Object-Oriented and Classical Software Engineering

THE SOFTWARE PROCESS

Overview

- The Unified Process
- Iteration and incrementation within the object-oriented paradigm
- The requirements workflow
- The analysis workflow
- The design workflow
- The implementation workflow
- The test workflow

Overview (contd)

- Postdelivery maintenance
- Retirement
- The phases of the Unified Process
- One- versus two-dimensional life-cycle models
- Improving the software process
- Capability maturity models
- Other software process improvement initiatives
- Costs and benefits of software process improvement

3.1 The Unified Process

- Until recently, three of the most successful object-oriented methodologies were
  - Booch's method
  - Jacobson's Objectory
  - Rumbaugh's OMT

The Unified Process (contd)

- In 1999, Booch, Jacobson, and Rumbaugh published a complete object-oriented analysis and design methodology that unified their three separate methodologies
  - Original name: *Rational Unified Process* (RUP)
  - Next name: *Unified Software Development Process* (USDP)
  - Name used today: *Unified Process* (for brevity)

The Unified Process (contd)

- The Unified Process is *not* a series of steps for constructing a software product
  - No such single “one size fits all” methodology could exist
  - There is a wide variety of different types of software
- The Unified Process is an adaptable methodology
  - It has to be modified for the specific software product to be developed

The Unified Process (contd)

- UML is graphical
  - A picture is worth a thousand words
- UML diagrams enable software engineers to communicate quickly and accurately

- The Unified Process is a modeling technique
  - A *model* is a set of UML diagrams that represent various aspects of the software product we want to develop

- UML stands for unified *modeling* language
  - UML is the tool that we use to represent (model) the target software product

- The object-oriented paradigm is iterative and incremental in nature
  - There is no alternative to repeated iteration and incrementation until the UML diagrams are satisfactory

- The version of the Unified Process in this book is for
  - Software products small enough to be developed by a team of three students during the semester or quarter

- However, the modifications to the Unified Process for developing a large software product are also discussed

- The goals of this book include:
  - A thorough understanding of how to develop smaller software products
  - An appreciation of the issues that need to be addressed when larger software products are constructed

- We cannot learn the complete Unified Process in one semester or quarter
  - Extensive study and unending practice are needed
  - The Unified Process has too many features
  - A case study of a large-scale software product is huge

- In this book, we therefore cover much, but not all, of the Unified Process
  - The topics covered are adequate for smaller products

- To work on larger software products, experience is needed
  - This must be followed by training in the more complex aspects of the Unified Process

3.3 The Requirements Workflow

- The aim of the requirements workflow
  - To determine the client's needs

Overview of the Requirements Workflow

- First, gain an understanding of the *application domain* (or *domain*, for short)
  - That is, the specific business environment in which the software product is to operate

- Second, build a business model
  - Use UML to describe the client’s business processes
  - If at any time the client does not feel that the cost is justified, development terminates immediately

Overview of the Requirements Workflow (contd)

- It is vital to determine the client's constraints
  - Deadline
    - Nowadays, software products are often mission critical
  - Parallel running
  - Portability
  - Reliability
  - Rapid response time
  - Cost
    - The client will rarely inform the developer how much money is available
    - A bidding procedure is used instead

Overview of the Requirements Workflow (contd)

- The aim of this *concept exploration* is to determine
  – What the client needs
  – *Not* what the client wants

3.4 The Analysis Workflow

- The aim of the analysis workflow
  - To analyze and refine the requirements
- Why not do this during the requirements workflow?
  - The requirements artifacts must be totally comprehensible by the client
- The artifacts of the requirements workflow must therefore be expressed in a natural (human) language
  - All natural languages are imprecise

The Analysis Workflow (contd)

- Example from a manufacturing information system:
  - "A part record and a plant record are read from the database. If it contains the letter A directly followed by the letter Q, then calculate the cost of transporting that part to that plant"

- To what does it refer?
  - The part record?
  - The plant record?
  - Or the database?

The Analysis Workflow (contd)

- Two separate workflows are needed
  - The requirements artifacts must be expressed in the language of the client
  - The analysis artifacts must be precise, and complete enough for the designers

The Specification Document (contd)

- Specification document (“specifications”)
  - It constitutes a contract
  - It must not have imprecise phrases like “optimal,” or “98% complete”
- Having complete and correct specifications is essential for
  - Testing and
  - Maintenance

The Specification Document (contd)

- The specification document must not have
  - Contradictions
  - Omissions
  - Incompleteness

Software Project Management Plan

- Once the client has signed off the specifications, detailed planning and estimating begins

- We draw up the software project management plan, including
  – Cost estimate
  – Duration estimate
  – Deliverables
  – Milestones
  – Budget

- This is the earliest possible time for the SPMP

3.5 The Design Workflow

- The aim of the design workflow is to refine the analysis workflow until the material is in a form that can be implemented by the programmers
  - Many nonfunctional requirements need to be finalized at this time, including
    - » Choice of programming language
    - » Reuse issues
    - » Portability issues

Classical Design

- Architectural design
  - Decompose the product into modules

- Detailed design
  - Design each module:
    - » Data structures
    - » Algorithms

Object-Oriented Design

- Classes are extracted during the object-oriented analysis workflow and
  - Designed during the design workflow

- Accordingly
  - Classical architectural design corresponds to part of the object-oriented analysis workflow
  - Classical detailed design corresponds to part of the object-oriented design workflow

The Design Workflow (contd)

- Retain design decisions
  - For when a dead-end is reached
  - To prevent the maintenance team reinventing the wheel

3.6 The Implementation Workflow

- The aim of the implementation workflow is to implement the target software product in the selected implementation language
  - A large software product is partitioned into subsystems
  - The subsystems consist of *components* or *code artifacts*

3.7 The Test Workflow

- The test workflow is the responsibility of
  - *Every* developer and maintainer, and
  - The quality assurance group

- Traceability of artifacts is an important requirement for successful testing

3.7.1 Requirements Artifacts

- Every item in the analysis artifacts must be traceable to an item in the requirements artifacts
  - Similarly for the design and implementation artifacts

3.7.2 Analysis Artifacts

- The analysis artifacts should be checked by means of a review
  - Representatives of the client and analysis team must be present
- The SPMP must be similarly checked
  - Pay special attention to the cost and duration estimates

3.7.3 Design Artifacts

- Design reviews are essential
  - A client representative is not usually present

3.7.4 Implementation Artifacts

- Each component is tested as soon as it has been implemented
  - *Unit testing*
- At the end of each iteration, the completed components are combined and tested
  - *Integration testing*
- When the product appears to be complete, it is tested as a whole
  - *Product testing*
- Once the completed product has been installed on the client’s computer, the client tests it
  - *Acceptance testing*

Implementation Artifacts (contd)

- COTS software is released for testing by prospective clients
  - Alpha release
  - Beta release

- There are advantages and disadvantages to being an alpha or beta release site

3.8 Postdelivery Maintenance

- Postdelivery maintenance is an essential component of software development
  - More money is spent on postdelivery maintenance than on all other activities combined
- Problems can be caused by
  - Lack of documentation of all kinds

Postdelivery Maintenance (contd)

- Two types of testing are needed
  - Testing the changes made during postdelivery maintenance
  - Regression testing
- All previous test cases (and their expected outcomes) need to be retained

3.9 Retirement

- Software can be unmaintainable because
  - A drastic change in design has occurred
  - The product must be implemented on a totally new hardware/operating system
  - Documentation is missing or inaccurate
  - Hardware is to be changed — it may be cheaper to rewrite the software from scratch than to modify it

- These are instances of maintenance (rewriting of existing software)

Retirement (contd)

- True retirement is a rare event
- It occurs when the client organization no longer needs the functionality provided by the product

3.10 The Phases of the Unified Process

- The increments are identified as phases

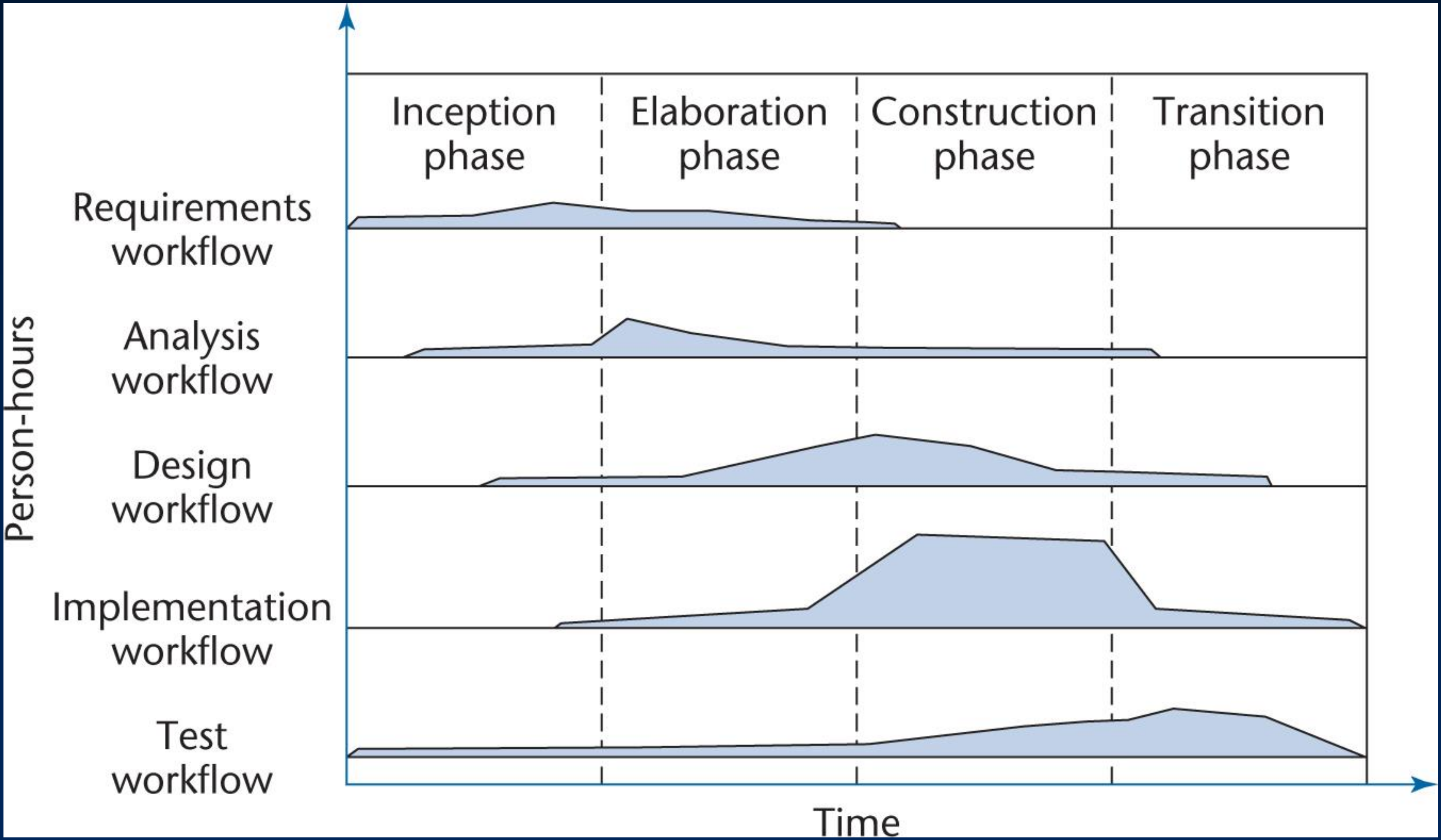

Figure 3.1

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 Phases of the Unified Process (contd)

- In theory, there could be any number of increments
  - In practice, development seems to consist of four increments

- Every step performed in the Unified Process falls into
  - One of the five core workflows and *also*
  - One of the four phases

- Why does each step have to be considered twice?

The Phases of the Unified Process (contd)

- Workflow
  - Technical context of a step

- Phase
  - Business context of a step

3.10.1 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 (Chapter 5) 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 (Chapter 10)
  - 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

3.10.2 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

3.10.3 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

3.10.4 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

3.11 One- and Two-Dimensional Life-Cycle Models

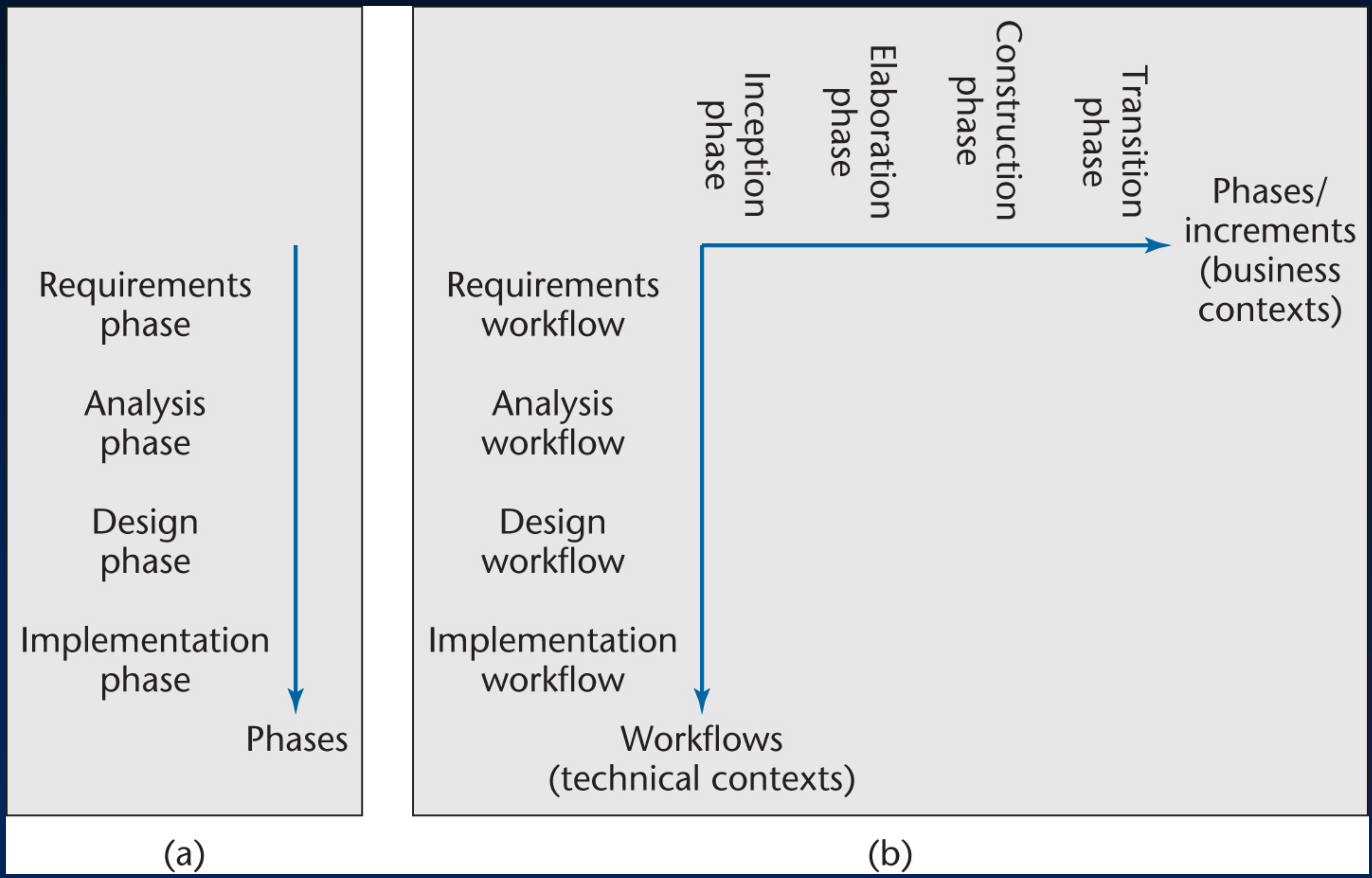

Figure 3.2

Why a Two-Dimensional Model?

- A traditional life cycle is a one-dimensional model
  - Represented by the single axis on the previous slide
    - Example: Waterfall model

- The Unified Process is a two-dimensional model
  - Represented by the two axes on the previous slide

- The two-dimensional figure shows
  - The workflows (technical contexts) and
  - The phases (business contexts)

Why a Two-Dimensional Model? (contd)

- The waterfall model

- One-dimensional

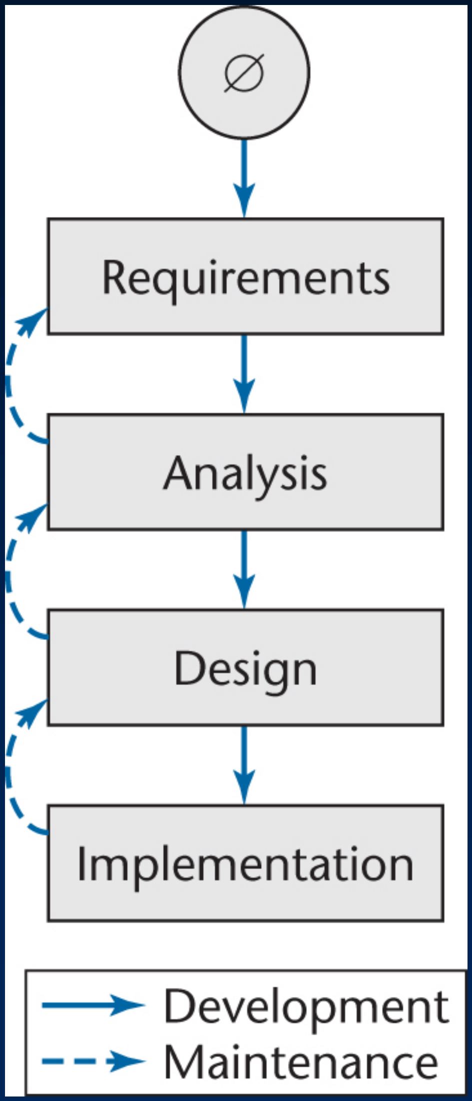

Figure 2.3 (again)

Why a Two-Dimensional Model? (contd)

- Evolution tree model
- Two-dimensional

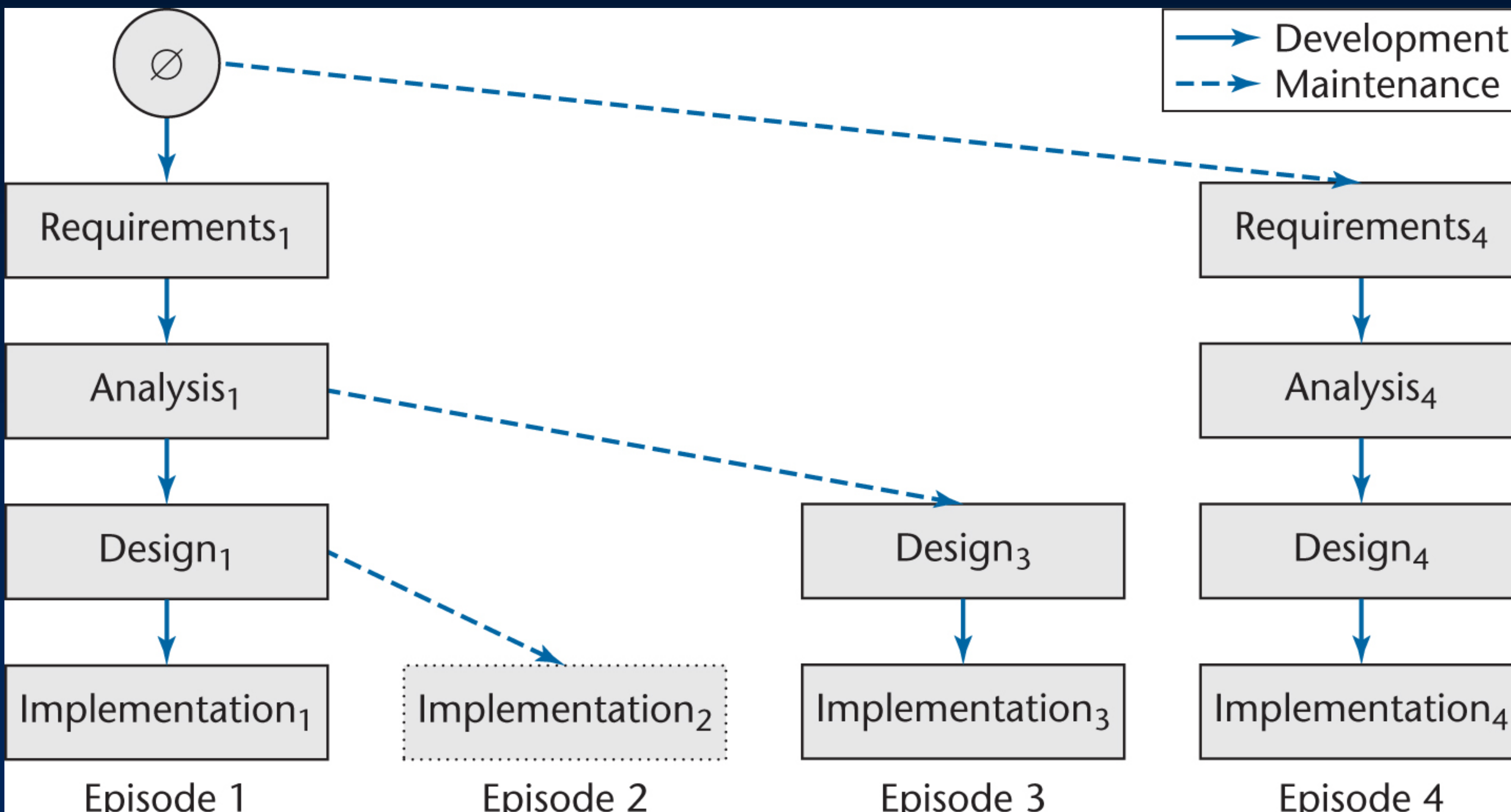

Figure 2.2 (again)

Why a Two-Dimensional Model? (contd)

- Are all the additional complications of the two-dimensional model necessary?

- In an ideal world, each workflow would be completed before the next workflow is started

Why a Two-Dimensional Model? (contd)

- In reality, the development task is too big for this

- As a consequence of Miller's Law
  - The development task has to be divided into increments (phases)
  - Within each increment, iteration is performed until the task is complete

Why a Two-Dimensional Model? (contd)

- At the beginning of the process, there is not enough information about the software product to carry out the requirements workflow
  - Similarly for the other core workflows

- A software product has to be broken into subsystems

- Even subsystems can be too large at times
  - Components may be all that can be handled until a fuller understanding of all the parts of the product as a whole has been obtained

Why a Two-Dimensional Model? (contd)

- The Unified Process handles the inevitable changes well
  - The moving target problem
  - The inevitable mistakes

- The Unified Process is the best solution found to date for treating a large problem as a set of smaller, largely independent subproblems
  - It provides a framework for incrementation and iteration
  - In the future, it will inevitably be superseded by some better methodology

3.12 Improving the Software Process

- Example:
- U.S. Department of Defense initiative
- Software Engineering Institute (SEI)
- The fundamental problem with software
  - The software process is badly managed

Improving the Software Process (contd)

- Software process improvement initiatives
  - Capability maturity model (CMM)
  - ISO 9000-series
  - ISO/IEC 15504

3.13 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

Figure 3.3

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

Key Process Areas

- There are key process areas (KPAs) for each level

Key Process Areas (contd)

- Level-2 KPAs include:
  - Requirements management
  - Project planning
  - Project tracking
  - Configuration management
  - Quality assurance

- Compare
  - Level 2: Detection and correction of faults
  - Level 5: Prevention of faults

Goals

- Original goal:
  - Defense contracts would be awarded only to capable firms
- The U.S. Air Force stipulated that every Air Force contractor had to attain SW–CMM level 3 by 1998
  - The DoD subsequently issued a similar directive
- The CMM has now gone far beyond the limited goal of improving DoD software

3.14 Other Software Process Improvement Initiatives

- Other software process improvement (SPI) initiatives include:
  - ISO 9000-series
  - ISO/IEC 15504

ISO 9000

- A set of five standards for industrial activities
  - ISO 9001 for quality systems
  - ISO 9000-3, guidelines to apply ISO 9001 to software
  - There is an overlap with CMM, but they are not identical
  - *Not* process improvement
  - There is a stress on documenting the process
  - There is an emphasis on measurement and metrics
  - ISO 9000 is required to do business with the EU
  - Also required by many U.S. businesses, including GE
  - More and more U.S. businesses are ISO 9000 certified

ISO/IEC 15504

- Original name: Software Process Improvement Capability dEtermination (SPICE)
  - International process improvement initiative
  - Started by the British Ministry of Defence (MOD)
  - Includes process improvement, software procurement
  - Extends and improves CMM, ISO 9000
  - A framework, not a method
    - » CMM, ISO 9000 conform to this framework
  - Now referred to as ISO/IEC 15504
  - Or just 15504 for short

- Hughes Aircraft (Fullerton, CA) spent $500K (1987–90)
  - Savings: $2M per year, moving from level 2 to level 3
- Raytheon moved from level 1 in 1988 to level 3 in 1993
  - Productivity doubled
  - Return of $7.70 per dollar invested in process improvement

Costs and Benefits of Software Process Improvement (contd)

- Tata Consultancy Services (India) used ISO 9000 and CMM (1996–90)
  - Errors in estimation decreased from 50% to 15%
  - Effectiveness of reviews increased from 40% to 80%

- Motorola GED has used CMM (1992–97)
  - Results are shown in the next slide

Results of 34 Motorola Projects

CMM Level	Number of Projects	Relative Decrease in Duration	Faults per MEASL Detected during Development	Relative Productivity
Level 1	3	1.0	—	—
Level 2	9	3.2	890	1.0
Level 3	5	2.7	411	0.8
Level 4	8	5.0	205	2.3
Level 5	9	7.8	126	2.8

Figure 3.4

- MEASL – Million equivalent assembler source lines

- Motorola does not reveal productivity data
  - Productivity is measured relative to that of a selected level-2 project
  - No fault or productivity data available for level-1 projects (by definition)

Reanalysis of 85 CMM Projects

- Galin and Avrahami (2006) analyzed 85 projects that had been reported as having advanced one level after implemented CMM

- The projects were divided into four groups
  - CMM level 1 to level 2
  - CMM level 2 to level 3
  - CMM level 3 to level 4
  - CMM level 4 to level 5

Reanalysis of 85 CMM Projects (contd)

- For the four groups:
  - Median fault density (number of faults per KLOC) decreased by between 26 and 63 percent
  - Median productivity (KLOC per person month) increased by between 26 and 187 percent
  - Median rework decreased by between 34 and 40 percent
  - Median project duration decreased by between 28 and 53 percent

Reanalysis of 85 CMM Projects (contd)

- Fault detection effectiveness (percentage of faults detected during development of the total detected project faults) increased as follows:
  - For the three lowest groups, the median increased by between 70 and 74 percent, and
  - By 13 percent for the highest group (CMM level 4 to level 5)

- The return on investments varied between 120 to 650 percent
  - With a median value of 360 percent

Costs and Benefits of Software Process Improvement (contd)

- Published studies such as these are convincing more and more organizations worldwide that process improvement is cost effective

Costs and Benefits of Software Process Improvement (contd)

- There is interplay between
  - Software engineering standards organizations and
  - Software process improvement initiatives
- ISO/IEC 12207 (1995) is a full life-cycle software standard
- In 1998, the U.S. version (IEEE/EIA 12207) was published that incorporated ideas from CMM
- ISO 9000-3 now incorporates part of ISO/IEC 12207