Unit 2: Software Primitives and Process Management Framework

B.E. Software VIII Semester

Gandaki Engineering College of Science

Lamachaur, Pokhara

Software Process Life Cycle Phases

- Characteristic of a successful software development process is the well-defined separation between "research and development" activities and "production" activities. Most unsuccessful projects exhibit one of the following characteristics:
  - An overemphasis on research and development
  - An overemphasis on production.

Software Process Life Cycle Phases

- Successful modern projects-and even successful projects developed under the conventional process-tend to have a very well-defined project milestone when there is a noticeable transition from a research attitude to a production attitude.
- A modern software development process must be defined to support the following:
  - Evolution of the plans, requirements, and architecture, together with well defined synchronization points
  - Risk management and objective measures of progress and quality
  - Evolution of system capabilities through demonstrations of increasing functionality

Engineering and Production Stages

- To achieve economies of scale and higher returns on investment, we must move toward a software manufacturing process driven by technological improvements in process automation and component-based development.
- Two stages of the life cycle are:
- ***The engineering stage***, *driven by less predictable but smaller teams doing design and synthesis activities*
- ***The production stage***, *driven by more predictable but larger teams doing construction, test, and deployment activities*

Engineering and Production Stages

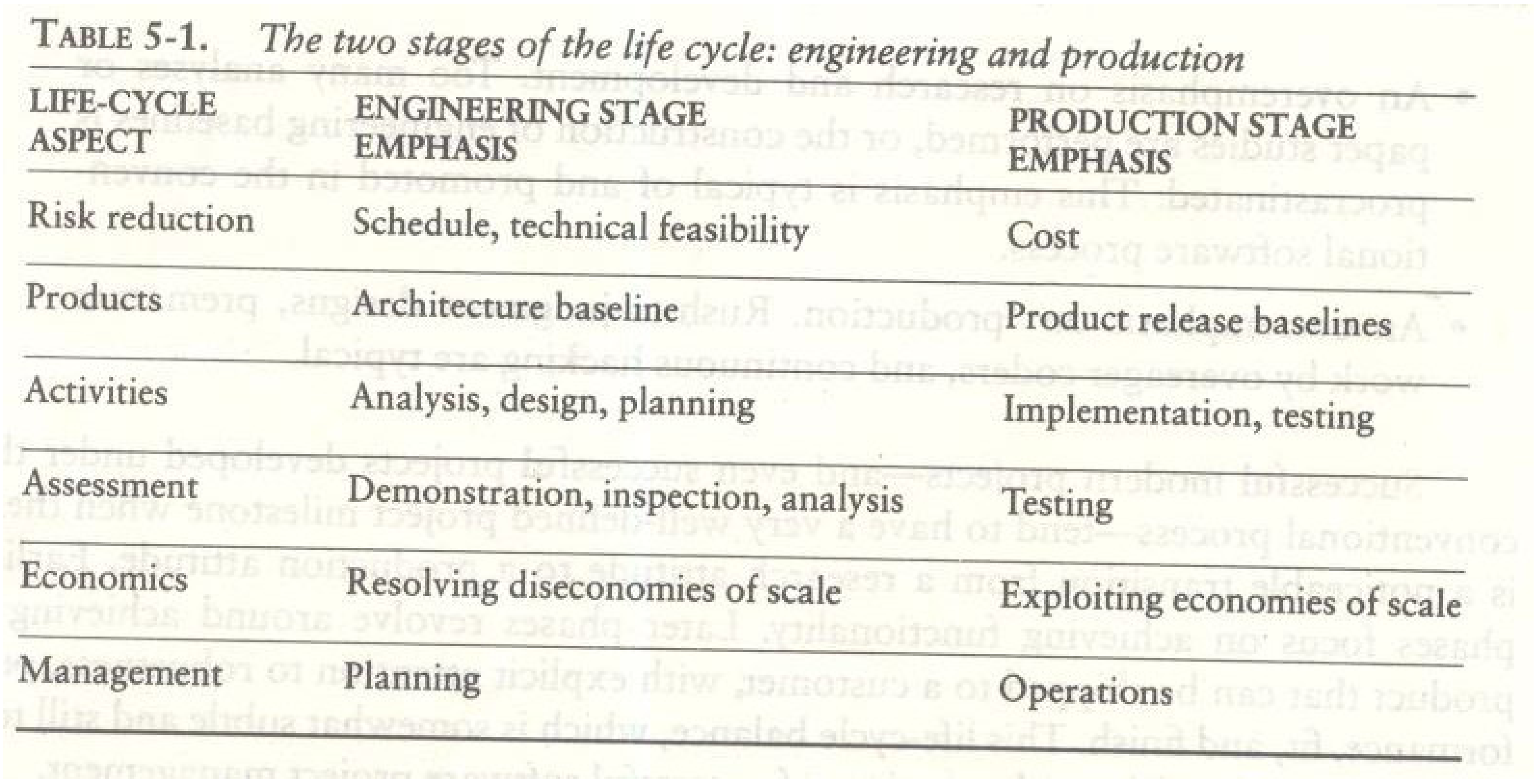


TABLE 5-1. *The two stages of the life cycle: engineering and production*

LIFE-CYCLE ASPECT	ENGINEERING STAGE EMPHASIS	PRODUCTION STAGE EMPHASIS
Risk reduction	Schedule, technical feasibility	Cost
Products	Architecture baseline	Product release baselines
Activities	Analysis, design, planning	Implementation, testing
Assessment	Demonstration, inspection, analysis	Testing
Economics	Resolving diseconomies of scale	Exploiting economies of scale
Management	Planning	Operations

Engineering and Production Stages

- The transition between engineering and production is a crucial event for the various stakeholders.
- The production plan has been agreed upon, and there is a good enough understanding of the problem and the solution that all stakeholders can make a firm commitment to go ahead with production.
- Engineering stage is decomposed into two distinct phases, inception and elaboration, and the production stage into construction and transition.

Engineering and Production Stages

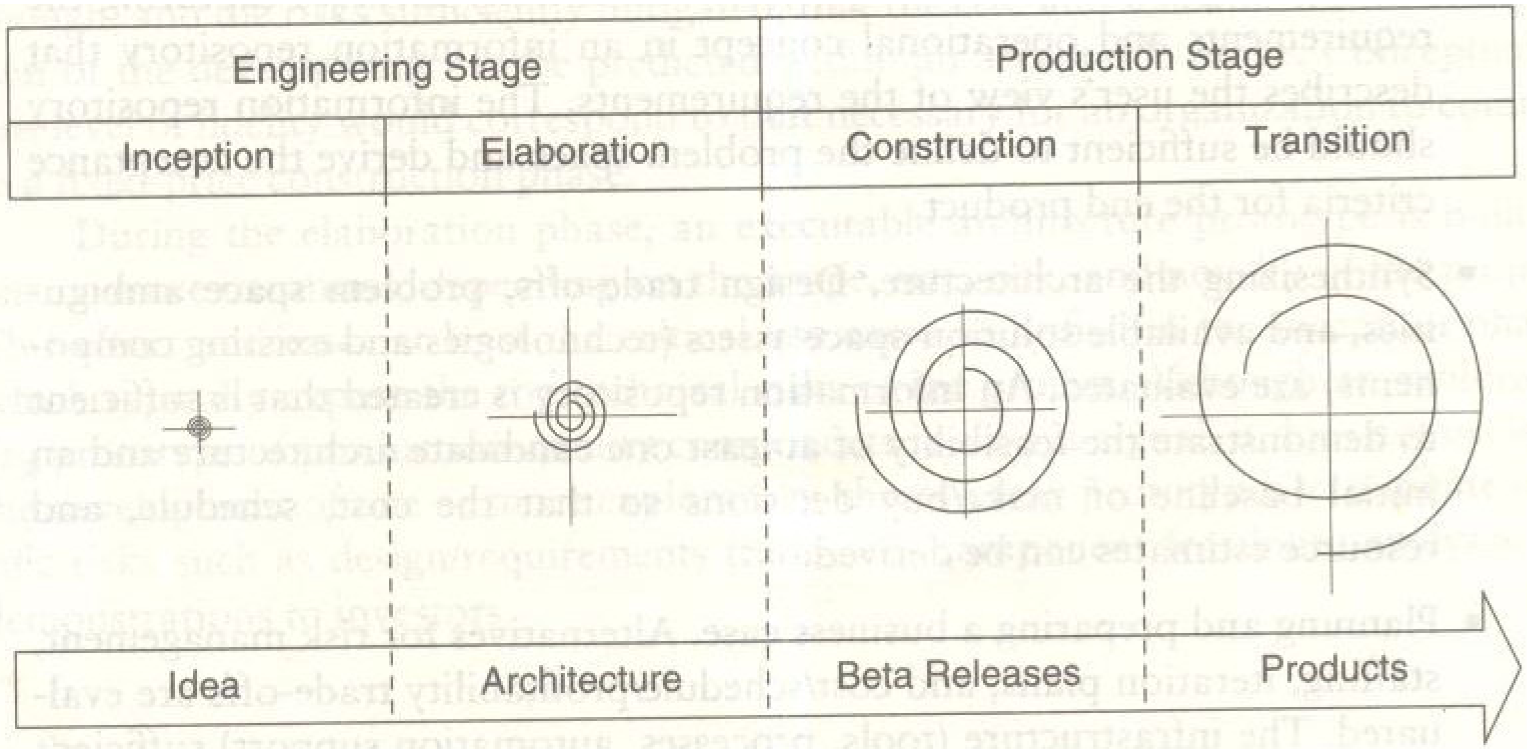


FIGURE 5-1. *The phases of the life-cycle process*

Inception Phase

- The overriding goal of the inception phase is to achieve concurrence among stakeholders on the life-cycle objectives for the project.
- **Primary Objectives:**
  - Establishing the project's software scope and boundary conditions, including an operational concept, acceptance criteria, and a clear understanding of what is and is not intended to be in the product
  - Discriminating the critical use cases of the system and the primary scenarios of operation that will drive the major design trade-offs
  - Demonstrating at least one candidate architecture against some of the primary scenarios
  - Estimating the cost and schedule for the entire project (including detailed estimates for the elaboration phase)
  - Estimating potential risks (sources of unpredictability)

Inception Phase: Essential Activities

- Formulating the scope of the project. The information repository should be sufficient to define the problem space and derive the acceptance criteria for the end product.
- Synthesizing the architecture. An information repository is created that is sufficient to demonstrate the feasibility of at least one candidate architecture and an, initial baseline of make/buy decisions so that the cost, schedule, and resource estimates can be derived.
- Planning and preparing a business case. Alternatives for risk management, staffing, iteration plans, and cost/schedule/profitability trade-offs are evaluated.

Inception Phase: Evaluation Criteria

- Do all stakeholders concur on the scope definition and cost and schedule estimates?
- Are requirements understood, as evidenced by the fidelity of the critical use cases?
- Are the cost and schedule estimates, priorities, risks, and development processes credible?
- Do the depth and breadth of an architecture prototype demonstrate the preceding criteria? (The primary value of prototyping candidate architecture is to provide a vehicle for understanding the scope and assessing the credibility of the development group in solving the particular technical problem.)
- Are actual resource expenditures versus planned expenditures acceptable

Elaboration Phase

- At the end of this phase, the "engineering" is considered complete.
- The elaboration phase activities must ensure that the architecture, requirements, and plans are stable enough, and the risks sufficiently mitigated, that the cost and schedule for the completion of the development can be predicted within an acceptable range.
- During the elaboration phase, an executable architecture prototype is built in one or more iterations, depending on the scope, size, & risk.

Elaboration Phase: Primary Objectives

- Baseline the architecture as rapidly as practical (establishing a configuration-managed snapshot in which all changes are rationalized, tracked, and maintained)
- Baseline the vision
- Baseline a high-fidelity plan for the construction phase
- Demonstrating that the baseline architecture will support the vision at a reasonable cost in a reasonable time

Elaboration Phase: Essential Activities

- Elaborating the vision.
- Elaborating the process and infrastructure.
- Elaborating the architecture and selecting components.

Elaboration Phase: Primary Evaluation Criteria

- Is the vision stable?
- Is the architecture stable?
- Does the executable demonstration show that the major risk elements have been addressed and credibly resolved?
- Is the construction phase plan of sufficient fidelity, and is it backed up with a credible basis of estimate?
- Do all stakeholders agree that the current vision can be met if the current plan is executed to develop the complete system in the context of the current architecture?
- Are actual resource expenditures versus planned expenditures acceptable?

Construction Phase

- During the construction phase, all remaining components and application features are integrated into the application, and all features are thoroughly tested.
- Newly developed software is integrated where required.
- The construction phase represents a production process, in which emphasis is placed on managing resources and controlling operations to optimize costs, schedules, and quality.

Construction Phase: Primary Objectives

- Minimizing development costs by optimizing resources and avoiding unnecessary scrap and rework
- Achieving adequate quality as rapidly as practical
- Achieving useful versions (alpha, beta, and other test releases) as rapidly as practical

Construction Phase: Essential Activities

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

Construction Phase: Primary Evaluation Criteria

- Is this product baseline mature enough to be deployed in the user community? (Existing defects are not obstacles to achieving the purpose of the next release.)
- Is this product baseline stable enough to be deployed in the user community? (Pending changes are not obstacles to achieving the purpose of the next release.)
- Are the stakeholders ready for transition to the user community?
- Are actual resource expenditures versus planned expenditures acceptable?

Transition Phase

- The transition phase is entered when a baseline is mature enough to be deployed in the end-user domain.
- This phase could include any of the following activities:
  - Beta testing to validate the new system against user expectations
  - Beta testing and parallel operation relative to a legacy system it is replacing
  - Conversion of operational databases
  - Training of users and maintainers
- The transition phase concludes when the deployment baseline has achieved the complete vision.

Transition Phase: Primary Objectives

- Achieving user self-supportability
- Achieving stakeholder concurrence that deployment baselines are complete and consistent with the evaluation criteria of the vision
- Achieving final product baselines as rapidly and cost-effectively as practical

Transition Phase: Essential Activities

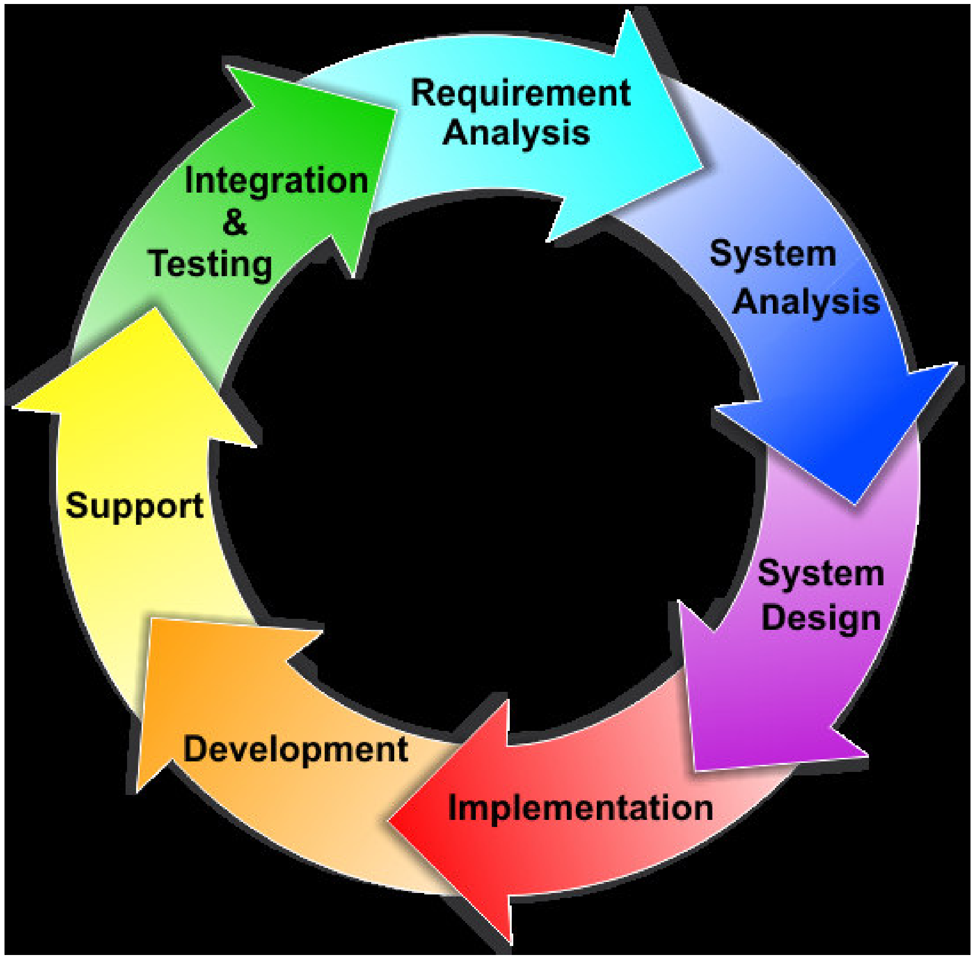
- Synchronization and integration of concurrent construction increments into consistent deployment baselines
- Deployment-specific engineering (cutover, commercial packaging and production, sales rollout kit development, field personnel training)
- Assessment of deployment baselines against the complete vision and acceptance criteria in the requirements set

Transition Phase: Evaluation Criteria

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

- Each of the four phases consists of one or more iterations in which some technical capability is produced in demonstrable form and assessed against a set of the criteria.
- The transition from one phase to the nest maps more to a significant business decision than to the completion of specific software activity.

- So, in software engineering, a software development process is the process of dividing software development work into distinct phases to improve design, product management, and project management. It is also known as a software development life cycle.


Requirement Analysis
System Analysis
System Design
Implementation
Development
Support
Integration & Testing

- SDLC is a process which defines the various stages involved in the development of software for delivering a high-quality product.
- SDLC stages cover the complete life cycle of a software i.e. from inception to retirement of the product.
- Adhering to the SDLC process leads to the development of the software in a systematic and disciplined manner.

Purpose of SDLC:

- Purpose of SDLC is to deliver a high-quality product which is as per the customer’s requirement.
- SDLC has defined its phases as, Requirement gathering, Designing, Coding, Testing, and Maintenance.
- It is important to adhere (to stick an object) to the phases to provide the Product in a systematic manner.

- **For Example:**
- A software has to be developed and a team is divided to work on a feature of the product and is allowed to work as they want.
- One of the developers decides to design first whereas the other decides to code first and the other on the documentation part.
- This will lead to project failure because of which it is necessary to have a good knowledge and understanding among the team members to deliver an expected product.

Given below are the various phases of SDLC:

- Requirement gathering
- Analysis
- Design
- Coding/Implementation
- Testing
- Final Implementation
- Testing/Deployment
- Maintenance

Elements of Analysis Model

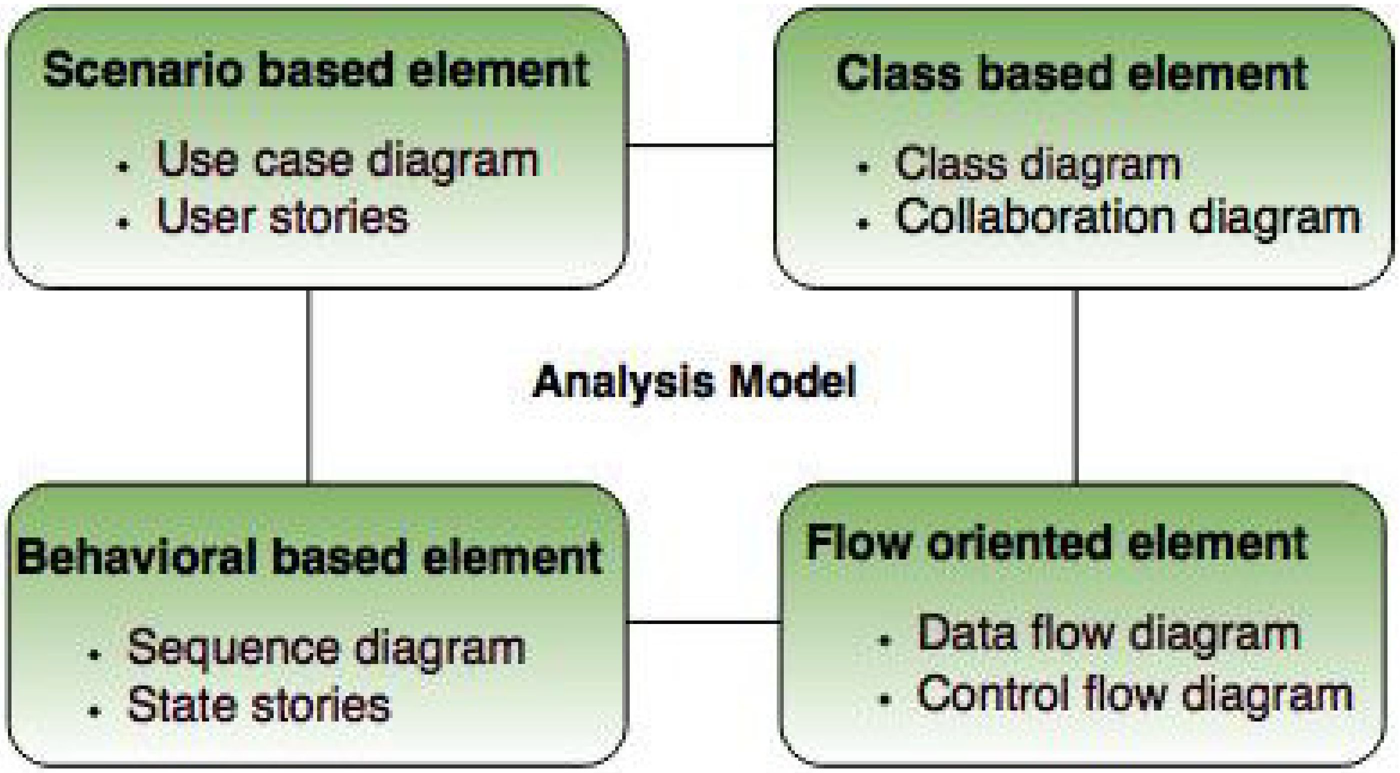


Fig. - Elements of analysis model

Elements of the Software Process

- Management elements
- Engineering elements
- Pragmatic elements

Management Elements

- The management set includes several elements :
- **Work Breakdown Structure –** budgeting and collecting costs.
- The software project manager must have insight into project costs and how they are expended.
- If the WBS is structured improperly, it can drive the evolving design in the wrong direction.

- **Business Case – provides all the** information necessary to determine whether the project is worth investing in.
- It details the expected revenue, expected cost, technical and management plans.

Release Specifications

- Typical release specification outlines are:
- **Iteration (step-by-step process) content**
- **Measurable objectives**
  - Evaluation criteria
  - Follow-through approach
- **Demonstration plan**
  - Schedule of activities
  - Team responsibilities
- **Operational scenarios (use cases demonstrated)**
  - Demonstration procedures

Software Development Plan

- Software Development Plan is the defining document for the project’s process.
- It must comply with the contract, comply with the organization standards, evolve along with the design and requirements.

Deployment

- **Deployment – depending on the project, it** could include several document subsets for transitioning the product into operational status.
- It could also include computer system operations manuals, software installation manuals, plans and procedures for cutover etc.

Environment

- **Environment – A robust development** environment must support automation of the development process.
- It should include : requirements management visual modeling document automation automated regression testing

Engineering Element of Software Process

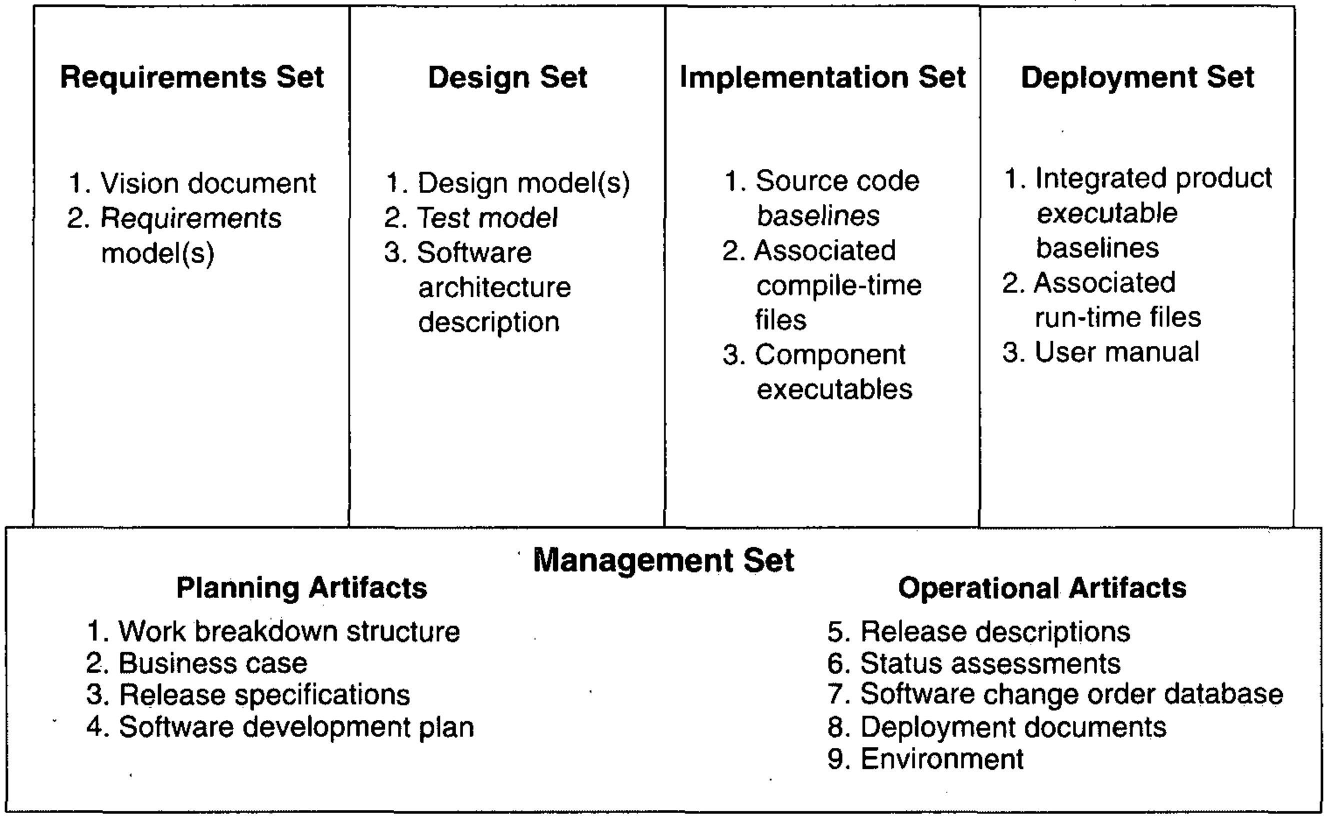
- **Software process**
- **Reuse: Not just of software, but also** sets of requirements, parts of designs, or group of test scripts.
- **Measurement: Trying to quantify project** goals to evaluate progress (for example – number of bugs per 100 lines of code)
- **Tools and Integrated Environment:** For example, CASE (Computer-aided software engineering) tools.

- **Architecture Description – it is extracted from the** design model and includes views of the design, implementation, and deployment sets sufficient to understand how the operational concept of the requirements set will be achieved.
- **Software User Manual – it should include** installation procedures, usage procedures and guidance, operational constraints, and a user interface description. (written by test team)

Pragmatic Elements

- Over the past 30 years, the quality of documents become more important than the quality of the engineering information they represented.
- The reviewer must be knowledgeable in the engineering notation.
- Human-readable engineering artifacts should use rigorous notations that are complete, consistent, and used in a self- documenting manner.
- Paper is tangible, electronic artifacts are too easy to change.
- Short documents are more useful than long ones.

Artifacts of the Process



Requirements Set	Design Set	Implementation Set	Deployment Set
1. Vision document 2. *Requirements* model(s)	1. Design model(s) 2. *Test model* 3. Software architecture description	1. Source code *baselines* 2. Associated compile-time files 3. Component executables	1. Integrated product *executable* baselines 2. Associated run-time files 3. User manual

Management Set

Planning Artifacts	Operational Artifacts
1. Work breakdown structure 2. Business case 3. Release specifications 4. Software development plan	5. Release descriptions 6. Status assessments 7. Software change order database 8. Deployment documents 9. Environment

FIGURE 6-1. *Overview of the artifact sets*

Technical and Management Perspective of Software Architecture

Software Architectures: A Management Perspective

- From a management perspective, there are three different aspects of an architecture :
- An ***architecture (the intangible design concept) is the*** design of software system, as different or similar to design of a component.
- An ***architecture baseline (the tangible artifacts) is a*** slice (portion) of information across the engineering artifact sets sufficient to satisfy all stakeholders that the vision can be achieved within the parameters of the business case (cost, profit, time, people).

- An ***architecture description (a human-readable*** representation of an architecture) is an organizes subsets of information extracted from the design set model.
- **The importance of software architecture can be summarized as follows:**
  - Architecture representations provide a basis for balancing the trade-offs between the problem space and the solution space.
  - Poor architectures and immature processes are often given as reasons for project failures.

- A mature process, an understanding of the primary requirements, and a demonstrable architecture are important prerequisites for predictable planning.
- Architecture development and process definition are the intellectual steps that map the problem to a solution without violating the constraints.

Perspective

- The model which draws on the foundation of architecture developed at *Rational Software Corporation and particularly on* Philippe Kruchten's concepts of software architecture :

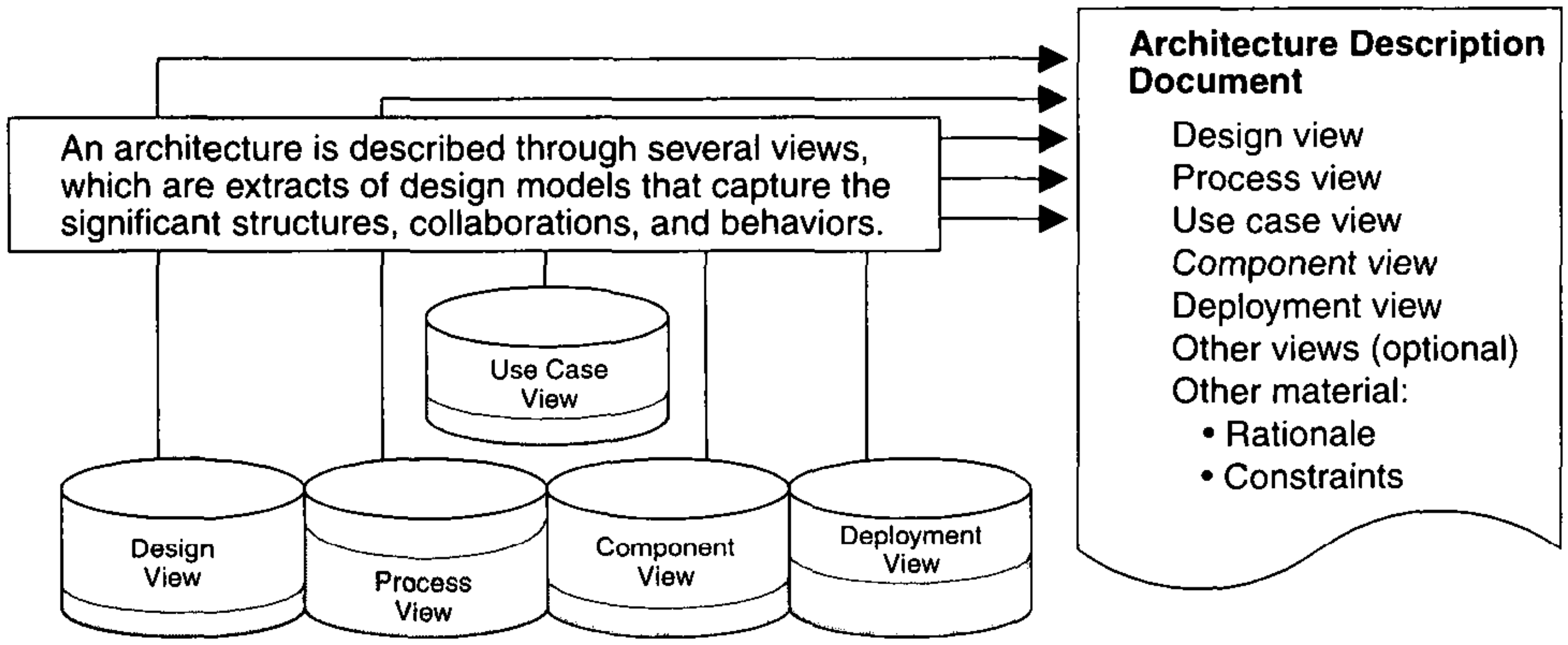

An architecture is described through several views, which are extracts of design models that capture the significant structures, collaborations, and behaviors.
Use Case View
Design View
Process View
Component View
Deployment View
Architecture Description Document
Design view
Process view
Use case view
Component view
Deployment view
Other views (optional)
Other material:
• Rationale
• Constraints

FIGURE 7-1. *Architecture, an organized and abstracted view into the design models*

- An architecture framework is defined in terms of views that are abstractions of the UML models in the design set. The design model includes the full breadth and depth of information.
- An architecture view is an abstraction of the design model; it contains only the architecturally significant information. Most real-world systems require four views: design, process, component, and deployment. The purposes of these views are as follows:
- Design:
  - describes architecturally significant structures and functions of the design model
- Process:
  - describes concurrency and control thread relationships among the design, component, and deployment views
- Component:
  - describes the structure of the implementation set
- Deployment:
  - describes the structure of the deployment set

Use Case View

- The ***use case view*** *describes how the system's* critical use cases are realized by elements of the design model. It is modeled statically using case diagrams, and dynamically using any of the UML behavioral diagrams.

Design View

- The ***design view*** *describes the architecturally significant elements of the* design model. This view, an abstraction of the design model, addresses the basic structure and functionality of the solution. It is modeled statically using class and object diagrams, and dynamically using any of the UML behavioral diagrams.

Process View

- The ***process view*** *addresses the run-time* collaboration issues involved in executing the architecture on a distributed deployment model, including the logical software network topology, interprocess communication and state management.
- This view is modeled statically using deployment diagrams, and dynamically using any of the UML behavioral diagrams.

Component View

- The *component view describes the architecturally significant elements of* the implementation set.
- This view, an abstraction of the design model, addresses the software source code realization of the system from the perspective of the project's integrators and developers, especially with regard to releases and configuration management.
- It is modeled statically using component diagrams, and dynamically using any of the UML behavioral diagrams.

Deployment View

- The *deployment view addresses the executable realization of the system,* including the allocation of logical processes in the distribution view to physical resources of the deployment network.
- It is modeled statically using deployment diagrams, and dynamically using any of the UML behavioral diagrams.

- Architecture descriptions take on different forms and styles in different organizations and domains.
- Generally, an architecture baseline should include the following:
- Requirements:
  - critical use cases, system-level quality objectives, and priority relationships among features and qualities
- Design:
  - names, attributes, structures, behaviors, groupings, and relationships of significant classes and components
- Implementation:
  - source component inventory and bill of materials (number, name, purpose, cost) of all primitive components
- Deployment:
  - executable components sufficient to demonstrate the critical use cases and the risk associated with achieving the system qualities

Software Process Workflow and Iteration Workflow

Software Process Workflows

- The next-level process description is the microprocesses, or workflows, that produce the artifacts.
- The term *workflow is used to mean a thread of cohesive and mostly sequential activities.*

Software Process Workflows

- There are seven top-level workflows:
- Management workflow:
  - controlling the process and ensuring win conditions for all stakeholders
- Environment workflow:
  - automating the process and evolving the maintenance environment
- Requirements workflow:
  - analyzing the problem space and evolving the requirements artifacts
- Design workflow:
  - modeling the solution and evolving the architecture and design artifacts
- Implementation workflow:
  - programming the components and evolving the implementation and deployment artifacts
- Assessment workflow:
  - assessing the trends in process and product quality
- Deployment workflow:
  - transitioning the end products to the user

Four basic key principles

- ***Architecture-first approach***
- ***Iterative life-cycle process***
- ***Round-trip engineering***
- ***Demonstration-based approach***

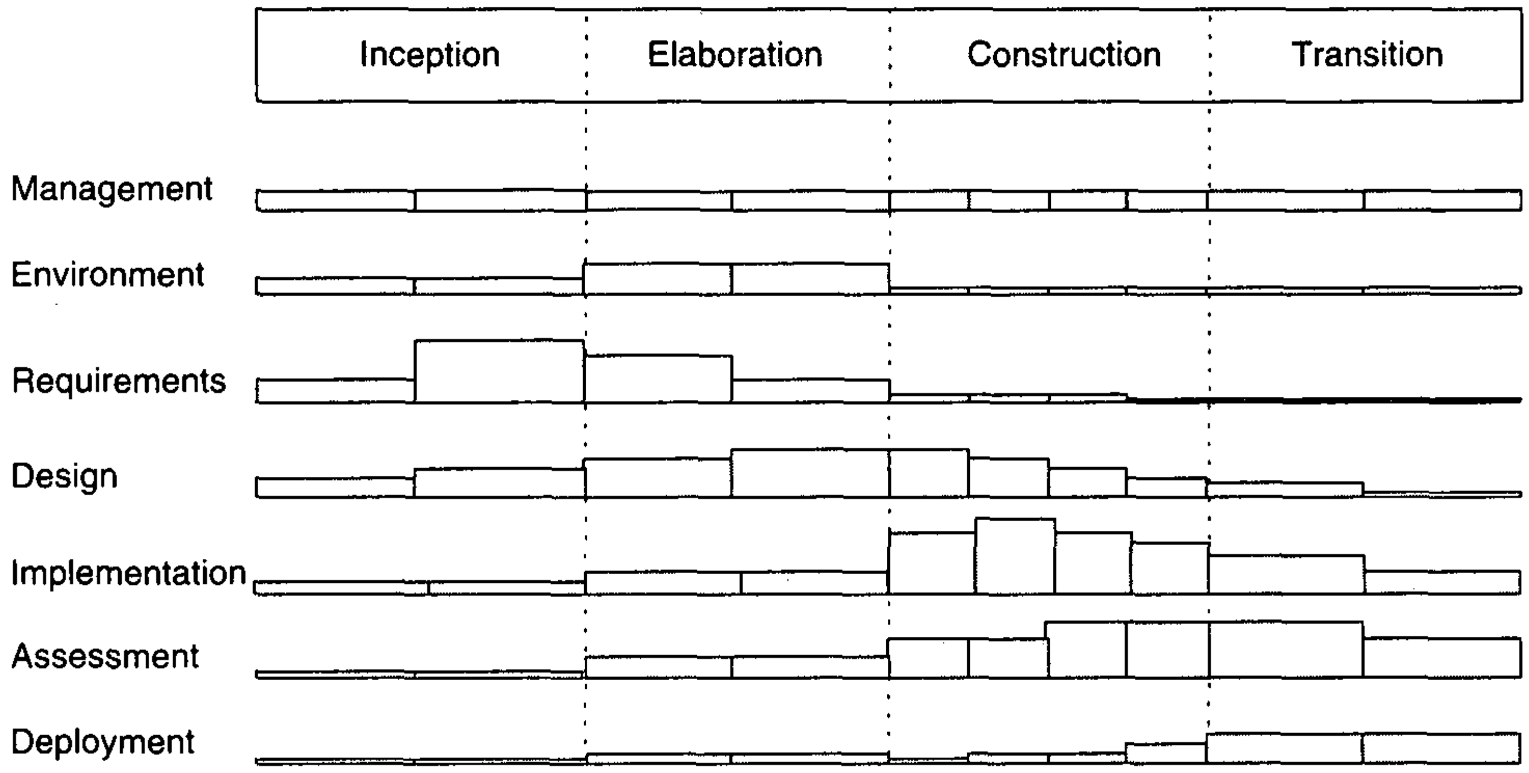


FIGURE 8-1. *Activity levels across the life-cycle phases*

Architecture-first approach

- Extensive requirements analysis, design, implementation, and assessment activities are performed before the construction phase, when full-scale implementation is the focus.
- This early life-cycle focus on implementing and testing the architecture must precede full-scale development and testing of all the components and must precede the downstream focus on completeness and quality of the entire breadth of the product features.

Iterative life-cycle process

- The activities and artifacts of any given workflow may require more than one pass to achieve adequate results.

Round-trip engineering

- Raising the environment activities to a first-class workflow is critical.
- The environment is the tangible embodiment of the project’s process, methods, and notations for producing the artifacts.

Demonstration-based approach

- Implementation and assessment activities are initiated nearly in the life-cycle, reflecting the emphasis on constructing executable subsets of the involving architecture.

The artifacts and life-cycle emphases associated with each workflow

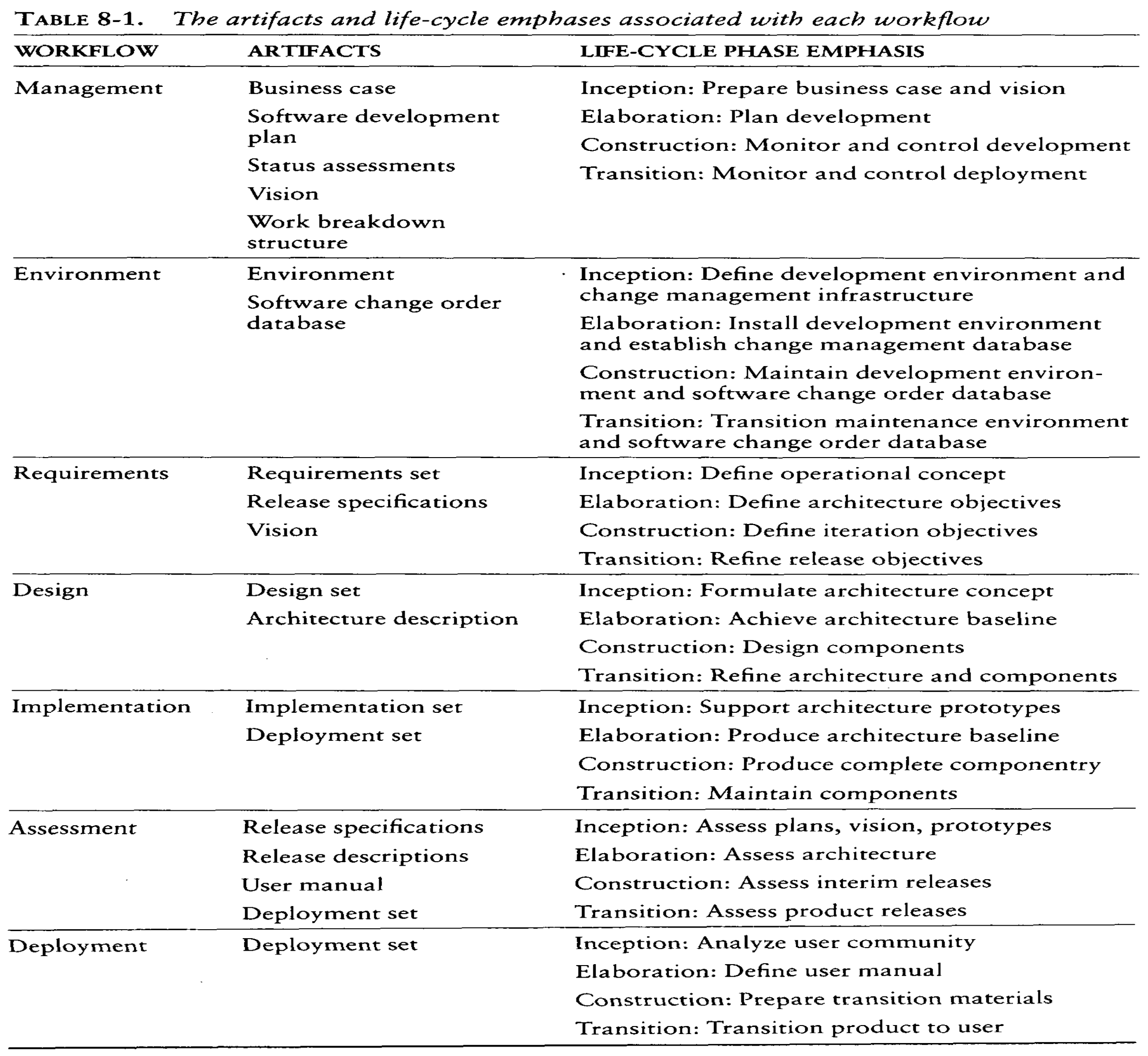


TABLE 8-1. *The artifacts and life-cycle emphases associated with each workflow*

WORKFLOW	ARTIFACTS	LIFE-CYCLE PHASE EMPHASIS
Management	Business case Software development plan Status assessments Vision Work breakdown structure	Inception: Prepare business case and vision Elaboration: Plan development Construction: Monitor and control development Transition: Monitor and control deployment
Environment	Environment Software change order database	Inception: Define development environment and change management infrastructure Elaboration: Install development environment and establish change management database Construction: Maintain development environment and software change order database Transition: Transition maintenance environment and software change order database
Requirements	Requirements set Release specifications Vision	Inception: Define operational concept Elaboration: Define architecture objectives Construction: Define iteration objectives Transition: Refine release objectives
Design	Design set Architecture description	Inception: Formulate architecture concept Elaboration: Achieve architecture baseline Construction: Design components Transition: Refine architecture and components
Implementation	Implementation set Deployment set	Inception: Support architecture prototypes Elaboration: Produce architecture baseline Construction: Produce complete componentry Transition: Maintain components
Assessment	Release specifications Release descriptions User manual Deployment set	Inception: Assess plans, vision, prototypes Elaboration: Assess architecture Construction: Assess interim releases Transition: Assess product releases
Deployment	Deployment set	Inception: Analyze user community Elaboration: Define user manual Construction: Prepare transition materials Transition: Transition product to user

Iteration Workflow

- An iteration consists of a loosely sequential set of activities in various proportions, depending on where the iteration is located in the development cycle.
- Each iteration is defined in terms of a set of allocated usage scenarios.
- The components needed to implement all selected scenarios are developed and integrated with the results of previous iterations.

Iteration Workflow

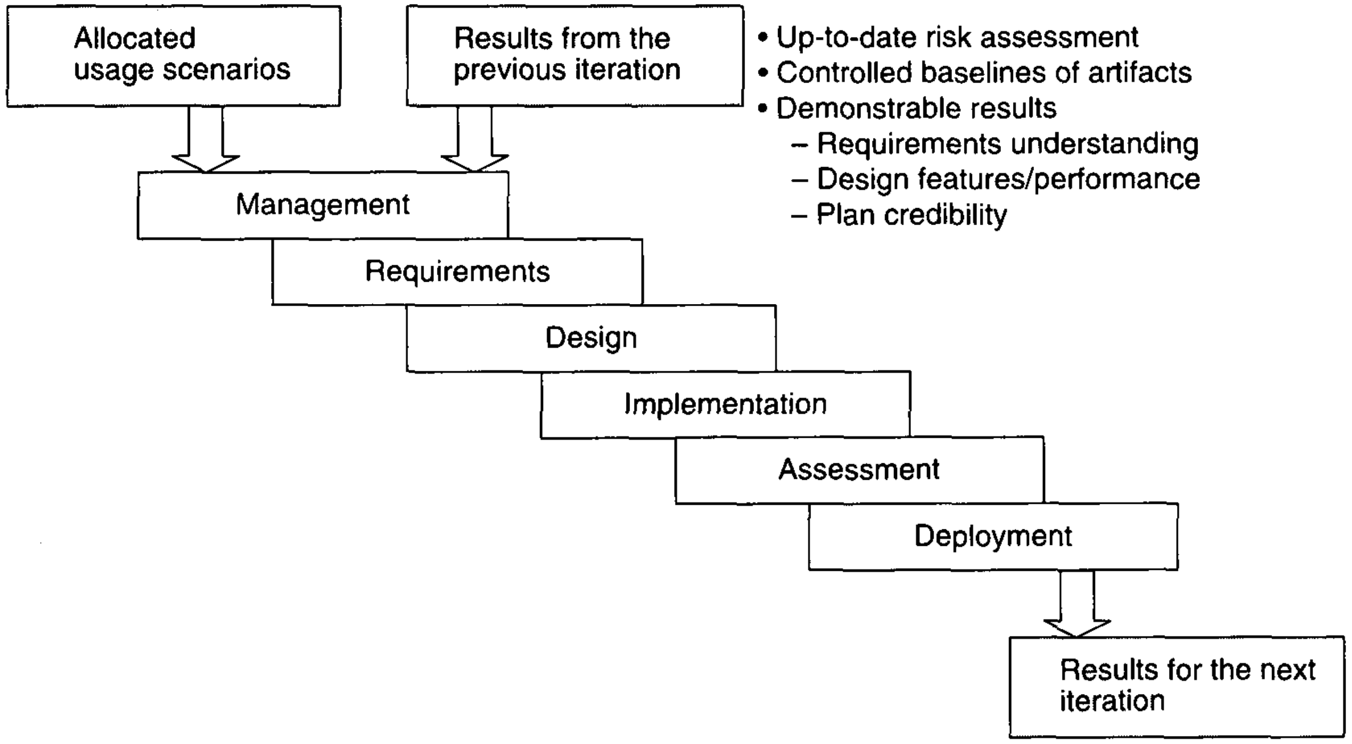

Allocated usage scenarios
Results from the previous iteration
• Up-to-date risk assessment
• Controlled baselines of artifacts
• Demonstrable results
– Requirements understanding
– Design features/performance
– Plan credibility
Management
Requirements
Design
Implementation
Assessment
Deployment
Results for the next iteration

FIGURE 8-2. *The workflow of an iteration*

Iteration emphasis across the life cycle

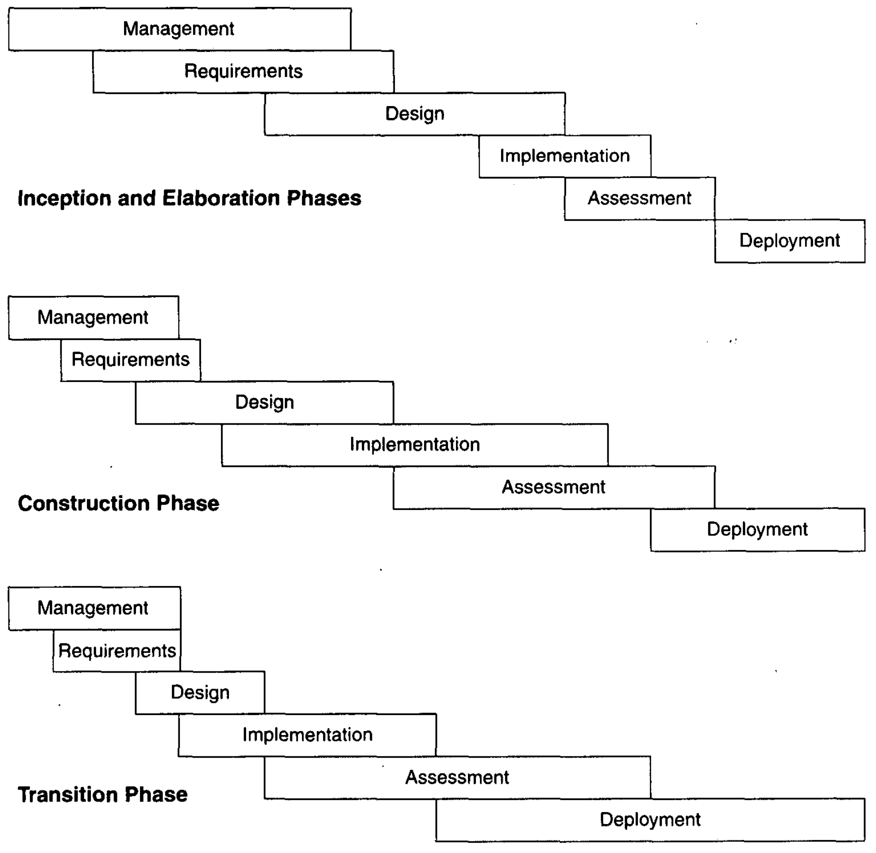


FIGURE 8-3. *Iteration emphasis across the life cycle*

- Management:
  - iteration planning to determine the content of the release and develop the detailed plan for the iteration; assignment of work packages, or tasks, to the development team
- Environment:
  - evolving the software change order database to reflect all new baselines and changes to existing baselines for all product, test, and environment components

- Requirements:
  - analyzing the baseline plan, the baseline architecture, and the baseline requirements set artifacts to fully elaborate the use cases to be demonstrated at the end of this iteration and their evaluation criteria; updating any requirements set artifacts to reflect changes necessitated by results of this iteration’s engineering activities
- Design:
  - evolving the baseline architecture and the baseline design set artifacts to elaborate fully the design model and test model components necessary to demonstrate against the evaluation criteria allocated to this iteration; updating design set artifacts to reflect changes necessitated by the results of this iteration’s engineering activities

- Implementation:
  - developing or acquiring any new components, and enhancing or modifying any existing components, to demonstrate the evaluation criteria allocated to this iteration; integrating and testing all new and modified components with existing baselines (previous versions)
- Assessment:
  - evaluating the results of the iteration, including compliance with the allocated evaluation criteria and the quality of the current baselines; identifying any rework required and determining whether it should be performed before deployment of this release or allocated to the next release; assessing results to improve the basis of the subsequent iteration’s plan

- Deployment:
  - transitioning the release either to an external organization (such as a user, independent verification and validation contractor, or regulatory agency) or to internal closure by conducting a post-mortem so that lessons learned can be captured and reflected in the next iteration

Status Monitoring – Software Process Checkpoints and Milestone

Checkpoints of the process

- It is always important to have visible milestones in the life cycle where various stakeholders meet, face to face, to discuss progress and plans.
- The purpose of these events is not only to demonstrate how well a project is performing but also to achieve the following:
  - Synchronize stakeholder expectations and achieve concurrence on three evolving perspectives: the requirements, the design, and the plan
  - Synchronize related artifacts into a consistent and balanced state
  - Identify the important risks, issues, and out-of-tolerance conditions
  - Perform a global assessment for the whole life cycle, not just the current situation of an individual perspective or intermediate product

- Three types of joint management reviews are conducted throughout the process:
- ***Major milestones:-***
  - *These system wide events are held at the end of each* development phase. They provide visibility to system wide issues, synchronize the management and engineering perspectives, and verify that the aims of the phase have been achieved.
- ***Minor milestones:-***
  - *These iteration-focused events are conducted to review* the content of an iteration in detail and to authorize continued work.
- ***Status assessments:-***
  - *These periodic events provide management with frequent* and regular insight into the progress being made.

TABLE 9-1. *The general status of plans, requirements, and products across the major milestones*

MILESTONES	PLANS	UNDERSTANDING OF PROBLEM SPACE (REQUIREMENTS)	SOLUTION SPACE PROGRESS (SOFTWARE PRODUCT)
Life-cycle objectives milestone	Definition of stakeholder responsibilities Low-fidelity life-cycle plan High-fidelity elaboration phase plan	Baseline vision, including growth vectors, quality attributes, and priorities Use case model	Demonstration of at least one feasible architecture Make/buy/reuse trade-offs Initial design model
Life-cycle architecture milestone	High-fidelity construction phase plan (bill of materials, labor allocation) Low-fidelity transition phase plan	Stable vision and use case model Evaluation criteria for construction releases, initial operational capability Draft user manual	Stable design set Make/buy/reuse decisions Critical component prototypes
Initial operational capability milestone	High-fidelity transition phase plan	Acceptance criteria for product release Releasable user manual	Stable implementation set Critical features and core capabilities Objective insight into product qualities
Product release milestone	Next-generation product plan	Final user manual	Stable deployment set Full features Compliant quality

Assignment Questions of Unit 2

- Define Software Process Life Cycle Phase.
- Explain about the various phases of Software Development Life Cycle.
- Explain about the Software Architecture on the basis of Management Perspective and Technical Perspective .
- Define Software Process Workflows.
- What are the four basic key principles of software process workflows.
- Discuss about the Software Process Checkpoints and milestone.

Any Queries?