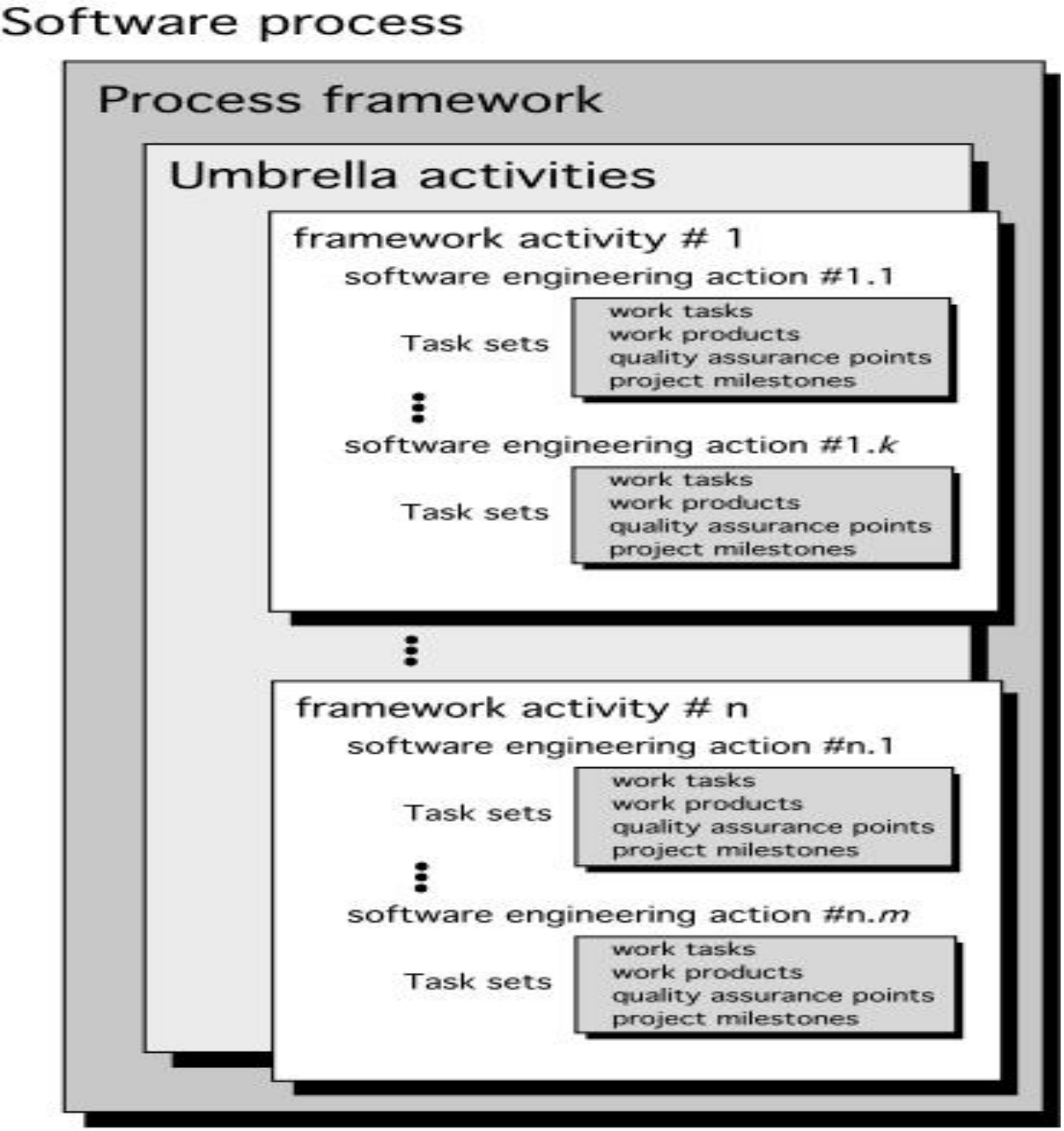
Process Models

Adapted from Pressman and Schach

A Generic Process Model



Process

Collection of work activities, actions and tasks

Process model

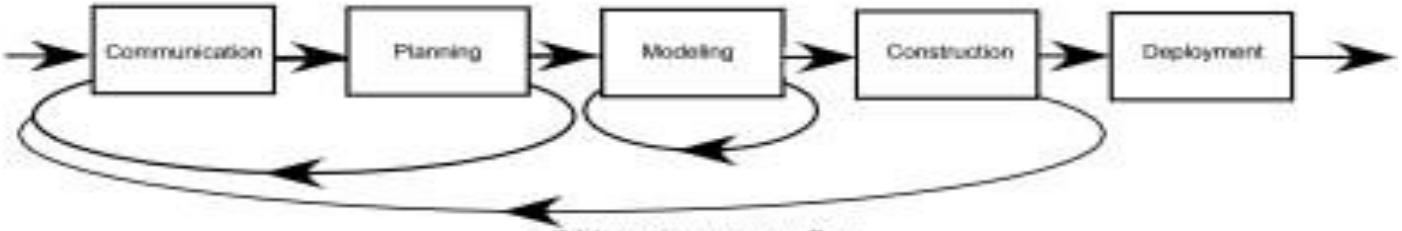
Activities, actions and tasks reside in a framework or model that defines their relationship with the process and with one another

Umbrella activities

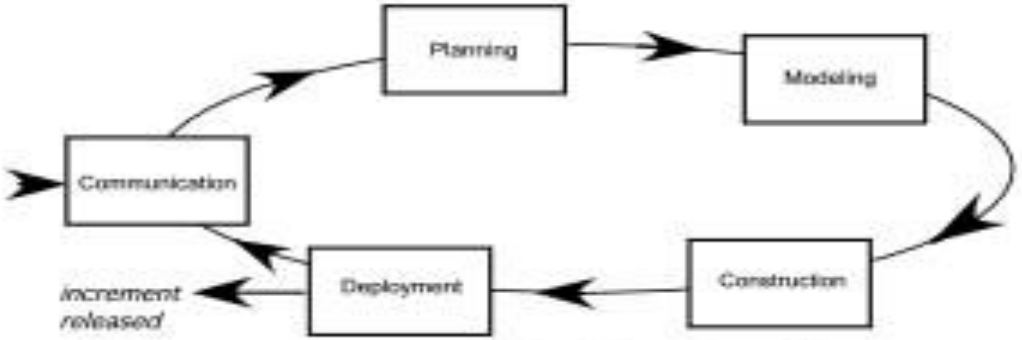
Project tracking & control
Risk management
QA
Configuration mngt
Tech. reviews

Process Flow

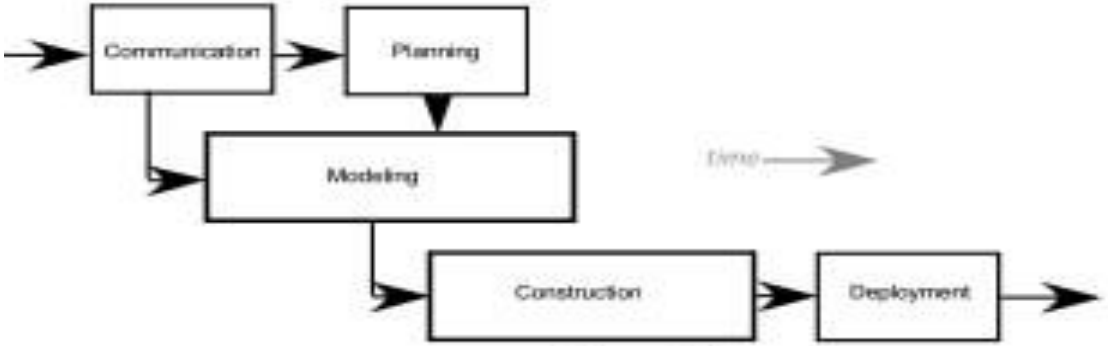
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

Communication

- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation

Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied

Process Patterns

- A *process pattern*
  - describes a process-related problem that is encountered during software engineering work,
  - identifies the environment in which the problem has been encountered, and
  - suggests one or more proven solutions to the problem.
- Stated in more general terms, a process pattern provides you with a *template* [Amb98]—a consistent method for describing problem solutions within the context of the software process.

Process Pattern Types

- *Stage patterns*—defines a problem associated with a framework activity for the process.
  - Eg. Establishing communication
- *Task patterns*—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice
  - Eg. Requirements gathering
- *Phase patterns*—define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature.
  - Eg. Spiral model or prototyping

Process Assessment and Improvement

- **Standard CMMI Assessment Method for Process Improvement (SCAMPI)** — provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting and learning.
- **CMM-Based Appraisal for Internal Process Improvement (CBA IPI)**—provides a diagnostic technique for assessing the relative maturity of a software organization; uses the SEI CMM as the basis for the assessment [Dun01]
- **SPICE—The SPICE (ISO/IEC15504)** standard defines a set of requirements for software process assessment. The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined software process. [ISO08]
- **ISO 9001:2000 for Software—**a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. Therefore, the standard is directly applicable to software organizations and companies. [Ant06]

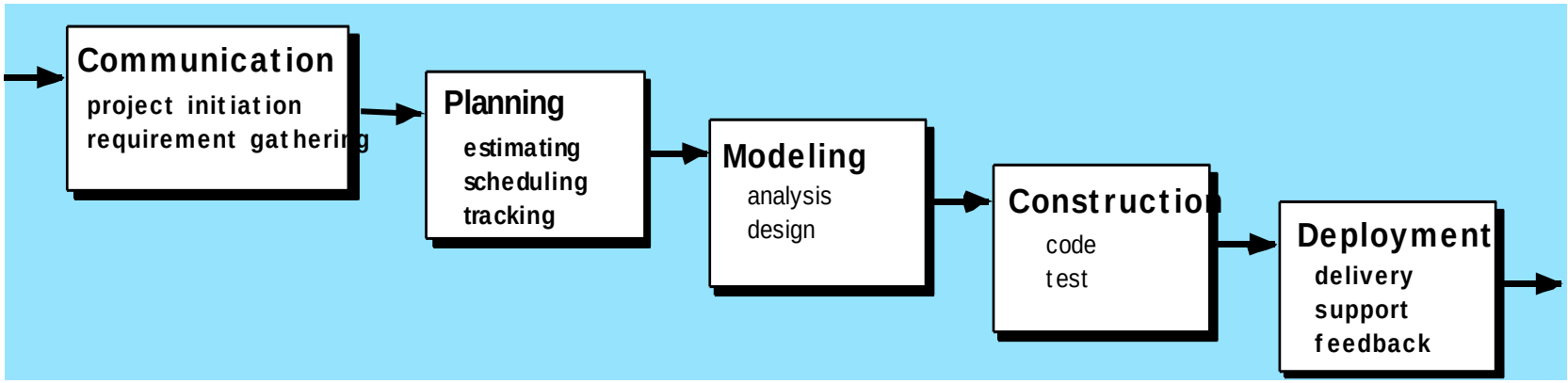
Prescriptive Models

- Prescriptive process models advocate an orderly approach to software engineering

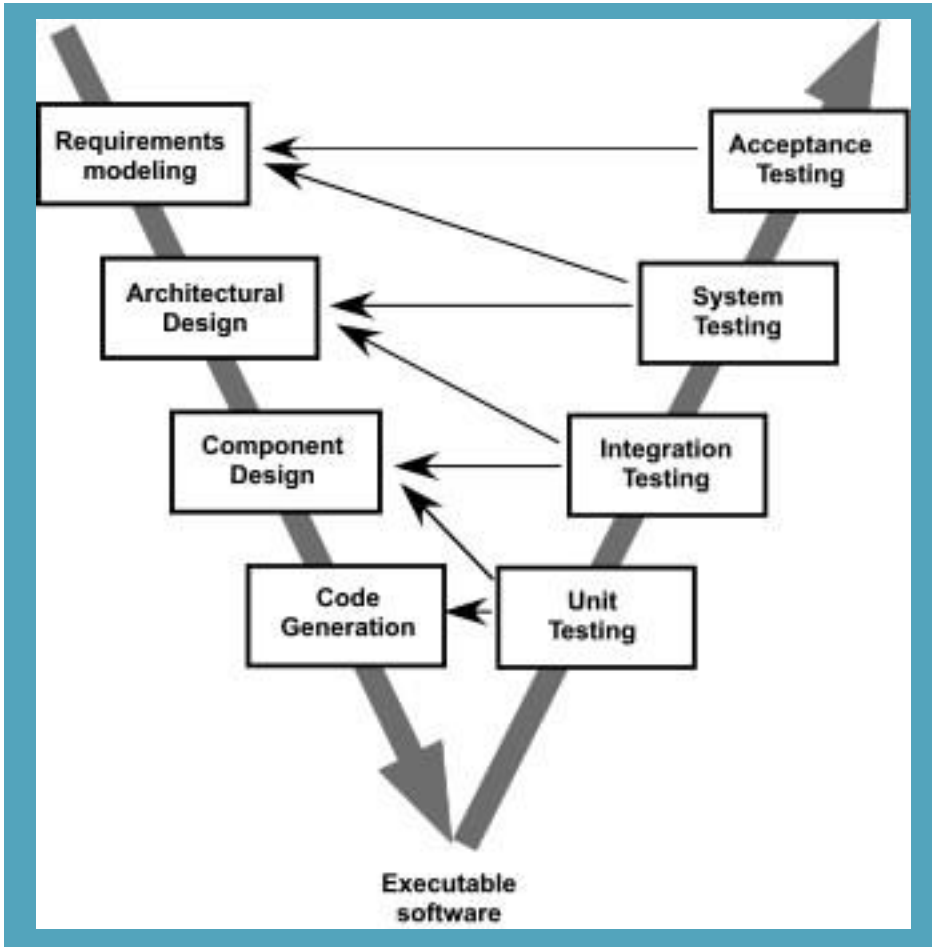
That leads to a few questions …

- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

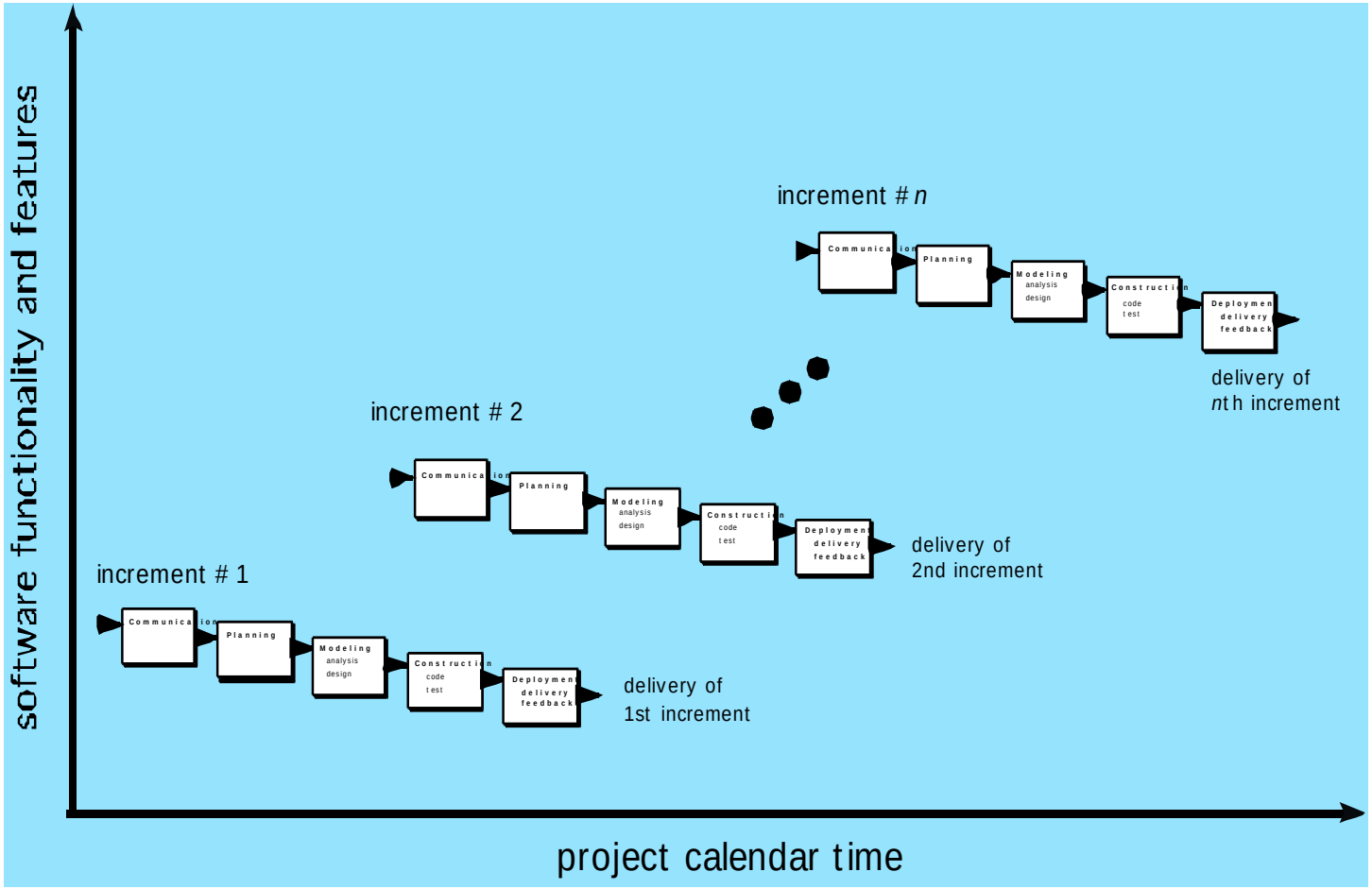
The Waterfall Model



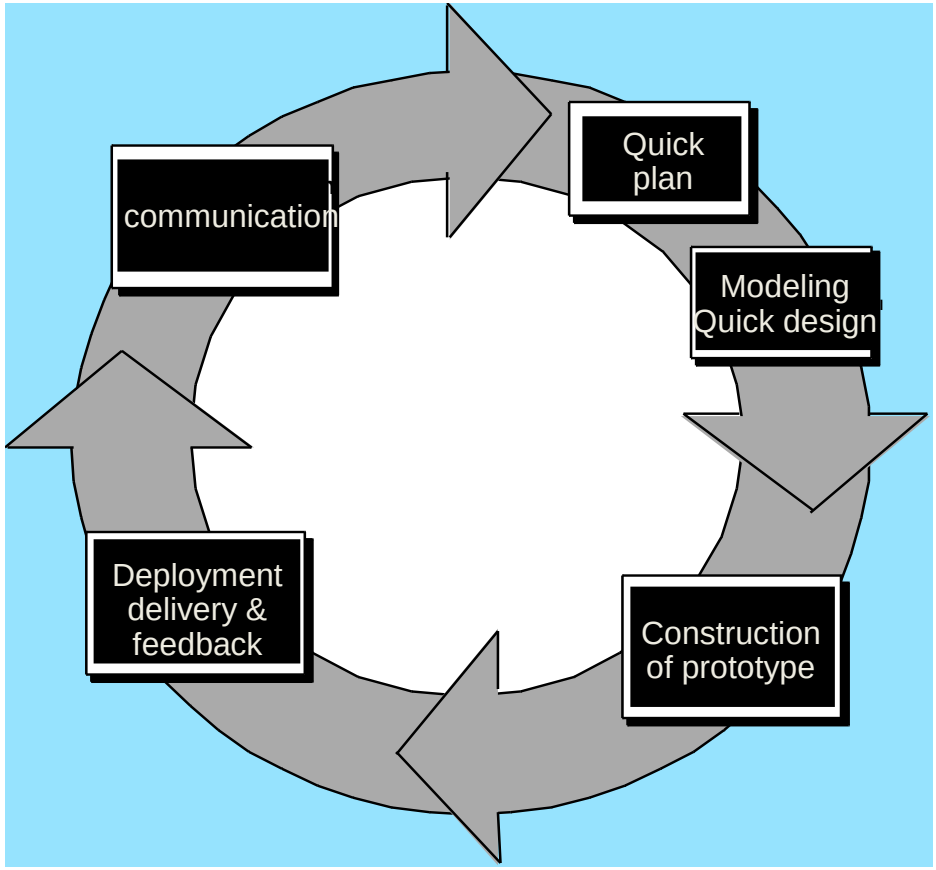
The V-Model


Requirements modeling
Acceptance Testing
Architectural Design
System Testing
Component Design
Integration Testing
Code Generation
Unit Testing
Executable software

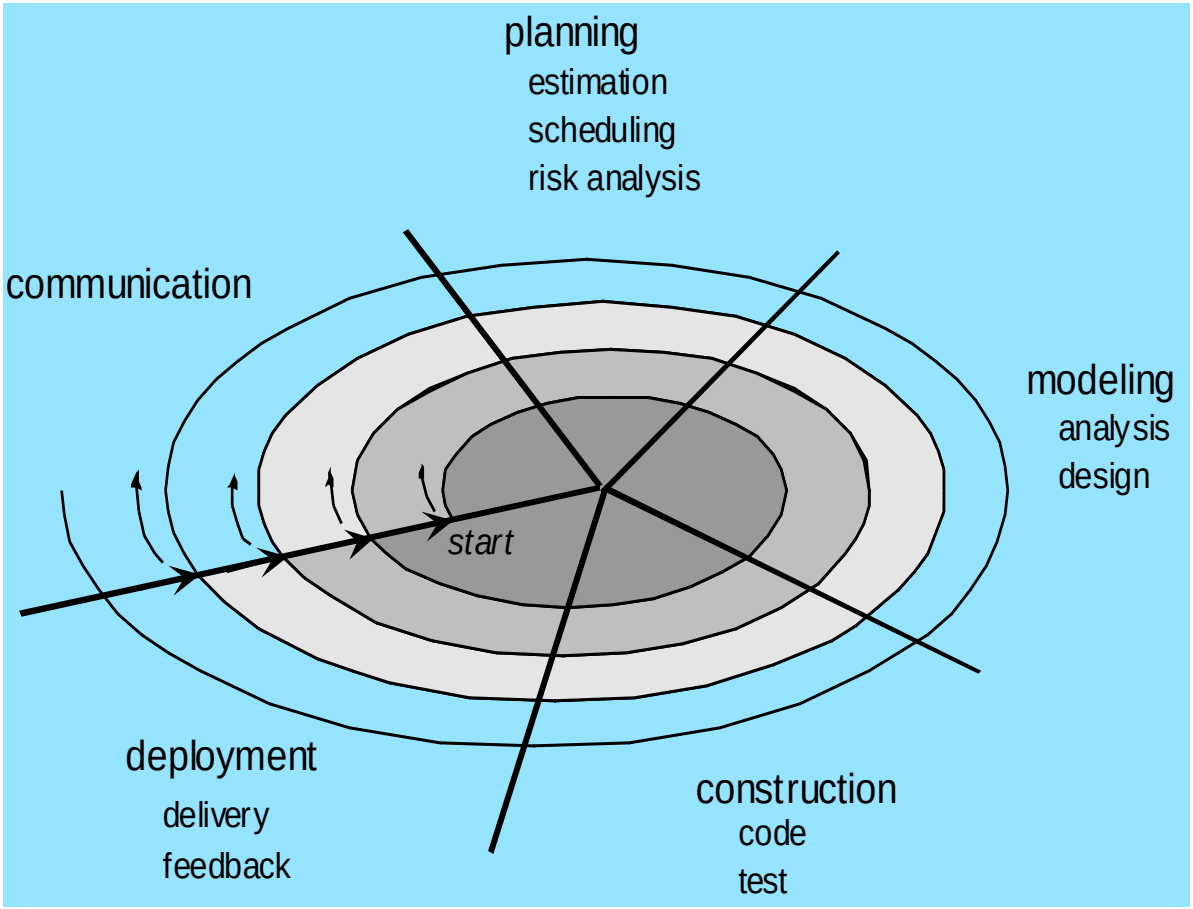
The Incremental Model

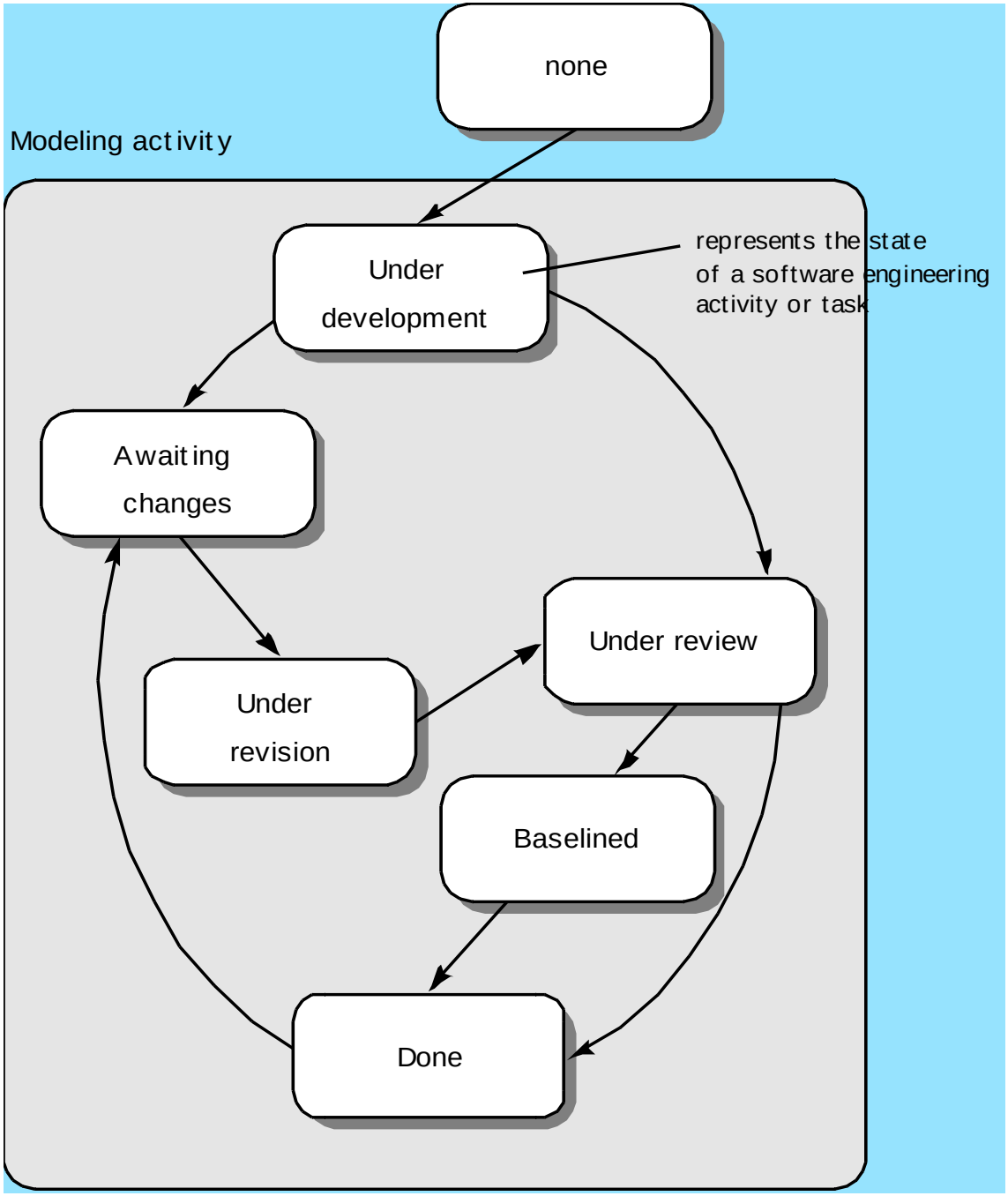


Evolutionary Models: Prototyping



Evolutionary Models: The Spiral



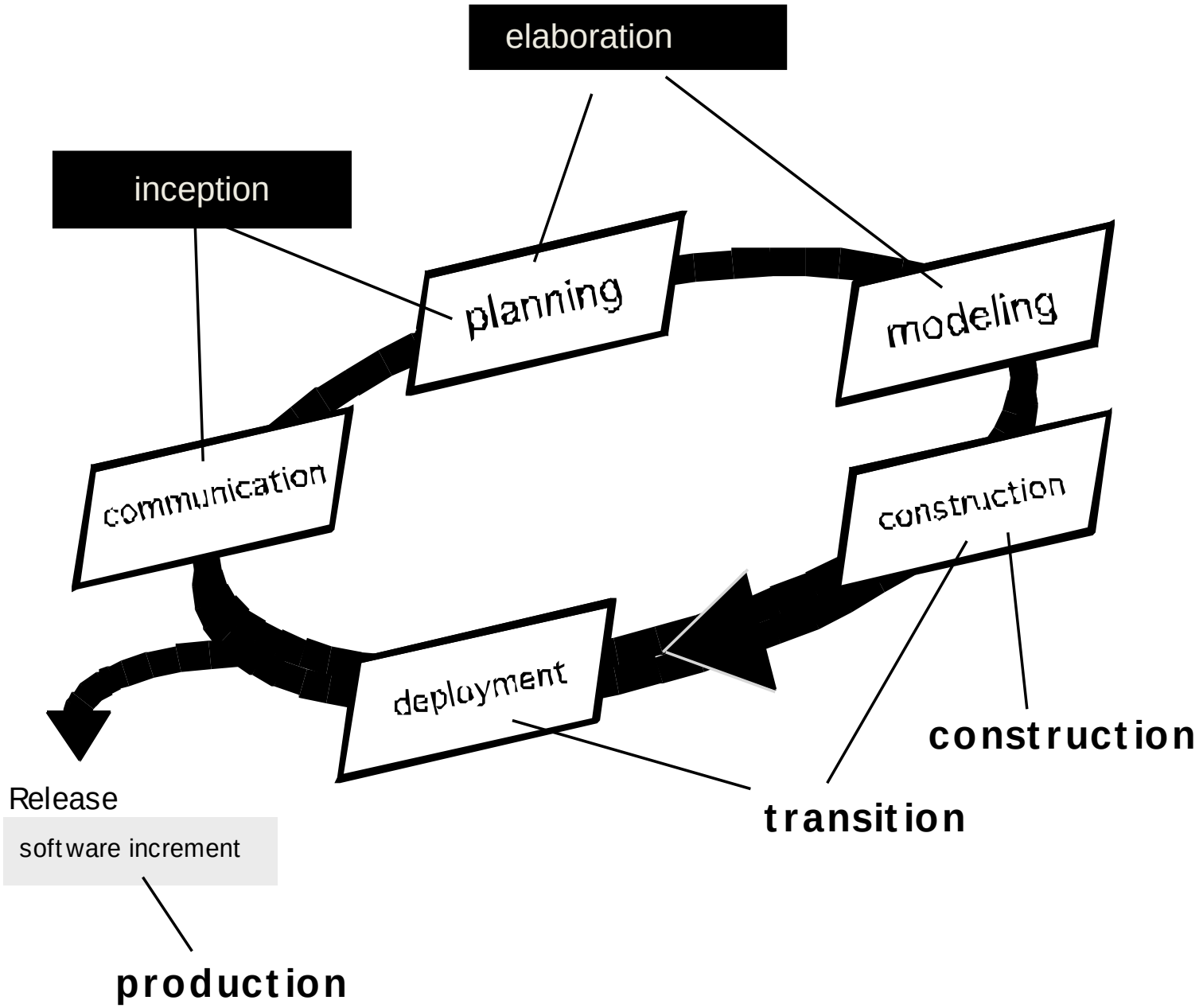


Evolutionary Models: Concurrent

Still Other Process Models

- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

The Unified Process (UP)



The Phases of the Unified Process (contd)

- The four increments are labeled
  - Inception phase
  - Elaboration phase
  - Construction phase
  - Transition phase

- The phases of the Unified Process are the increments

The Inception Phase

- The aim of the inception phase is to determine whether the proposed software product is economically viable

The Inception Phase (contd)

- 1. Gain an understanding of the domain
- 2. Build the business model
- 3. Delimit the scope of the proposed project
  - Focus on the subset of the business model that is covered by the proposed software product
- 4. Begin to make the initial business case

The Inception Phase : The Initial Business Case

- Questions that need to be answered include:
  - Is the proposed software product cost effective?
  - How long will it take to obtain a return on investment?
  - Alternatively, what will be the cost if the company decides not to develop the proposed software product?
  - If the software product is to be sold in the marketplace, have the necessary marketing studies been performed?
  - Can the proposed software product be delivered in time?
  - If the software product is to be developed to support the client organization's own activities, what will be the impact if the proposed software product is delivered late?

The Inception Phase: The Initial Business Case

- What are the risks involved in developing the software product
- How can these risks be mitigated?
  - Does the team who will develop the proposed software product have the necessary experience?
  - Is new hardware needed for this software product?
  - If so, is there a risk that it will not be delivered in time?
  - If so, is there a way to mitigate that risk, perhaps by ordering back-up hardware from another supplier?
  - Are software tools needed?
  - Are they currently available?
  - Do they have all the necessary functionality?

The Inception Phase: The Initial Business Case

- Answers are needed by the end of the inception phase so that the initial business case can be made

The Inception Phase: Risks

- There are three major risk categories:
  - Technical risks
    - See earlier slide
  - The risk of not getting the requirements right
    - Mitigated by performing the requirements workflow correctly
  - The risk of not getting the architecture right
    - The architecture may not be sufficiently robust

The Inception Phase: Risks

- To mitigate all three classes of risks
  - The risks need to be ranked so that the critical risks are mitigated first
- This concludes the steps of the inception phase that fall under the requirements workflow

The Inception Phase: Analysis, Design Workflows

- A small amount of the analysis workflow may be performed during the inception phase
  - Information needed for the design of the architecture is extracted
- Accordingly, a small amount of the design workflow may be performed, too

The Inception Phase: Implementation Workflow

- Coding is generally not performed during the inception phase

- However, a *proof-of-concept prototype* is sometimes build to test the feasibility of constructing part of the software product

The Inception Phase: Test Workflow

- The test workflow commences almost at the start of the inception phase
  - The aim is to ensure that the requirements have been accurately determined

The Inception Phase: Planning

- There is insufficient information at the beginning of the inception phase to plan the entire development
  - The only planning that is done at the start of the project is the planning for the inception phase itself

- For the same reason, the only planning that can be done at the end of the inception phase is the plan for just the next phase, the elaboration phase

The Inception Phase: Documentation

- The deliverables of the inception phase include:
  - The initial version of the domain model
  - The initial version of the business model
  - The initial version of the requirements artifacts
  - A preliminary version of the analysis artifacts
  - A preliminary version of the architecture
  - The initial list of risks
  - The initial ordering of the use cases
  - The plan for the elaboration phase
  - The initial version of the business case

The Inception Phase: The Initial Business Case

- Obtaining the initial version of the business case is the overall aim of the inception phase

- This initial version incorporates
  - A description of the scope of the software product
  - Financial details
  - If the proposed software product is to be marketed, the business case will also include
    - Revenue projections, market estimates, initial cost estimates
  - If the software product is to be used in-house, the business case will include
    - The initial cost–benefit analysis

Elaboration Phase

- The aim of the elaboration phase is to refine the initial requirements
  - Refine the architecture
  - Monitor the risks and refine their priorities
  - Refine the business case
  - Produce the project management plan

- The major activities of the elaboration phase are refinements or elaborations of the previous phase

The Tasks of the Elaboration Phase

- The tasks of the elaboration phase correspond to:
  - All but completing the requirements workflow
  - Performing virtually the entire analysis workflow
  - Starting the design of the architecture

The Elaboration Phase: Documentation

- The deliverables of the elaboration phase include:
  - The completed domain model
  - The completed business model
  - The completed requirements artifacts
  - The completed analysis artifacts
  - An updated version of the architecture
  - An updated list of risks
  - The project management plan (for the rest of the project)
  - The completed business case

Construction Phase

- The aim of the construction phase is to produce the first operational-quality version of the software product
  - This is sometimes called the beta release

The Tasks of the Construction Phase

- The emphasis in this phase is on
  - Implementation and
  - Testing
    - Unit testing of modules
    - Integration testing of subsystems
    - Product testing of the overall system

The Construction Phase: Documentation

- The deliverables of the construction phase include:
  - The initial user manual and other manuals, as appropriate
  - All the artifacts (beta release versions)
  - The completed architecture
  - The updated risk list
  - The project management plan (for the remainder of the project)
  - If necessary, the updated business case

The Transition Phase

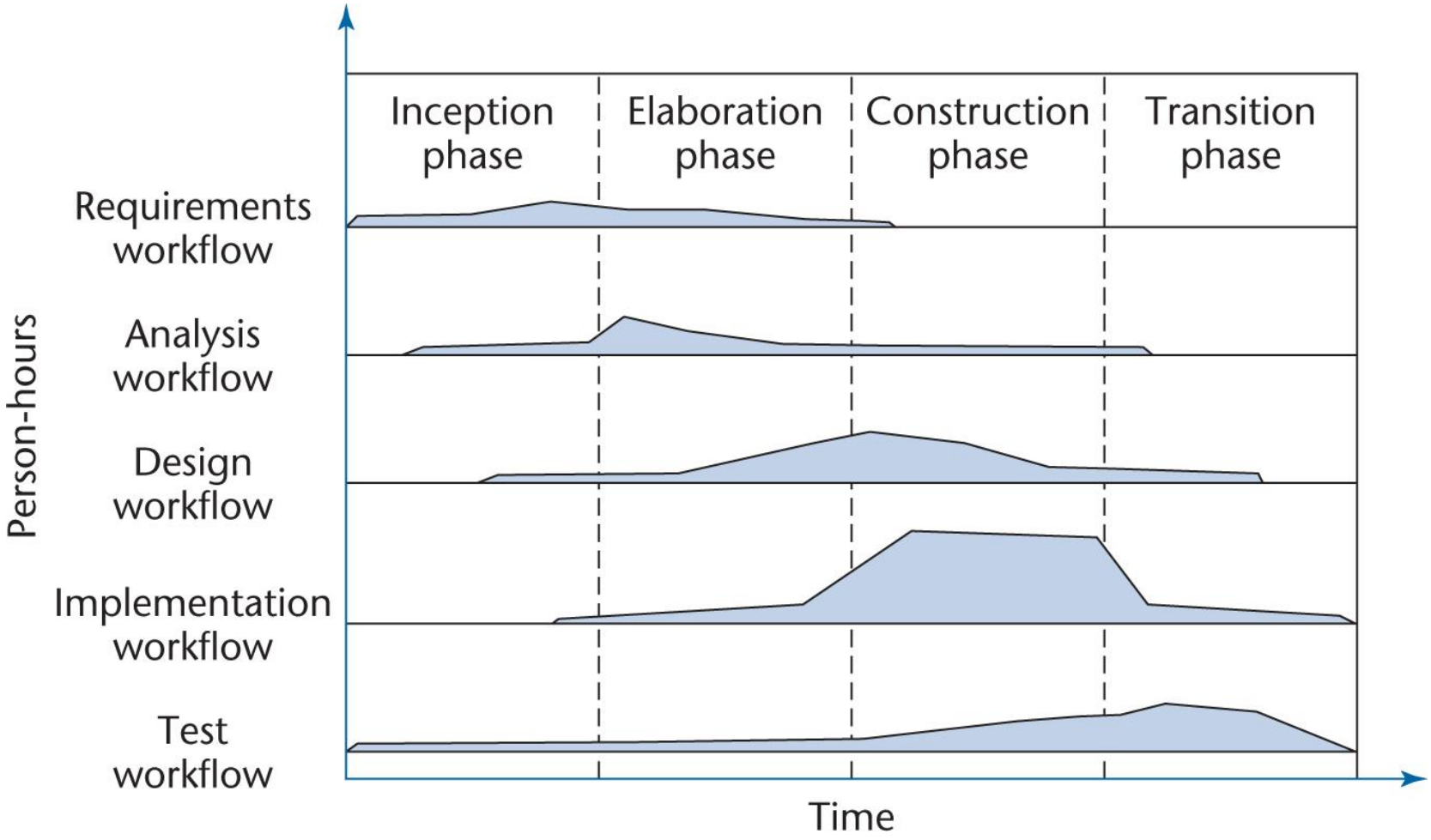
- The aim of the transition phase is to ensure that the client's requirements have indeed been met
  - Faults in the software product are corrected
  - All the manuals are completed
  - Attempts are made to discover any previously unidentified risks
- This phase is driven by feedback from the site(s) at which the beta release has been installed

The Transition Phase: Documentation

- The deliverables of the transition phase include:
  - All the artifacts (final versions)
  - The completed manuals

The Phases of the Unified Process

- The increments are identified as phases



UP Work Products

Inception phase

Vision document
Initial use-case model
Initial project glossary
Initial business case
Initial risk assessment.
Project plan,
phases and iterations.
Business model,
if necessary.
One or more prototypes

Elaboration phase

Use-case model
Supplementary requirements
including non-functional
Analysis model
Software architecture
Description.
Executable architectural
prototype.
Preliminary design model
Revised risk list
Project plan including
iteration plan
adapted workflows
milestones
technical work products
Preliminary user manual

Construction phase

Design model
Software components
Integrated software
increment
Test plan and procedure
Test cases
Support documentation
user manuals
installation manuals
description of current
increment

Transition phase

Delivered software increment
Beta test reports
General user feedback

Capability Maturity Models

- Not life-cycle models
- Rather, a set of strategies for improving the software process
  - SW–CMM for software
  - P–CMM for human resources (“people”)
  - SE–CMM for systems engineering
  - IPD–CMM for integrated product development
  - SA–CMM for software acquisition
- These strategies are unified into CMMI (capability maturity model integration)

SW–CMM

- A strategy for improving the software process
- Put forward in 1986 by the SEI
- Fundamental ideas:
  - Improving the software process leads to
    - Improved software quality
    - Delivery on time, within budget
  - Improved management leads to
    - Improved techniques

SW–CMM (contd)

- Five levels of *maturity* are defined
  - Maturity is a measure of the goodness of the process itself

- An organization advances stepwise from level to level

Level 1. Initial Level

- Ad hoc approach
  - The entire process is unpredictable
  - Management consists of responses to crises

- Most organizations world-wide are at level 1

Level 2. Repeatable Level

- Basic software management
  - Management decisions should be made on the basis of previous experience with similar products
  - Measurements (“metrics”) are made
  - These can be used for making cost and duration predictions in the next project
  - Problems are identified, immediate corrective action is taken

Level 3. Defined Level

- The software process is fully documented
  - Managerial and technical aspects are clearly defined
  - Continual efforts are made to improve quality and productivity
  - Reviews are performed to improve software quality
  - CASE environments are applicable *now* (and not at levels 1 or 2)

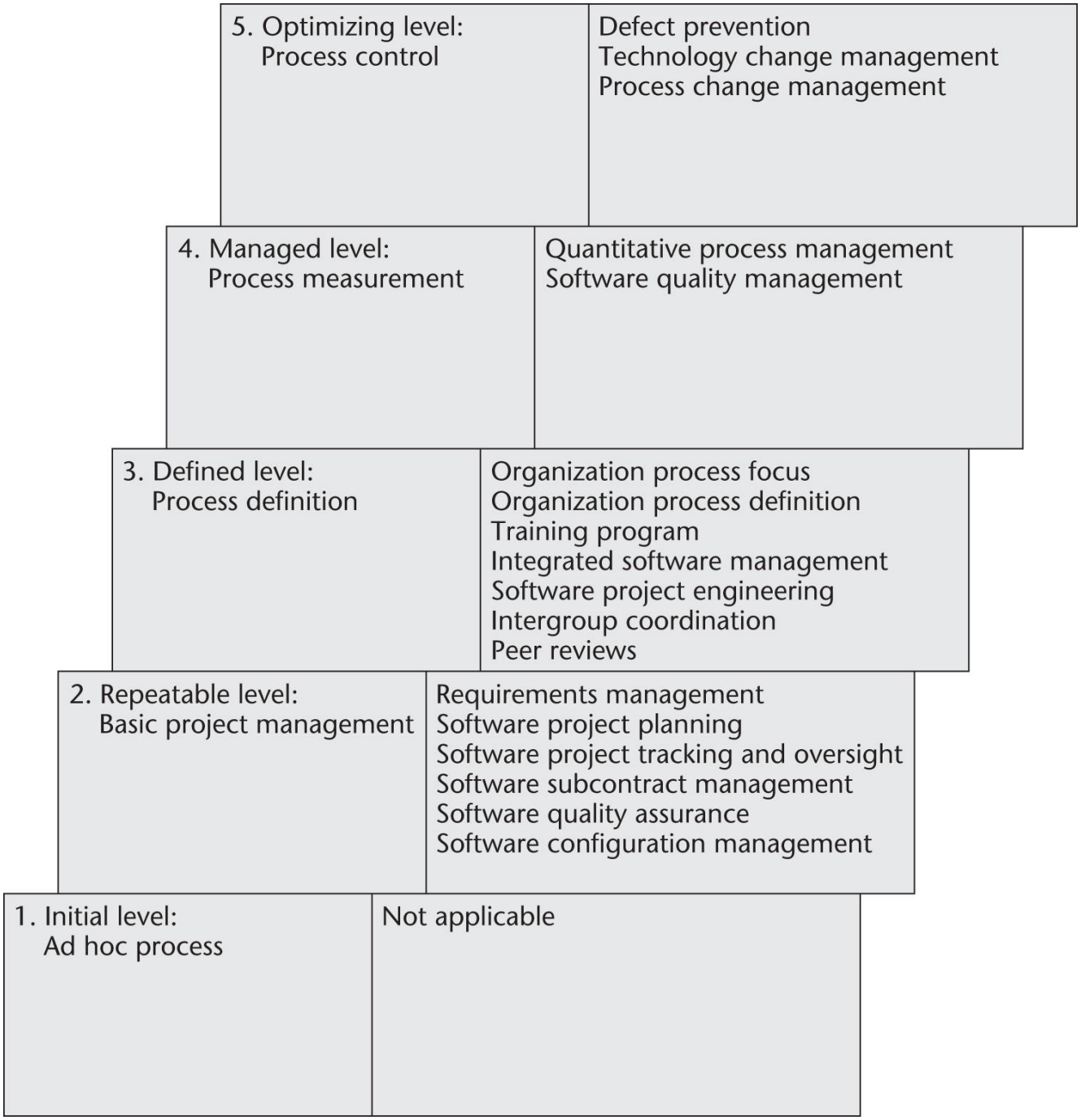
Level 4. Managed Level

- Quality and productivity goals are set for each project
  - Quality and productivity are continually monitored
  - Statistical quality controls are in place

Level 5. Optimizing Level

- Continuous process improvement
  - Statistical quality and process controls
  - Feedback of knowledge from each project to the next

Summary


5. Optimizing level:
Process control
Defect prevention
Technology change management
Process change management
4. Managed level:
Process measurement
Quantitative process management
Software quality management
3. Defined level:
Process definition
Organization process focus
Organization process definition
Training program
Integrated software management
Software project engineering
Intergroup coordination
Peer reviews
2. Repeatable level:
Basic project management
Requirements management
Software project planning
Software project tracking and oversight
Software subcontract management
Software quality assurance
Software configuration management
1. Initial level:
Ad hoc process
Not applicable

Experiences with SW–CMM

- It takes:
  - 3 to 5 years to get from level 1 to level 2
  - 1.5 to 3 years from level 2 to level 3
  - SEI questionnaires highlight shortcomings, suggest ways to improve the process

Personal Software Process (PSP)

- **Planning.** This activity isolates requirements and develops both size and resource estimates. In addition, a defect estimate (the number of defects projected for the work) is made. All metrics are recorded on worksheets or templates. Finally, development tasks are identified and a project schedule is created.
- **High-level design.** External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.
- **High-level design review.** Formal verification methods (Chapter 21) are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.
- **Development.** The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.
- **Postmortem.** Using the measures and metrics collected (this is a substantial amount of data that should be analyzed statistically), the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness.

Team Software Process (TSP)

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of three to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
  - The Capability Maturity Model (CMM), a measure of the effectiveness of a software process, is discussed in Chapter 30.
- Provide improvement guidance to high-maturity organizations.
- Facilitate university teaching of industrial-grade team skills.