

TUM

# Software Engineering II

## Unified Process

**Bernd Bruegge**

Technische Universität München

Institut für Informatik

Lehrstuhl für Angewandte Softwaretechnik

<http://www.bruegge.in.tum.de>

27 June 2006

# Outline of Today's Lecture

- Unified Process: An iterative process model
- States of a software system developed with the Unified Process:
  - Inception, Elaboration, Construction, Transition
- Artifacts Sets:
  - Management Set, Engineering Set
- Workflows:
  - Management, Environment, Requirements, Design, Implementation, Assessment, Deployment
- Iterations
- Managing iterations as software projects
- Mistakes in managing iterations



# Review of some Definitions

- **Software life cycle:**
  - Set of activities and their relationships to each other to support the development of a software system
- **Software development methodology:**
  - A collection of techniques for building models - applied across the software life cycle

# Software Life Cycle Questions (Review)

- Which activities should I select for the software project?
- What are the dependencies between activities?
- How should I schedule the activities?
- Questions to ask:
  - What is the problem?
  - What is the solution?
  - What are the mechanisms that best implement the solution?
  - How is the solution constructed?
  - Is the problem solved?
  - Can the customer use the solution?
  - How do we deal with changes that occur during the development? Are enhancements needed



# Life Cycle Modeling

- So far we have discussed the life cycle models
  - Waterfall model
  - V-model
  - Spiral model
  - Issue-based model
- Today we will cover another lifecycle model
  - Unified Software Process

# “Processes” in the Unified Process

The term **Process** is overloaded in the Unified Process.

- **Micro process:** Policies & practices for building an artifact
  - Focus: Intermediate baselines with adequate quality and functionality as economically and rapidly as practical
  - Same as “Process” in the IEEE Standard
- **Macro process:** A set of micro processes and the dependencies among them
  - Focus: Production of a software system within cost, schedule and quality constraints
  - Also called: Lifecycle Model
- **Meta process**
  - Focus: Organizational improvement, long-term strategies, and return on investment (ROI)
  - Also called: Business process

# The Unified Process

- The Unified Process supports the following
  1. Evolution of project plans, requirements and software architecture with well-defined synchronization points
  2. Risk management
  3. Evolution of system capabilities through demonstrations of increasing functionality

It emphasizes the difference between *engineering* and *production*.

# Difference: Engineering vs. Production

- **Engineering Stage**
  - Driven by less predictable but smaller teams, focusing on design and synthesis activities
- **Production Stage**
  - Driven by more predictable but larger teams, focusing on construction, test and deployment activities

<b>Focus</b>	<b>Engineering Stage Emphasis</b>	<b>Production Stage Emphasis</b>
Risk	Technical feasibility, Schedule	Cost
Artifacts	Planning, Requirements, System Design Documents	Baselines, Releases
Activities	Planning, Analysis, Design	Implementation, Integration
Quality Assessment	Demonstration, Inspection	Testing

# Phases in the Unified Process

- The two stages of the Unified Process are decomposed into four distinct phases
- Engineering stage
  - Inception phase
  - Elaboration phase
- Production phase
  - Construction phase
  - Transition phase.

# Transitioning from Engineering to Production

When the “engineering” of the system is complete, a decision must be made:

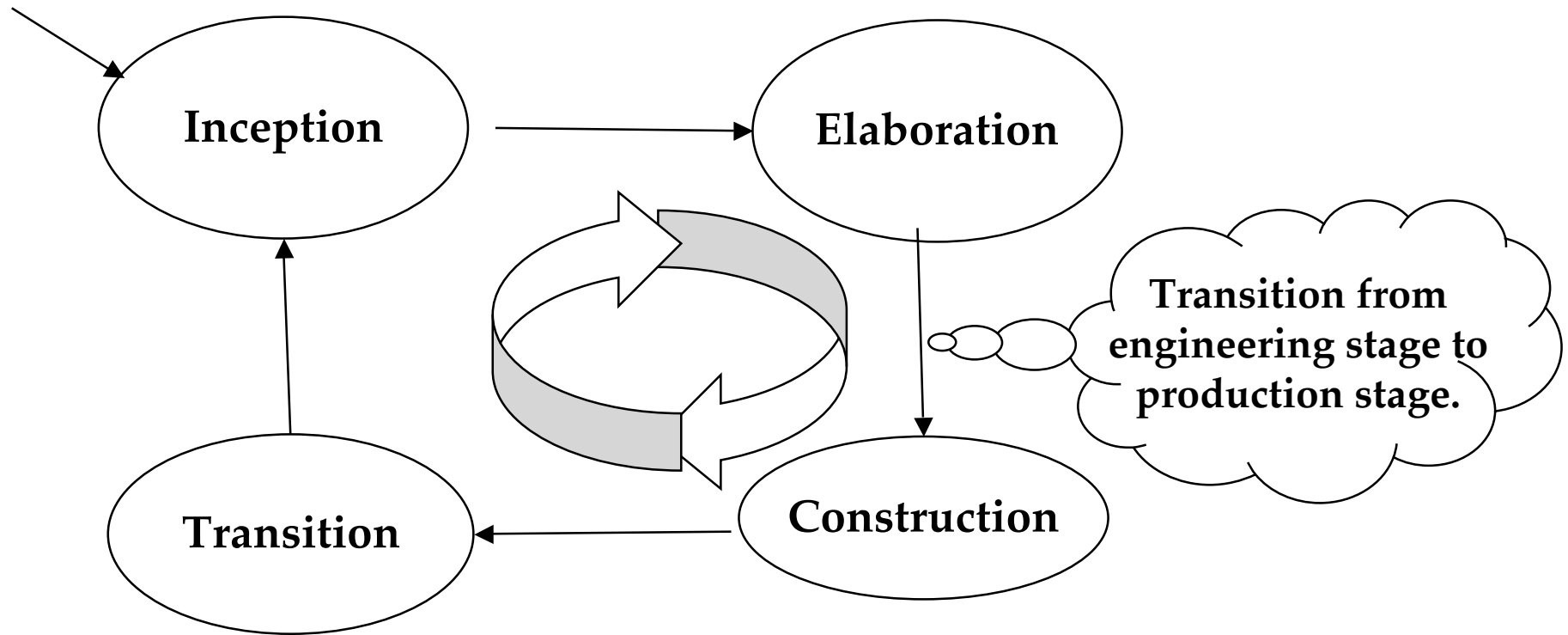
- Commit to production phase?
- Move to an operation with higher cost risk and inertia (i.e. bureaucracy)

Main questions:

- Are the system models and project plans stable enough?
- Have the risks been dealt with?
- Can we predict cost and schedule for the completion of the development for an acceptable range?

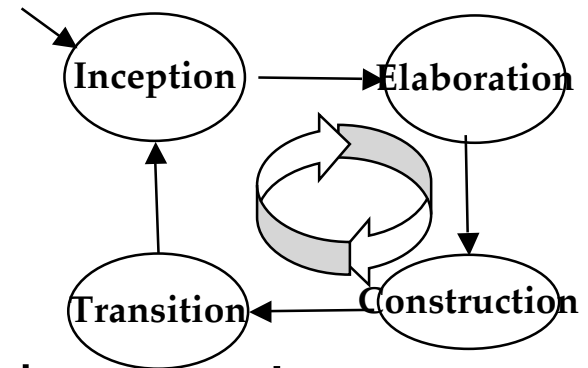


# States of a Software System in the UP




# Inception Phase: Objectives

- Establish the project scope
- Identify the critical use cases and scenarios
- Define acceptance criteria
- Demonstrate at least one candidate software architecture
- Estimate the cost and schedule for the project
- Define and estimate potential risks.



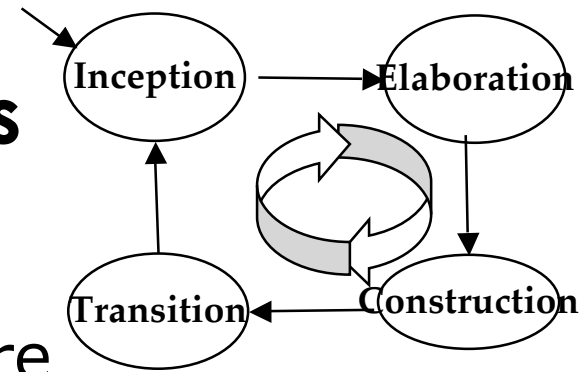
# Inception Phase: Activities

- Formulate the scope of the project
  - Capture requirements
  - Result: problem space and acceptance criteria are defined
- Design the software architecture
  - Evaluate design trade-offs, investigate solution space
  - Result: Feasibility of at least one candidate architecture is explored, initial set of build vs. buy decisions 
- Plan and prepare a business case
  - Evaluate alternatives for risks, staffing problems, plans.

# Inception Phase: Evaluation Criteria

- Do all stakeholders concur on the scope definition and cost and schedule estimates?
- Are the requirements understood, are the critical use cases adequately modeled?
- Is the software architecture understood?
- Are cost, schedule estimates, priorities, risks and development processes credible?
- Is there a prototype that helps in evaluating the criteria?

# Elaboration Phase: Objectives



- Baseline the software architecture
  - Establish a configuration management plan in which all changes are tracked and maintained
- Baseline the problem statement
- Base line the software project management plan for the construction phase
- Demonstrate that the architecture supports the requirements at a reasonable cost in a reasonable time

Question: Why does the Unified process not recommend the establishment of a configuration management plan during the inception phase?

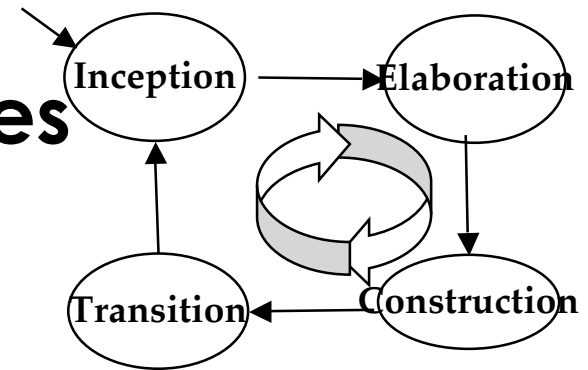
# Elaboration Phase: Activities

- Elaborate the problem statement (“vision”) by working out the critical use cases that drive technical and managerial decisions.
- Elaborate the infrastructure.
- Tailor the software process for the construction stage, identify tools.
- Establish intermediate milestones and evaluation criteria for these milestones.
- Identify buy/build (“make/buy”) problems and make decisions.
- Identify lessons learned from the inception phase to redesign the software architecture if necessary (“always necessary”:-)

# Elaboration Phase: Evaluation Criteria

- Apply the following questions to the results of the inception phase:
  - Is the problem statement stable?
  - Is the architecture stable?
  - Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?
  - Is the construction plan credible? By what claims is it backed up?
  - Do all stakeholders (project participants) agree that the vision expressed in the problem can be met if the current plan is executed?
  - Are actual resource expenditures versus planned expenditures so far acceptable?

# Construction Phase: Objectives



- Minimize development costs by optimizing resources and avoiding unnecessary “scrap and rework”
- Achieve adequate quality as rapidly as practical
- Achieve useful version (alpha, beta, and other test releases)

# Construction Phase: Activities

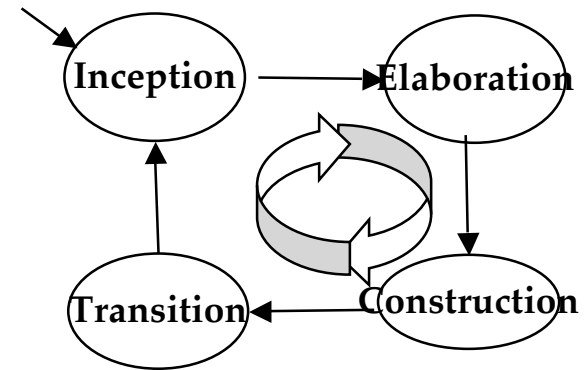
- Resource management, control and process optimization
- Complete component development and testing against evaluation criteria
- Assessment of product releases against acceptance criteria



# Construction Phase: Evaluation Criteria

- Apply the following questions to the results of the construction phase:
  - Is the product baseline *mature* enough to be deployed in the user community?
    - Existing faults are not obstacles to do the release
  - Is the product baseline *stable* enough to be deployed in the user community?
    - Pending changes are not obstacles to do the release
  - Are the stakeholders ready for the transition of the software system to the user community?
  - Are actual resource expenditures versus planned expenditures so far acceptable?

# Transition Phase



- The transition phase is entered when a baseline is mature
  - A usable subset of the system has been built with acceptable quality levels and user documents
  - It can be deployed to the user community
- For some projects the transition phase means the starting point for another version of the software system
- For other projects the transition phase means the complete delivery of the software system to a third party responsible for operation, maintenance and enhancement.

# Transition Phase: Objectives

- Achieve independence of user (users can support themselves)
- Deployment baseline is complete and consistent with the criteria in the project agreement
- The final baseline can be built as rapidly and cost-effectively as possible.



# Transition Phase: Activities

- Synchronization and integration of concurrent development increments into one consistent deployment baseline
- Commercial packaging and production
- Sales rollout kit development
- Field personnel training
- Test of deployment baseline against the acceptance criteria.

# Transition Phase: Evaluation Criteria

- Is the user satisfied?
- Are actual resource expenditures versus planned expenditures so far acceptable?

# Iterations in the Rationale Unified Process

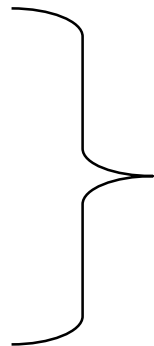
- Each of the four phases introduced so far (inception, elaboration, construction, transition) consists of one or more iterations
- An **iteration** represents a set of activities for which there is a milestone (“well-defined intermediate event”)
  - The scope and results of the iteration are captured via work products (called artifacts in the UP).

# Phase vs. Iteration

- A *phase* creates a formal, stake-holder approved version of artifacts
  - It leads to a “major milestone”
  - Phase to phase transition:
    - triggered by a significant business decision (not by the completion of a software development activity)
- An *iteration* creates an informal, internally controlled version of artifacts
  - It leads to a “minor milestone”
  - Iteration to iteration transition:
    - Triggered by a specific software development activity.

# Artifact Sets in the Unified Process

- **Artifact:** A work product in a uniform representation format (natural language, UML, Java, binary code,...)
- **Artifact set:**
  - A set of artifacts developed and reviewed as a single entity
- The Unified Process distinguishes five artifact sets
  - Management set
  - Requirements set
  - Design set
  - Implementation set
  - Deployment set

 **Also called the engineering set.**

# Artifact Sets in the Unified Process

## Engineering Set

<b>Requirements Set</b>	<b>Design Set</b>	<b>Implementation Set</b>	<b>Deployment Set</b>
<ol style="list-style-type: none"><li>1. Vision document</li><li>2. Requirements model(s)</li></ol>	<ol style="list-style-type: none"><li>1. Design model(s)</li><li>2. Test model</li><li>3. Software architecture</li></ol>	<ol style="list-style-type: none"><li>1. Source code baselines</li><li>2. Compile-time files</li><li>3. Component executables</li></ol>	<ol style="list-style-type: none"><li>1. Integrated product executable</li><li>2. Run-time files</li><li>3. User documentation</li></ol>

## Management Set

### Planning Artifacts

1. Software Project Management Plan (SPMP)
2. Software Configuration Management Plan (SCMP)
3. Work breakdown structure
4. Business Case
5. Release specifications

### Operational Artifacts

1. Release descriptions
2. Status assessments
3. Change Management database
4. Deployment documents
5. Environment.

# Representation of Artifact Sets (1)

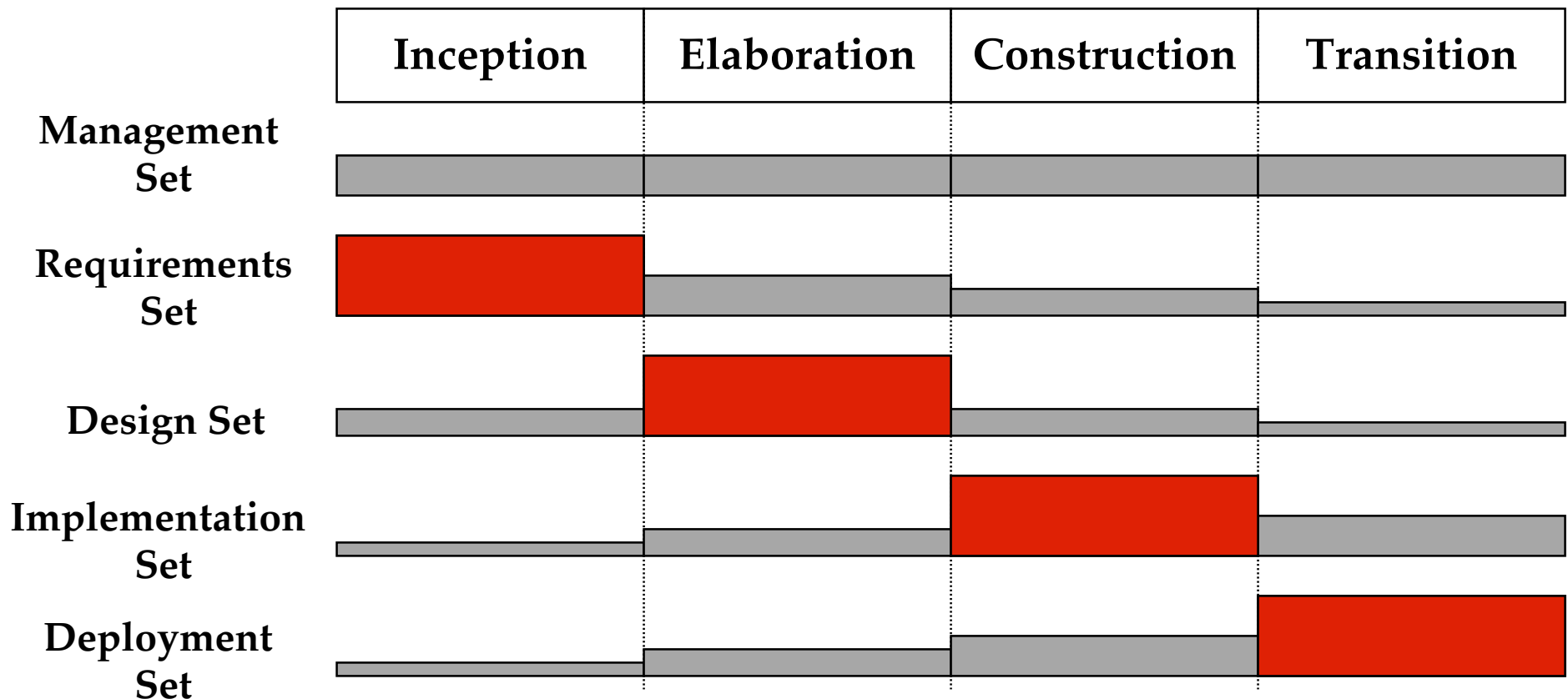
- **Management Set**
  - Goal: Capture plans, processes, objectives, acceptance criteria
  - Notation: Ad hoc text, graphics, textual use cases.
- **Requirements set**
  - Goal: Capture problem in language of problem domain
  - Notation: Structured text, UML models
- **Design set**
  - Goal: Capture the engineering blueprints
  - Notation: Structured text, UML models.

# Rationale for Selection of Artifact Sets (2)

- **Implementation set**
  - Goal: Capture the building blocks of the solution domain in human-readable format
  - Notation: Programming language
- **Deployment set**
  - Goal: Capture the solution in machine-readable format
  - Notation: Machine language.

# Life-cycle Focus on Artifact Sets

- Each artifact set is the predominant focus in one stage of the unified process.

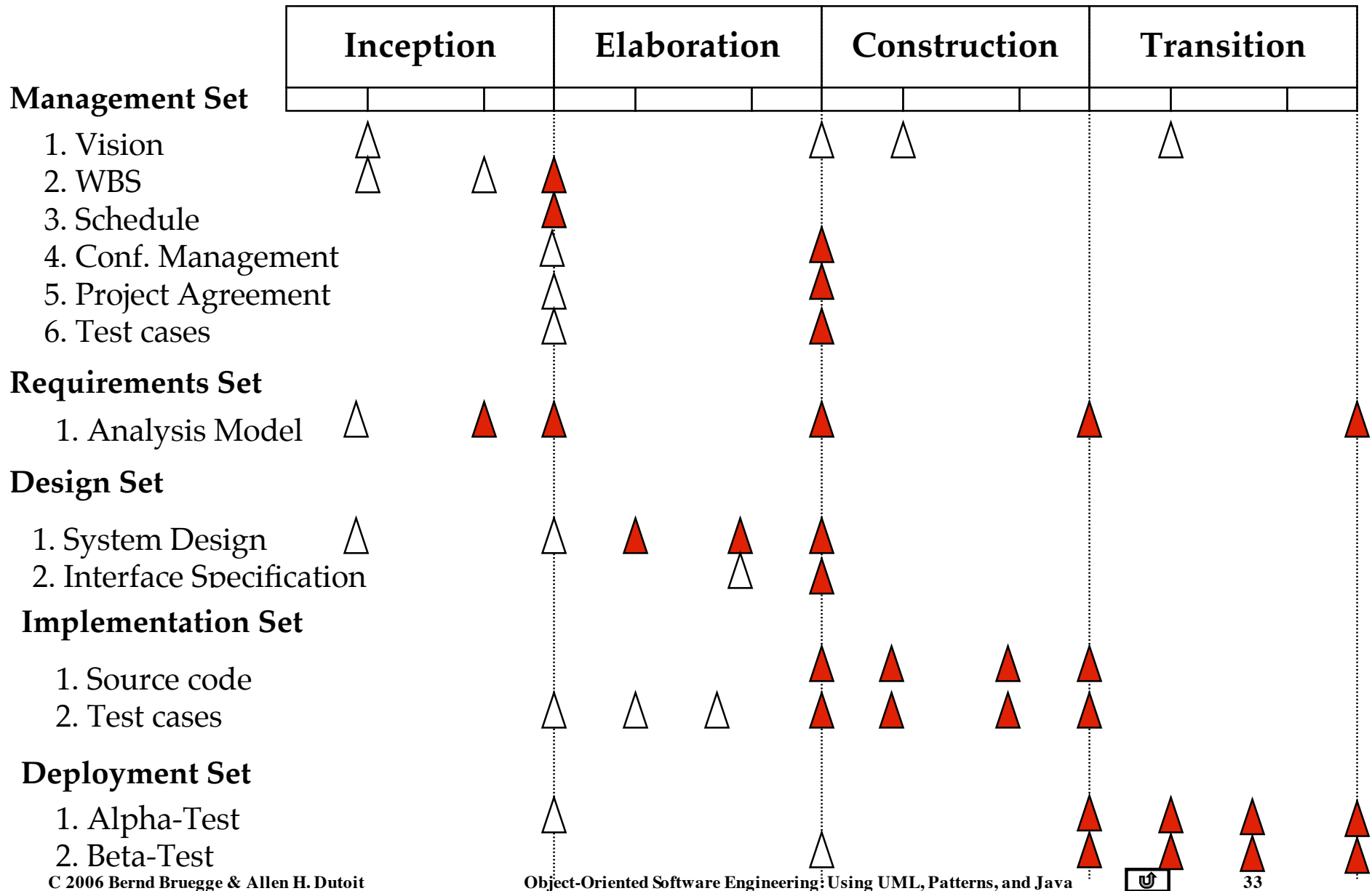


# Managing the Artifact Sets

- Some artifacts need to be updated at each major milestone (after a phase)
- Other artifacts must be updated at each minor milestone (after an iteration)
- **Artifact set roadmap**
  - Visualization of the updates of artifacts across the software life-cycle
- The software project manager is responsible for managing the artifact set roadmap
  - Artifact set roadmap: Focus on models
  - Artifact set roadmap: Focus on documents.

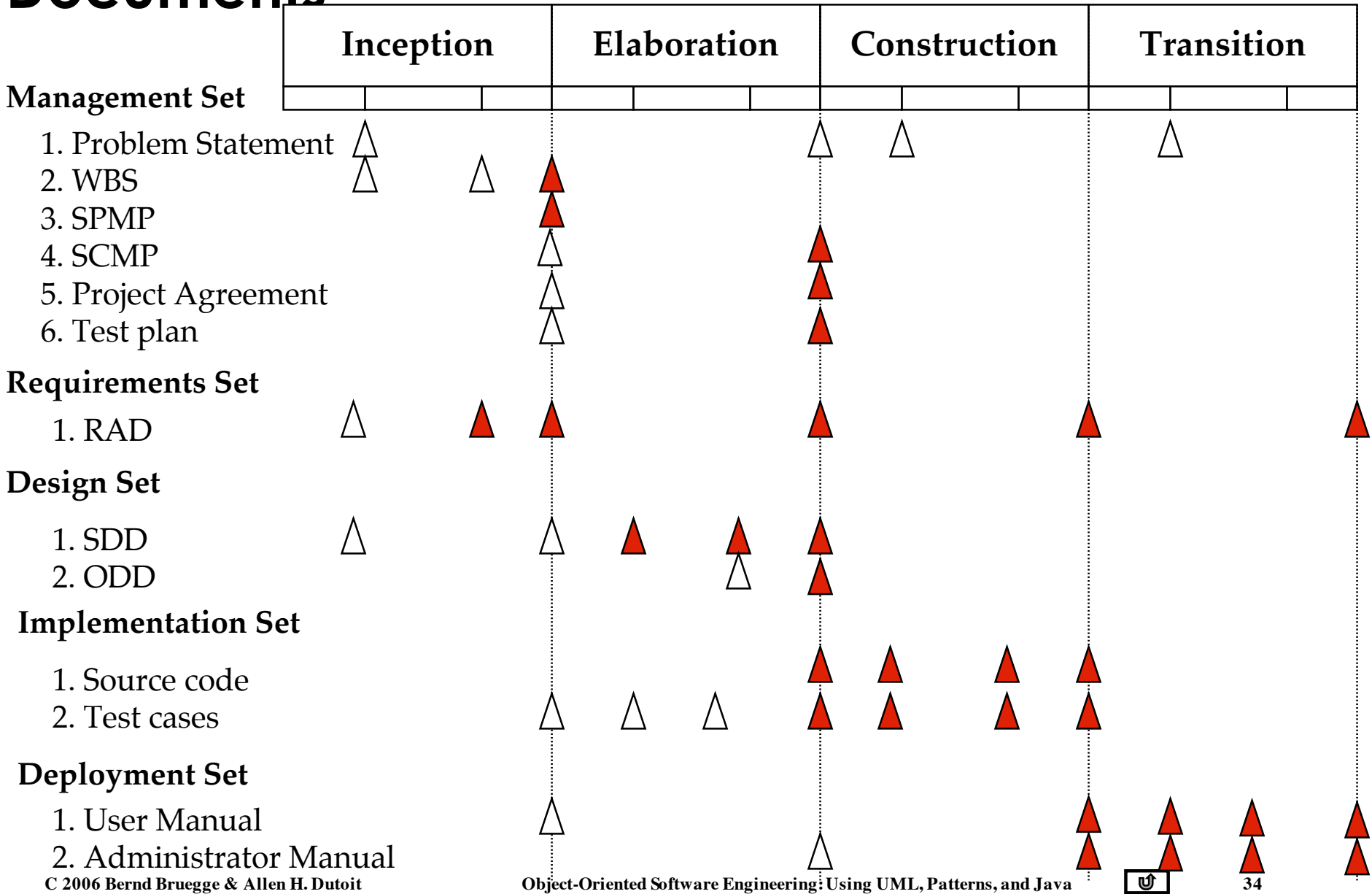
# Artifact Set Roadmap: Focus on Models

△ Informal  
▲ Baseline



# Artifact Set Roadmap: Focus on Documents

△ Informal  
▲ Baseline



# Models vs. Documents

- Many software project managers pay too much attention on the production of documents
- **Documentation-driven approach**
  - The production of the documents drives the milestones and deadlines
- **Model-driven approach**
  - The production of the models drive the milestones deadlines
- Main goal of a software development project:
  - Creation of models and construction of the software system
- The purpose of documentation is to support this goal.

# Historical Reasons for Documentation-Driven Approach

- People wanted to review information, but did not understand the language of the artifact
- People wanted to review information, but did not have access to the tools to view the information
- No rigorous engineering methods and languages were available for analysis and design models
  - Therefore paper documents with ad hoc text were used
- Conventional languages for implementation and deployment were highly cryptic
  - A more human-readable format was needed
- Managers needed “status”
  - Documents seemed to be a good mechanism for demonstrating progress.

# Artifact-Driven Approach

- Provide templates for documents at the start of the project
- Instantiate documents automatically from these templates
  - Enrich them with modeling and artifact information generated during the project
- Tools automatically generate documents from the models. Examples:
  - Generation of analysis and design documents (Commercial CASE tools)
  - Generation of the interface specification (Javadoc)
  - Test case generation (JUnit)
  - Schedule generation (Microsoft Project).

# Micro Processes in the Unified Process

- The Unified Process distinguishes between macro and micro process:
  - The macro process models the software lifecycle
  - The micro process models activities that produce artifacts
- The micro processes are also called workflows in the Unified Process.

# Workflows in the Unified Process

- Management workflow
- Environment workflow
- Requirements workflow
- Design workflow
- Implementation workflow
- Assessment workflow
- Deployment workflow.



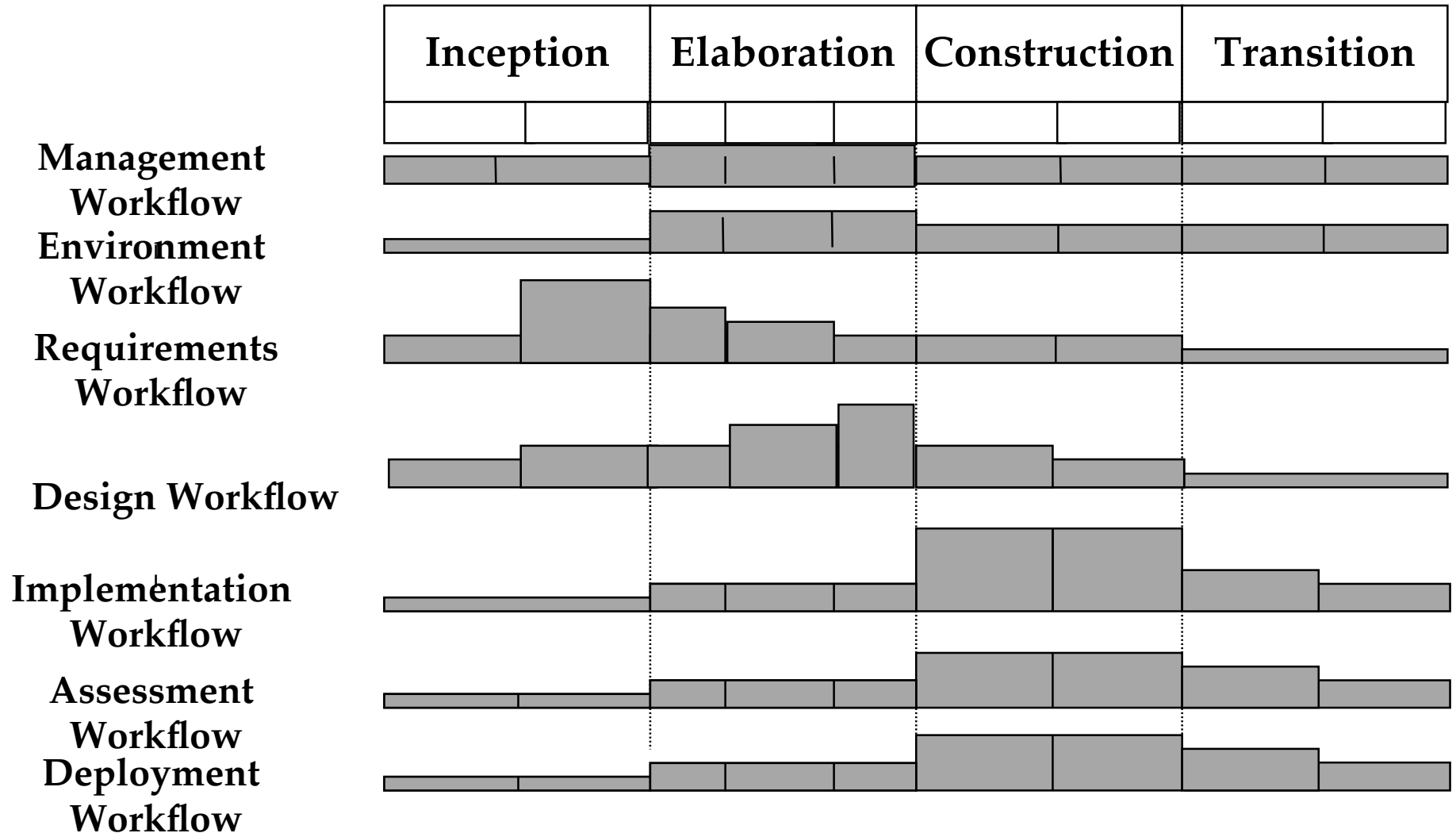
# Workflows in the Unified Process

- **Management workflow**
  - Planning the project (Problem statement, SPMP, SCMP, test plan)
- **Environment workflow**
  - Automation of process and maintenance environment. Setup of infrastructure (Communication, configuration management, ...)
- **Requirements workflow**
  - Analysis of application domain and creation of requirements artifacts (analysis model)
- **Design workflow**
  - Creation of solution and design artifacts (system design model, object design model).

# Workflows in the Unified Process (2)

- **Implementation workflow**
  - Implementation of solution, source code testing, maintenance of implementation and deployment artifacts (source code)
- **Assessment workflow**
  - Assess process and products (reviews, walkthroughs, inspections, testing...)
- **Deployment workflow**
  - Transition the software system to the end user.

# Workflows work across Phases



- Workflows create artifacts (documents, models)
- Workflows consist of one or more iterations per phase.

# Managing Projects in the Unified Process

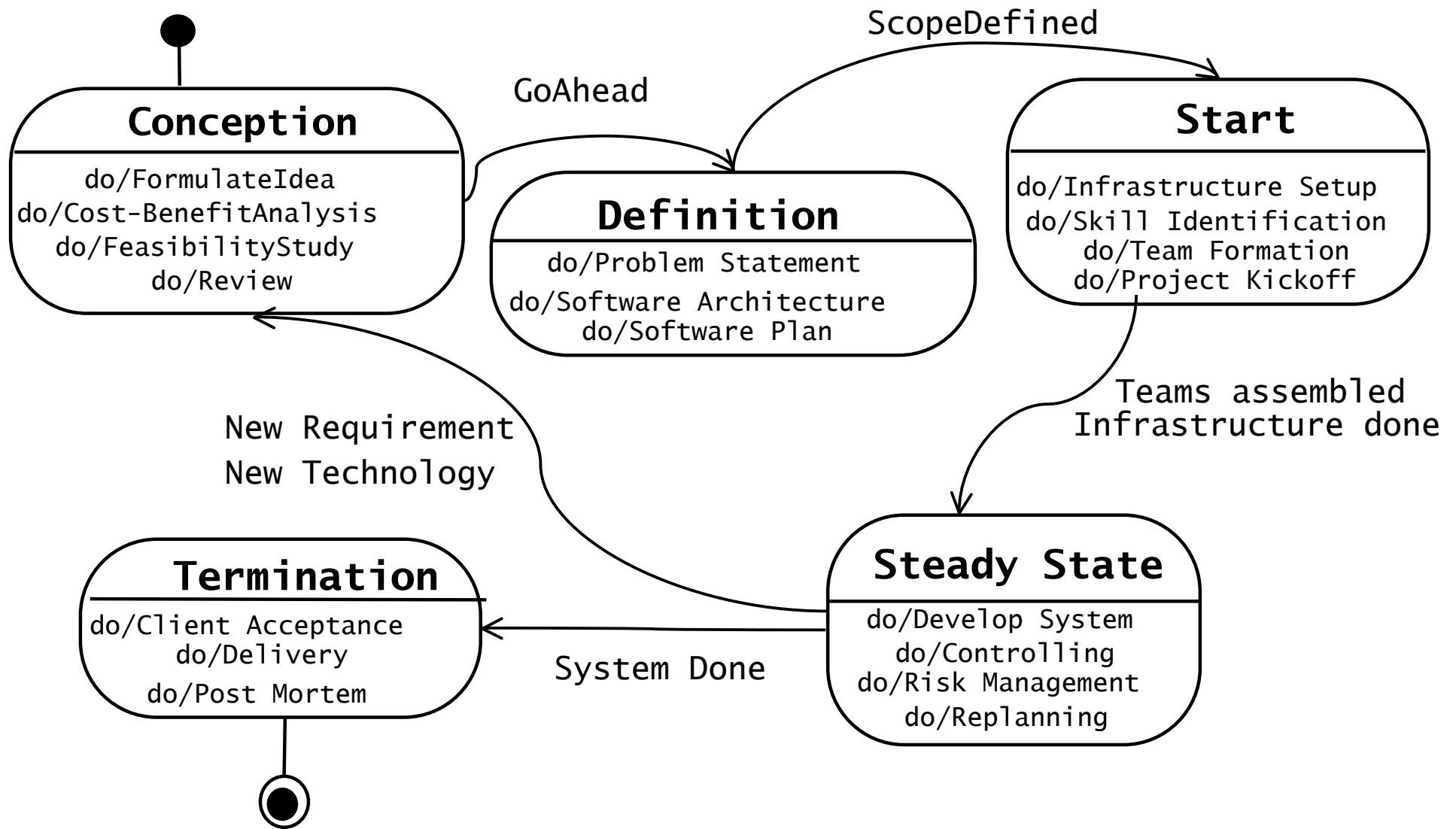
- How should we manage the construction of software systems with the Unified Process?
- Approach
  - Treat the development of a software system with the Unified Process as a set of several iterations
    - Some of these can be scheduled in parallel, others have to occur in sequence
  - Define a single project for each iteration
  - Establish work break down structures for each of the 7 workflows.

# Project Phases vs. Unified Process Phases

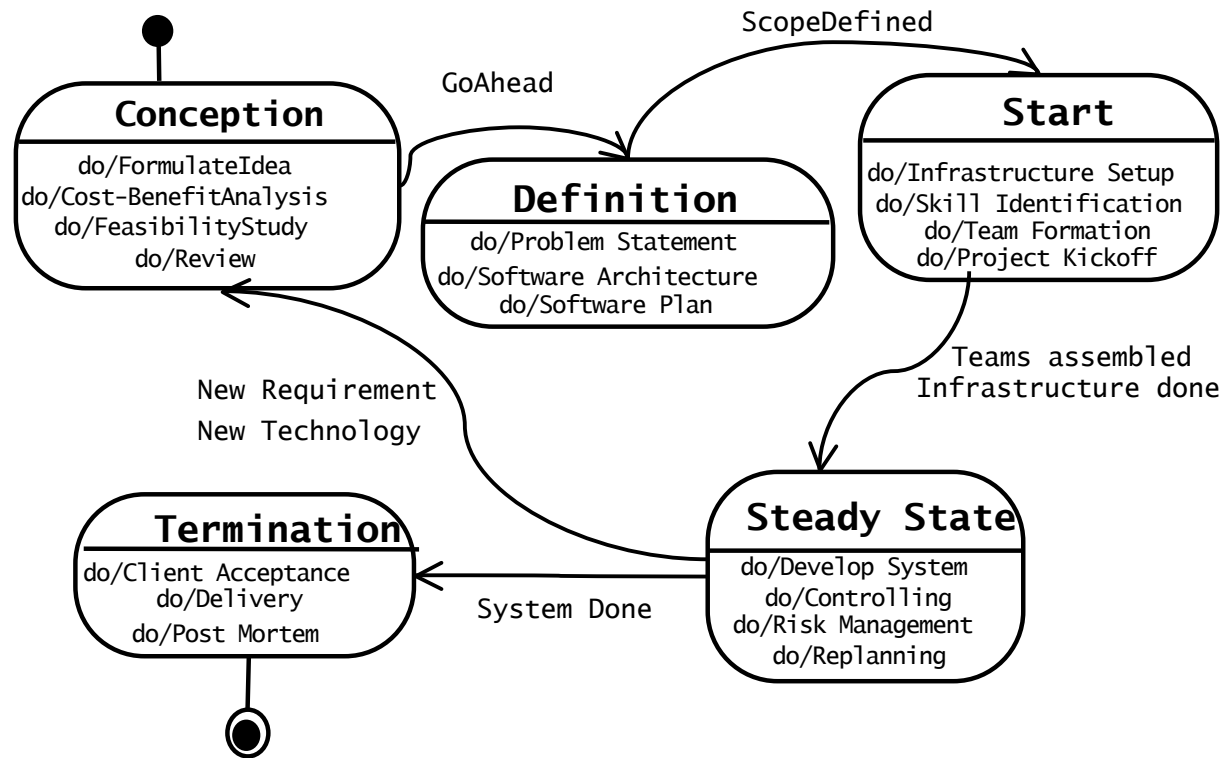
- Every project has at least 5 states
  - Conceiving: The idea is born
  - Defining: A plan is developed
  - Starting: Teams are formed
  - Performing: The work is being done
  - Closing: The project is finished.



# Phases of a Software Project

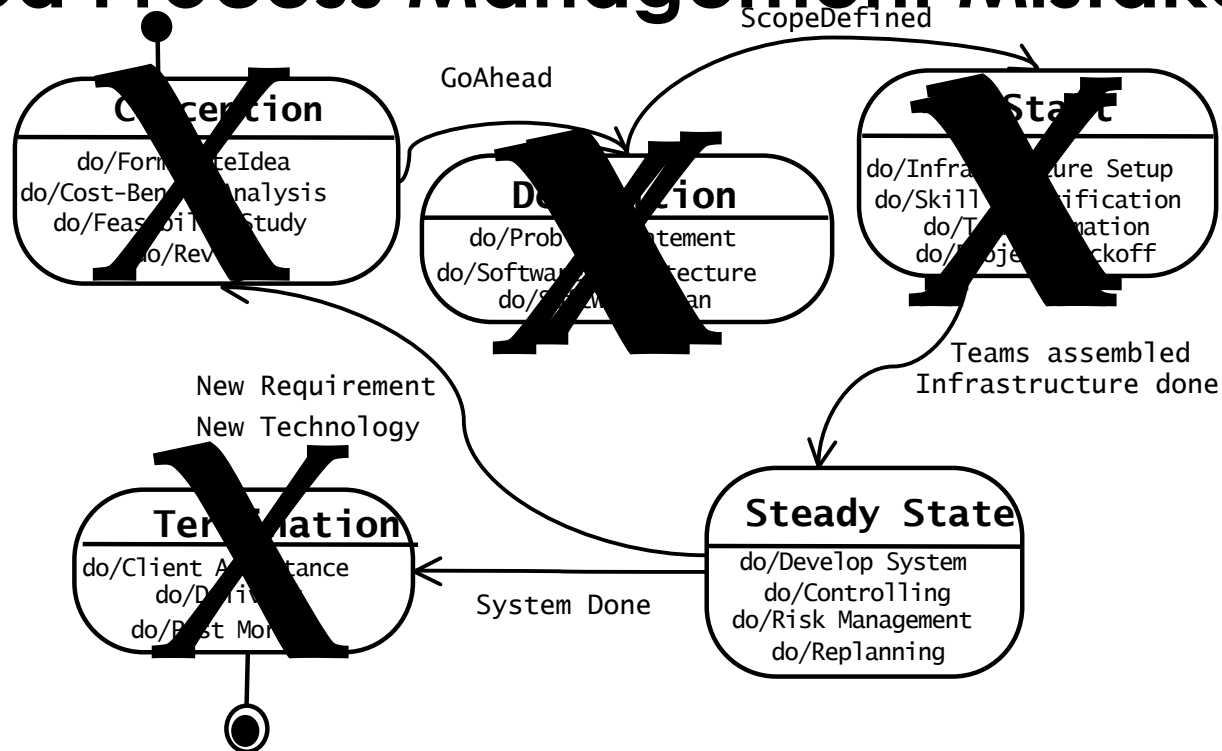


# Project Phases vs. Unified Process Phases



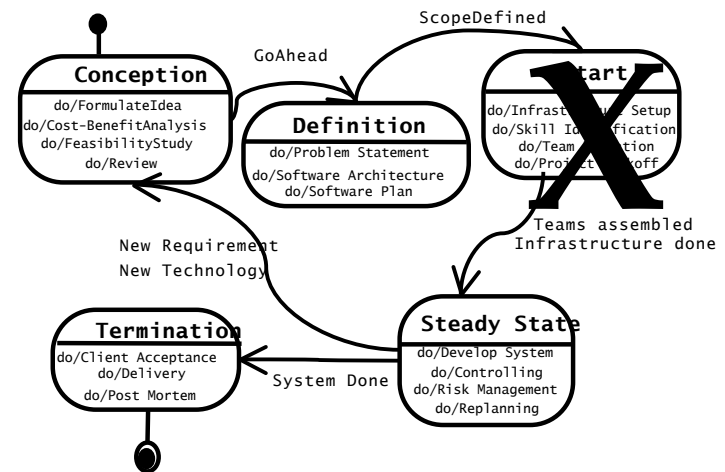
Each iteration in the unified process phases  
 Inception, Elaboration, Construction, Transition  
 should go through each of these 5 project phases!

# Unified Process Management Mistakes



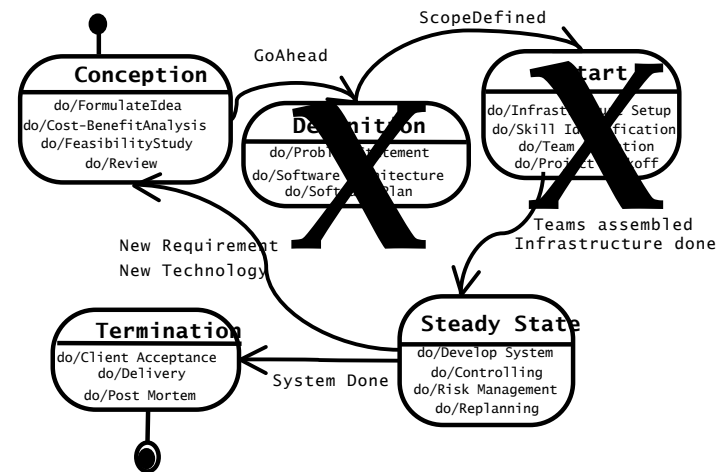
- Project manager skips the start phase
- Project manager skips the definition and start phase
- Project manager jumps straight to the steady state phase after joining the project late
- Project manager cancels the termination phase.

# Mistake: Skipping the Start Phase



- Main reason: Time pressure
- Reasons for start phase
  - Inform stakeholders that the project has been approved and when work will start
  - Confirm that stakeholders are able to support the project
  - Reevaluate and reconfirm work packages with developers
  - Explain your role as manager to stakeholders and developers.

# Mistake: Skipping Definition and Start Phase



- **Known territory argument**
  - “I have done this before, no need to waste time”
    - Even though a project may be similar to an earlier one, some things are always different
- **Unknown territory argument**
  - “My project is different from anything I have ever done before, so what good is it to plan?”
    - It is better to create a map if you are attempting to travel into unknown territory.

# Problem: Joining a Project Late

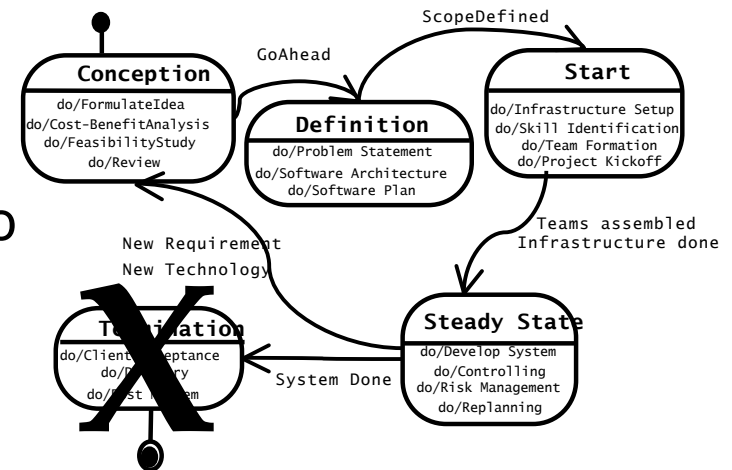
Joining a project late is not that uncommon

- Often the planning has been performed by another person, usually a high level manager, and you are asked to take the project over
- Or the project is in such a bad state, that the current project manager needs to be replaced
- Reason to jump right into steady state phase
  - “The plan has already been developed, so why should I go back to the conception and definition phases?”
- Reasons to reevaluate the conception and definition phase:
  1. See if you can identify any issues that may have been overlooked
  2. Try to understand the rationale behind the plan and to decide if you feel the plan is achievable.

# Mistake: No Termination Phase

- Reasons for skipping or not completing the termination phase:

- You leave a project to move on right to the next one. (Because you are a successful manager:-)
- Scarce resources and short deadlines
- A new project is always more challenging than wrapping up an old one



- Take the time to ensure that all tasks are completed or identified as open issues:

- Otherwise you never really know how successful your project was

- Try to learn from your mistakes (“lessons learned”):

- If you don't, you will make the the same mistakes again, and may even fail.

# Summary

- **Unified Process:** Iterative software lifecycle model
  - Emphasis on early construction of a software architecture
  - Emphasis on early demonstrations of the system
- **Definitions**
  - **Phase:** Status of the software system.
    - 4 phases: Inception, Elaboration, Construction, Transition
  - **Workflow:** Mostly sequential activity that produces artifacts
    - 7 workflows: Management, environment, requirements, design, implementation, assessment, deployment.
    - 5 artifact sets: Management set, requirements set, design set, implementation set, deployment set
  - **Iteration:** Repetition within a workflow.
- A unified process iteration should be treated as a software project.

# Additional References

- Walker Royce
  - Software Project Management, Addison-Wesley, 1998.
- Ivar Jacobsen, Grady Booch & James Rumbaugh
  - The Unified Software Development Process, Addison Wesley, 1999.
- Jim Arlow and Ila Neustadt
  - UML and the Unified Process: Practical Object-Oriented Analysis and Design, Addison Wesley, 2002.
- Philippe Kruchten
  - Rational Unified Process, Addison-Wesley, 2000.

# Backup and Example Slides



# Component Based Software Development

- Buy
  - Commercial of the shelf components (COTS), reusable objects, ...
- Build
  - Custom development, build everything from scratch,...
- Comparison: Buy vs. Build



# Commercial Components (“Buy”)

- ☺ Predictable license costs
- ☺ Broadly used, mature technology
- ☺ Available now
- ☺ Dedicated support organization
- ☺ Hardware/software independence (sometimes)
- ☺ Rich in functionality
- ☹ Frequent upgrades
- ☹ Up-front license fees
- ☹ Recurring maintenance fees
- ☹ Dependency on vendor
- ☹ Run-time efficiency sacrifices
- ☹ Functionality constraints
- ☹ Integration not always trivial
- ☹ No control over upgrades and maintenance
- ☹ Unnecessary features that consume extra resources
- ☹ Often inadequate reliability and stability
- ☹ Multiple-vendor incompatibilities.

# Custom Components (“Build”)

- ☺ Complete change freedom
- ☺ Smaller, often simpler implementations
- ☺ Often better performance
- ☺ Control of development and enhancement
- ☹ Expensive, unpredictable development
- ☹ Unpredictable availability date
- ☹ Undefined maintenance model
- ☹ Often immature and fragile
- ☹ Single-platform dependency
- ☹ Drain on expert resources.

# Model the Unified Process: System Design (1)

- Design Goals:
  - High performance, dependability, low cost, maintainability, usability
- Subsystems:
  - The workflows Management, Environment, Requirements, Design, Implementation, Assessment, Deployment
- Hardware/Software mapping:
  - Each subsystem is running on its own node.
- Concurrency:
  - The threads can run concurrently.
- Global control flow:
  - Event-driven. The subsystems communicate via events. Typical events are: „Requirement has changed“, „Review comments available“, „Time has expired“)

# Model the Unified Process: System Design (2)

- Persistent Data:
  - Vision, Process Model, Configuration Items, Analysis Model, System Design Model, Object Design Model, Communication data.
- Access control:
  - Stakeholders (End users, managers, customers, developers, ...) have access to the persistent data with access rights defined dynamically by environment workflow.
- Boundary Conditions
  - Startup of workflows: All workflows start simultaneously
  - Steady state of workflows: Workflows wake up on an event, process the event, and go to sleep afterwards.
  - Terminal conditions of workflows: A risk has occurred that cannot be dealt with

# Model the Unified Process (Analysis)

- Inputs:
  - Problem Statement
- Functional Requirements:
  - Top level use case: Develop software system that implements the problem statement.
- Outputs:
  - Requirements analysis document
  - Software project management plan
  - Software configuration management plan
  - System design document
  - Object design document
  - Test plan and test cases
  - Source code
  - User manual and administrator manual



# Lifecycle Improvement

- 3 Possibilities to improve a Software Lifecycle Model
- **Quality improvement:** We take an n-activity process and improve the efficiency of each step
  - Example: TQM (Total Quality Management)
- **Overhead reduction:** We take an n-step process and eliminate some of the steps
  - Example: Extreme Programming
- **Concurrency:** We take an n-step process and parallelize some of the activities or use more concurrency in the resources being used
  - Example: Unified Process.