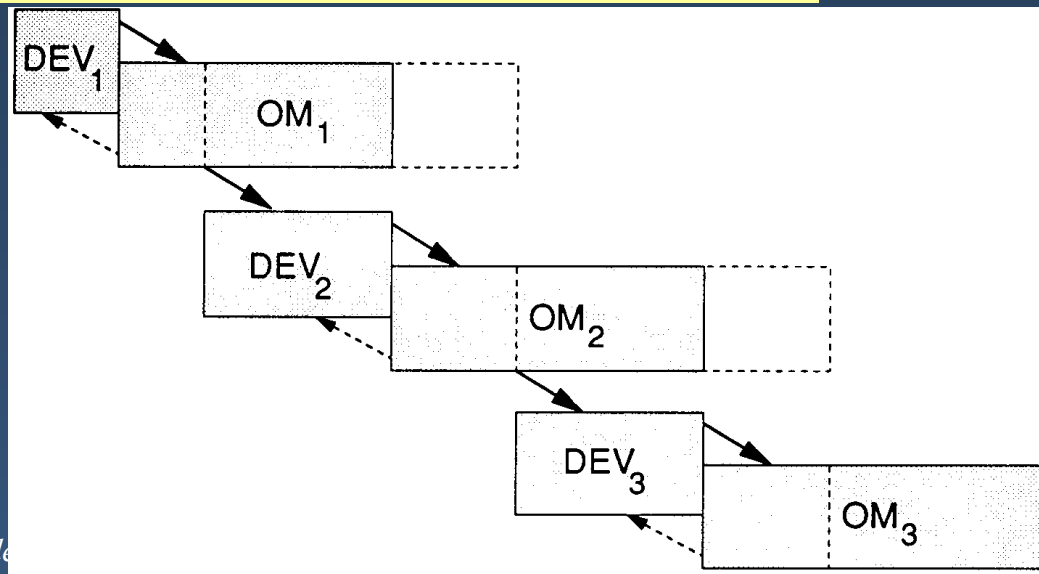




Evolutionary Design Approach

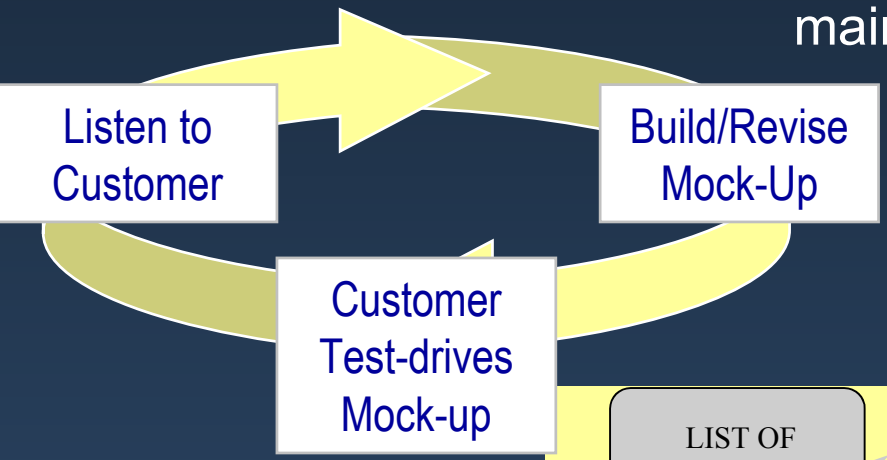


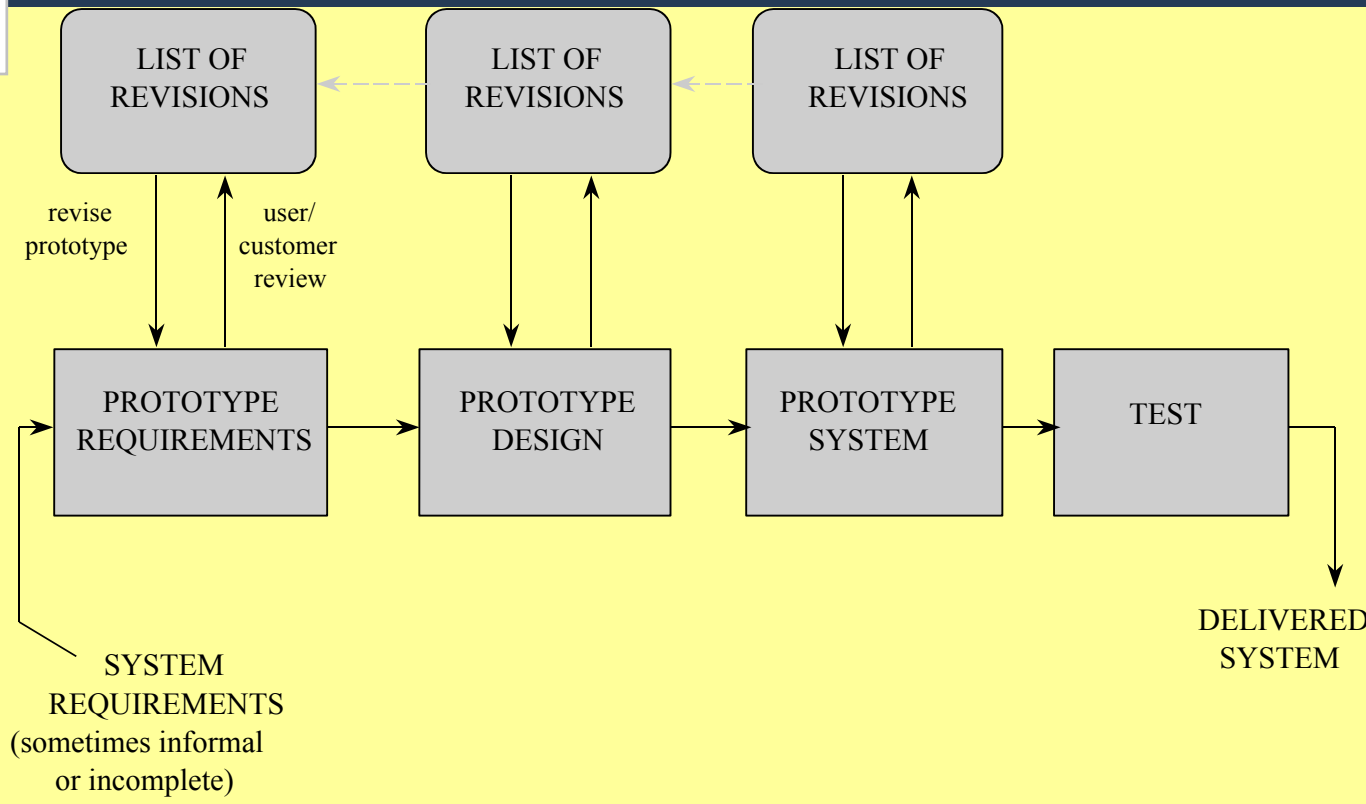
The DEV box is equivalent to the UR, SR, AD, DD, and TR phases



Prototyping

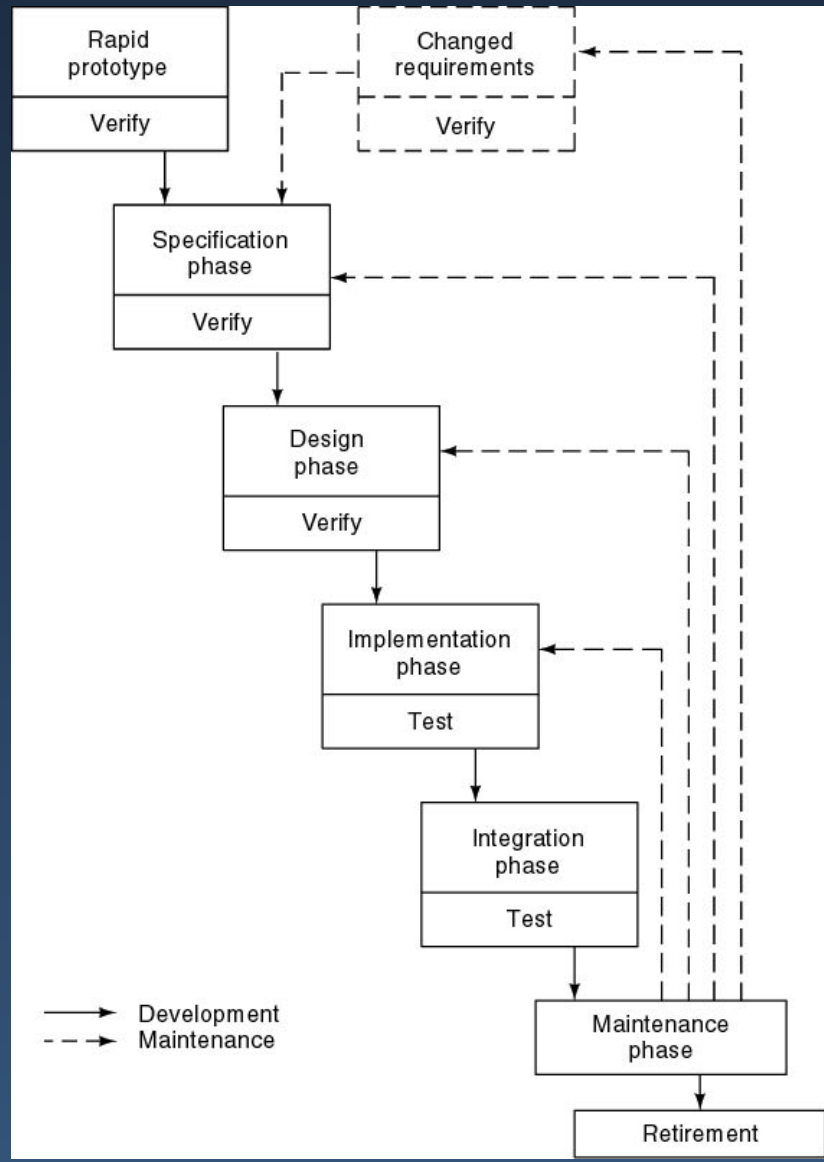
- Prototypes implement high risk functional, performance or user interface requirements
- They usually ignore quality, reliability, maintainability and safety requirements





Prototyping approaches

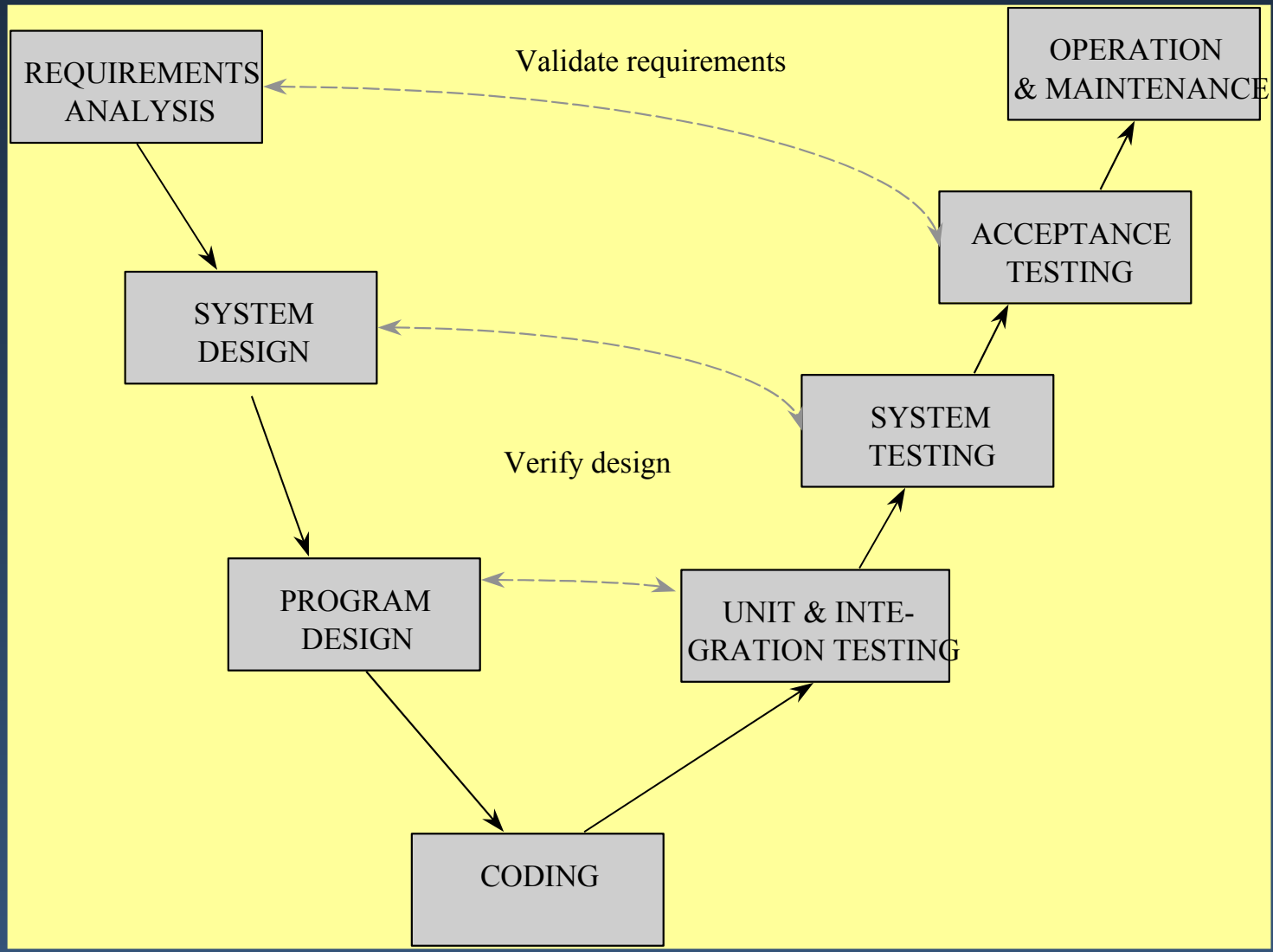
- Exploratory prototyping
  - Objective is to work with customers and to evolve a final system from an initial outline specification
  - Should start with well-understood requirements
- Throw-away prototyping
  - Objective is to understand the system requirements
  - Should start with poorly understood requirements



Evolutionary development

- Problems
  - Lack of process visibility
  - Systems are often poorly structured
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems
  - For parts of large systems (e.g. the user interface)
  - For short-lifetime systems

V Model


REQUIREMENTS ANALYSIS
Validate requirements
OPERATION & MAINTENANCE
ACCEPTANCE TESTING
SYSTEM DESIGN
SYSTEM TESTING
Verify design
PROGRAM DESIGN
UNIT & INTE-GRATION TESTING
CODING

Risk management

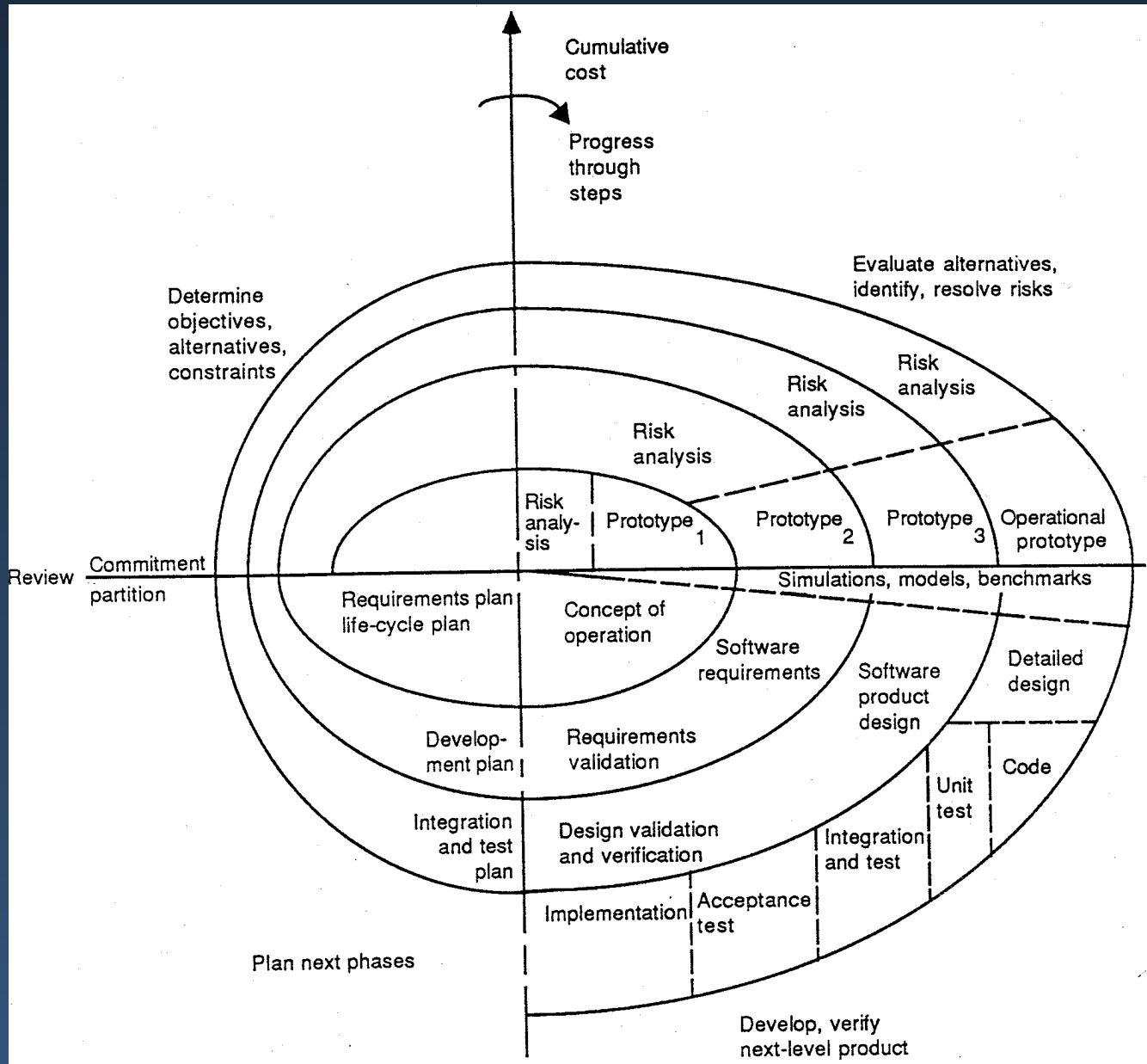
- Perhaps the principal task of a manager is to minimize risk
- The **risk** inherent in an activity is a measure of the uncertainty of the outcome of that activity
- High-risk activities cause schedule and cost overruns
- Risk is related to the amount and quality of available information
  - The less information, the higher the risk

Spiral model

- Identify risks
- Assign priorities to risks
- Develop a series of prototypes for the identified risks starting with the highest risk
- Use a waterfall model for each development (round or cycle)
- If a risk has successfully been resolved, evaluate the results of the round and plan the next round
- If a certain risk cannot be resolved, terminate the project immediately

Spiral Model

- Due to Barry Boehm (1988)
- Emphasizes risk management: has explicit risk analysis phases
- A non-linear view of the software life cycle
- It is a *meta-model* : it can incorporate the other models in its stages



Spiral model

- Concept of Operations
- Software Requirements
- Software Product Design
- Detailed Design
- Code
- Unit Test
- Integration and Test
- Acceptance Test
- Implementation

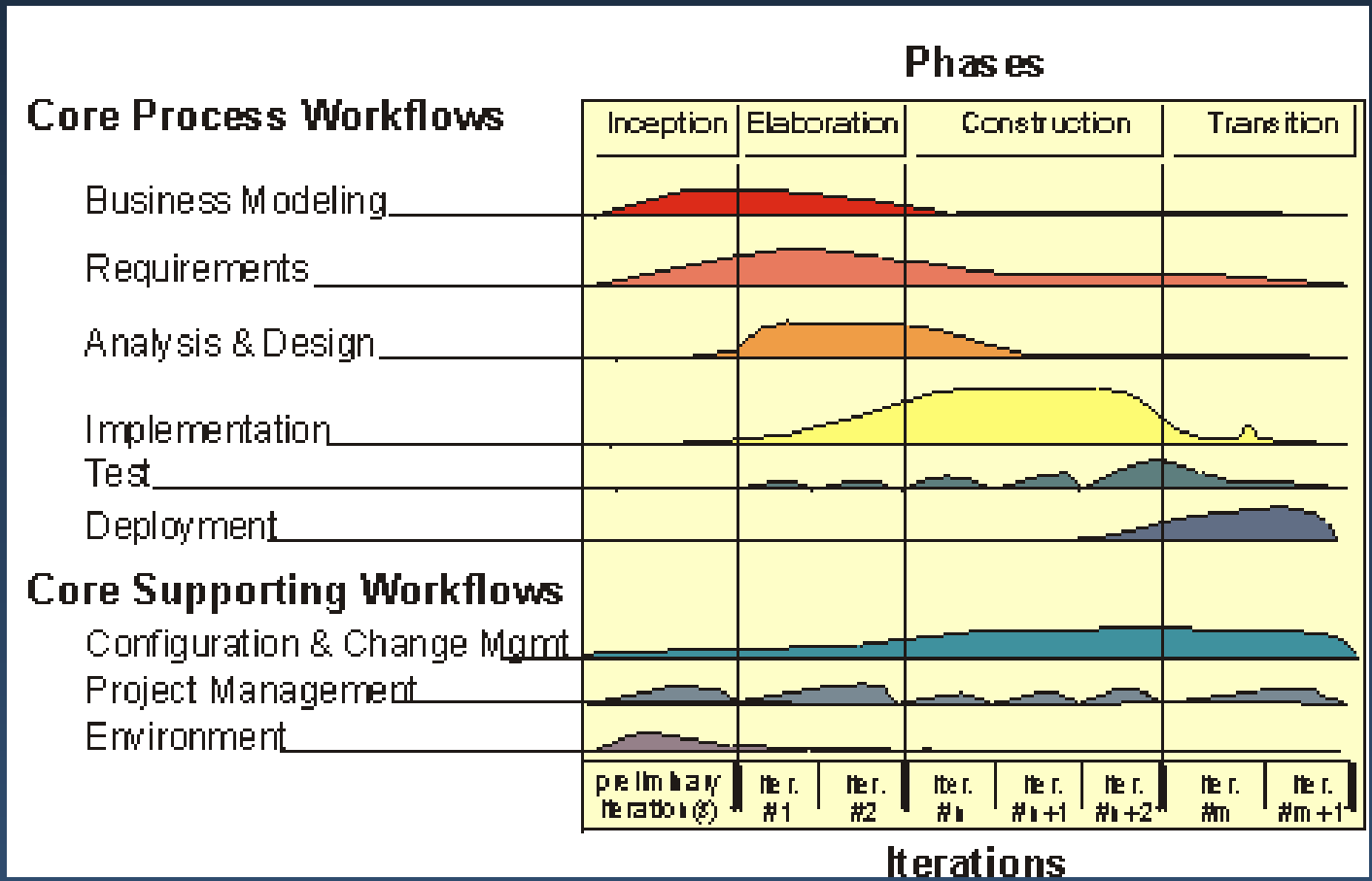
For each round go through the steps:

- Define objectives, alternatives, constraints
- Evaluate alternative, identify and resolve risks
- Develop, verify next level product
- Plan next activity (“round”, “phase”)

Phases of the spiral model

- Objective setting
  - Specific objectives for the project phase are identified
- Risk assessment and reduction
  - Key risks are identified, analyzed and information is sought to reduce these risks
- Development and validation
  - An appropriate model is chosen for the next phase of development
  -
- Planning
  - The project is reviewed and plans drawn up for the next round of the spiral

Unified Process – USDP/RUP



Illustrated in detail in a following lecture

What is Extreme Programming

Things we know from Software Engineering:

Software Eng. Practice	*XP Principles*	
Code Reviews are good	Review code all the time	Pair Programming
Testing is good	Everybody tests all the time	Unit and functional tests
Design is good	Part of daily business	Refactoring
Simplicity is good	Always have the simplest design that does the job	Simplest design that works
Architecture is important	Everybody works in refining architecture	Use of a metaphor
Integration Testing is important	Continuously integrate and test	Continuous Integration
Short Iterations are good	Make iterations really short	The Planning Game

Extreme programming model

- Somewhat controversial new model
- Very much a religion amongst proponents
- Might be good for rapidly changing designs

- Break build down to small tasks to be done in parallel
- Software team determines features (“stories”) client wants, do cost analysis on it (requirements/specification)
- Successive builds, client determines features/stories in each build
- Draw up test cases for task
- Code task in pairs
- Unusual properties:
  - Computers in center of room
  - Client with team at all times
  - No two-week successive overtime
  - No specialization
  - No overall design phase; *refactor* design during builds
- XP has not yet been proven for larger projects; easy to abuse

Process Questions

- How good are you or your organization at developing software?
- Are you doing all the right things?
- Are there other process guidelines/suggestions?
- Is there help getting started?

CMM: Background

- Developed by Software Engineering Institute (SEI), Carnegie Mellon University
- Funded by U.S. DoD, concerned with late and cancelled contracts
- Capability Maturity Model for Software (SW-CMM) v1.1, February 1993
- Capability Maturity Model - Integrated (CMMI-SW/SE) v1.0, August 2000
- For large projects, DoD contractors must be assessed at Level 3 or DoD equivalent (October 1999)

CMM – Capability Maturity Model

Capability = Organization's ability to manage processes and *control cost and schedule*

Depends on *best practices*

Maturity = Level or degree of control over cost and schedule.

From *ad hoc* to *optimizing*

Software Process: CMM

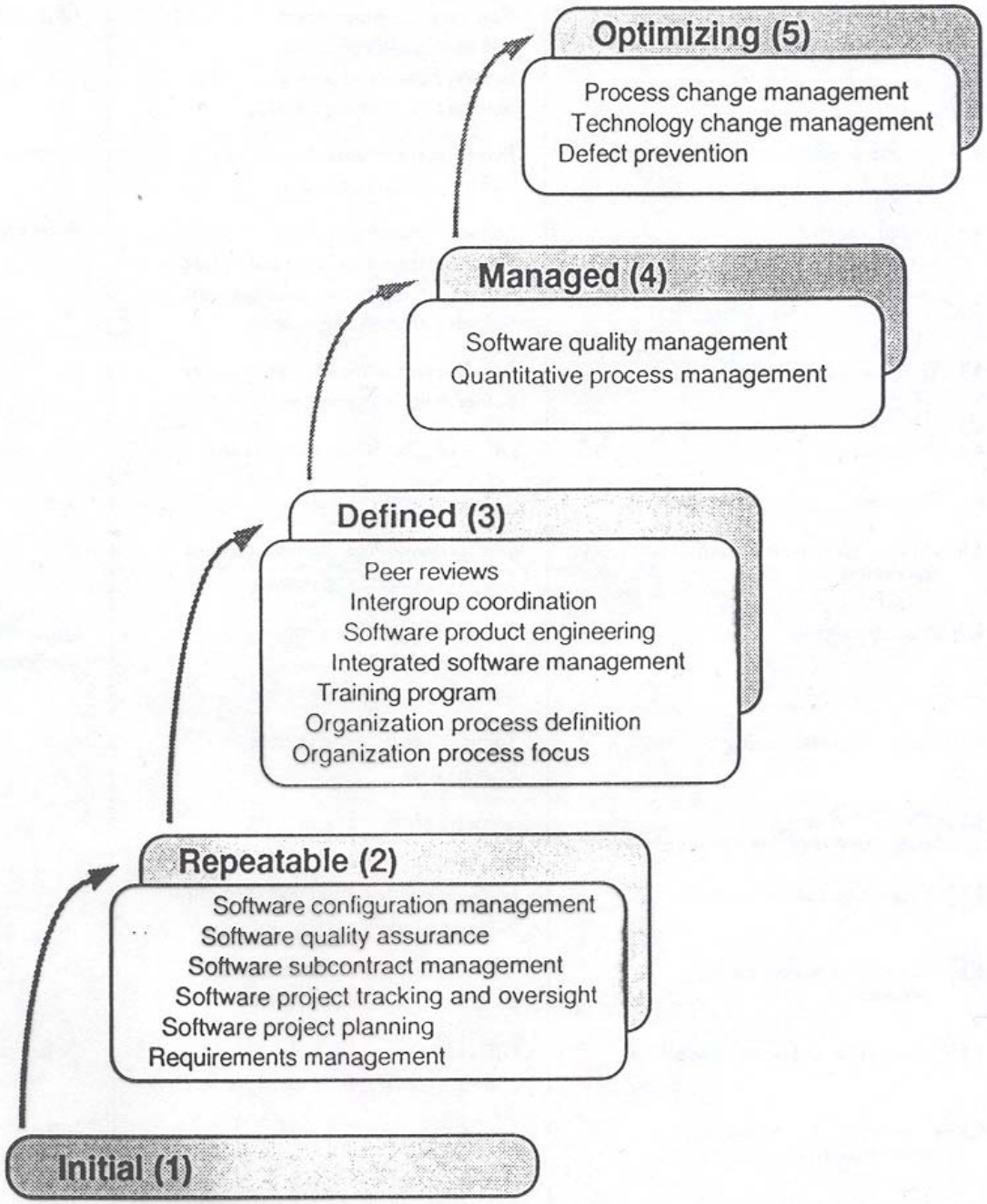
1. **Initial.** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

2. **Repeatable.** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

3. **Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

4. **Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5. **Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.



CMM: Maturity Levels

5. **Optimizing.** Continuous process *improvement*

4. **Managed.** Detailed *measures* of the software process and product quality are collected

3. **Defined.** Management and engineering activities are *documented*, standardized, institutionalized

2. **Repeatable.** Basic project management tracks *cost, schedule, and functionality*. Successes can be repeated for similar projects

1. **Initial.** *Ad hoc*. Success depends on individual effort and heroics

Capability Levels

Level 5 - Optimizing	Process is changed and adapted to meet business objectives
Level 4 - **Quantitatively Managed**	Process is controlled using statistical and quantitative techniques
Level 3 - Defined	Process is standardized – applied to all projects
Level 2 - Managed	Process is planned, documented, monitored, controlled
Level 1 - Performed	SPs are satisfied
Level 0 - Incomplete	Process not implemented or SPs not satisfied

Software Process: SPICE

- Software Process Improvement and Capability dEtermination (ISO 15540)
- ISO standard (in progress)

Primary		Organizational		Supporting
Customer-Supplier	Engineering	Management	Organization	Support
CUS.1 Acquisition	ENG.1 Development	MAN.1 Management	ORG.1 Organizational alignment	SUP.1 Documentation
CUS.1.1 Acquisition Preparation	ENG.1.1 System requirements analysis and design	MAN.2 Project Management	ORG.2 Improvement	SUP.2 Configuration Management
CUS.1.2 Supplier Selection	ENG.1.2 Software requirements analysis	MAN.3 Quality Management	ORG.2.1 Process establishment	SUP.3 Quality Assurance
CUS.1.3 Supplier Monitoring	ENG.1.3 Software design	MAN.4 Risk Management	ORG.2.2 Process assessment	SUP.4 Verification
CUS.1.4 Customer Acceptance	ENG.1.4 Software construction		ORG.2.3 Process improvement	SUP.5 Validation
CUS.2 Supply	ENG.1.5 Software integration		ORG.3 Human Resource Management	SUP.6 Joint Review
CUS.3 Requirements Elicitation	ENG.1.6 Software testing		ORG.4 Infrastructure	SUP.7 Audit
CUS.4 Operation	ENG.1.7 System integration and testing		ORG.5 Measurement	SUP.8 Problem Resolution
CUS.4.1 Operational Use	ENG.2 System and software maintenance		ORG.6 Reuse	
CUS.4.2 Customer Support				

Software Process: SPICE

5 process categories

40 processes
sub-processes

SPICE

For a given process to achieve a *Level*, all lower level attributes must be *Fully* met and all attributes at the target level must be *Fully* or *Largely* met

Level 5 - Optimizing (process optimized to meet business needs; new ideas /technologies are trialed; processes are adapted and dynamically changed under control)	PA 5.1 Process Change attribute PA 5.2 Continuous Improvement attribute
Level 4 - Predictable (defined process control limits exist and process performed within them consistently; measures of performance are collected/analyzed)	PA 4.1 Measurement attribute PA 4.2 Process Control attribute
Level 3 - Established (defined, documented process used with tailoring where applicable)	PA 3.1 Process Definition attribute PA 3.2 Process Resource attribute
Level 2 - Managed (evidence of planning and tracking; work product requirements specified and met, including quality, schedule, and resource needs)	PA 2.1 Performance Management attribute PA 2.2 Work Product Management attribute
Level 1 - Performed (generally achieve goals of the process though not rigorously planned or controlled; basic work products exist as evidence of performance)	PA 1.1 Performed attribute
Level 0 - Incomplete (general failure to meet goals of the process; few or no outputs/work products)	No attributes

Exercise for Wednesday 9 April

- The Therac-25 case
- Google search: Therac-25
  - optionally select type .pdf or .ps to get the original IEEE article
- Exercise: case study