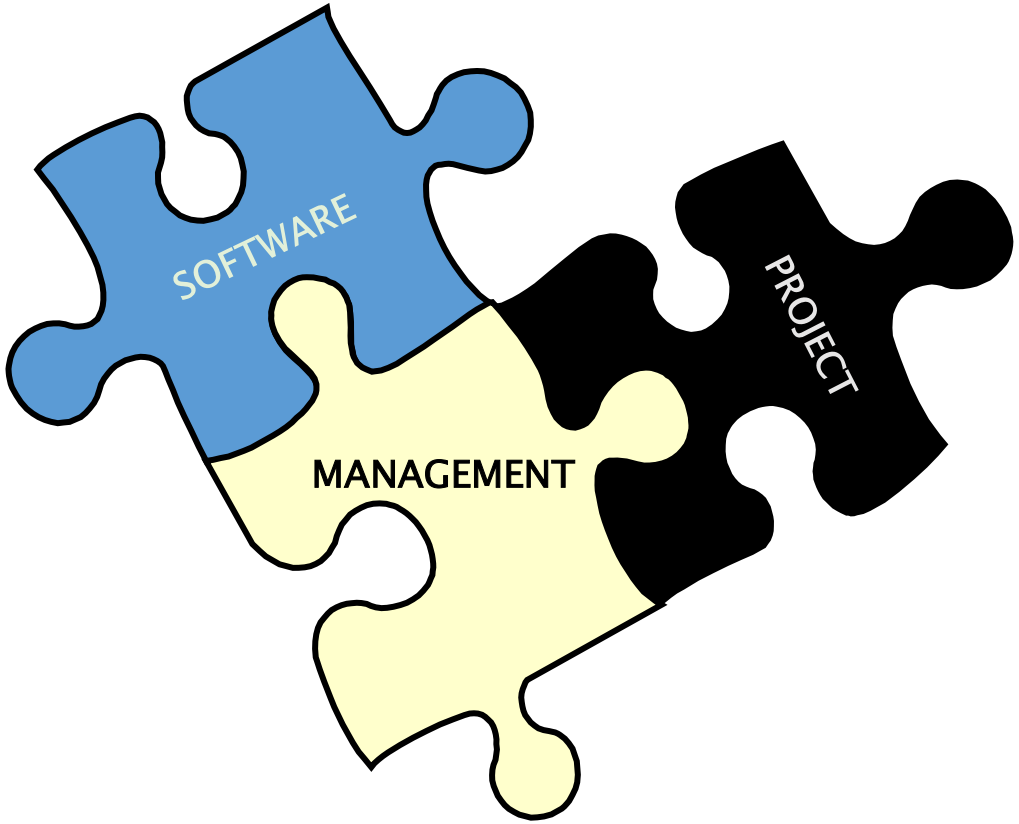
Software Project Management



Unit 2. Software Process Primitives and Process Management Frameworks

Prepared By:

BIJAY MISHRA

(बिजय मिश्र)

biizay@gmail.com

@jijibisha

UNIT 2. Software Process Primitives and Process Management Frameworks - 14 Hrs.

2.1 Software Process Life-Cycle Phases

2.2 Various Elements of the Software Process (Management, Engineering and Pragmatics)

2.3 Technical and Management Perspective of Software Architecture

2.4 Software Process Workflow and Iteration Workflow

2.5 Status Monitoring - Software Process Checkpoints and Milestones

2.1 Software Process Life-Cycle Phases

Life-Cycle Phases

Engineering and Production Stages

- 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.
- 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.
- Earlier phases focus on achieving functionality. Later phases revolve around achieving a product that can be shipped to a customer, with explicit attention to robustness, performance, and finish.
- 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

Life-Cycle Phases

Engineering and Production Stages

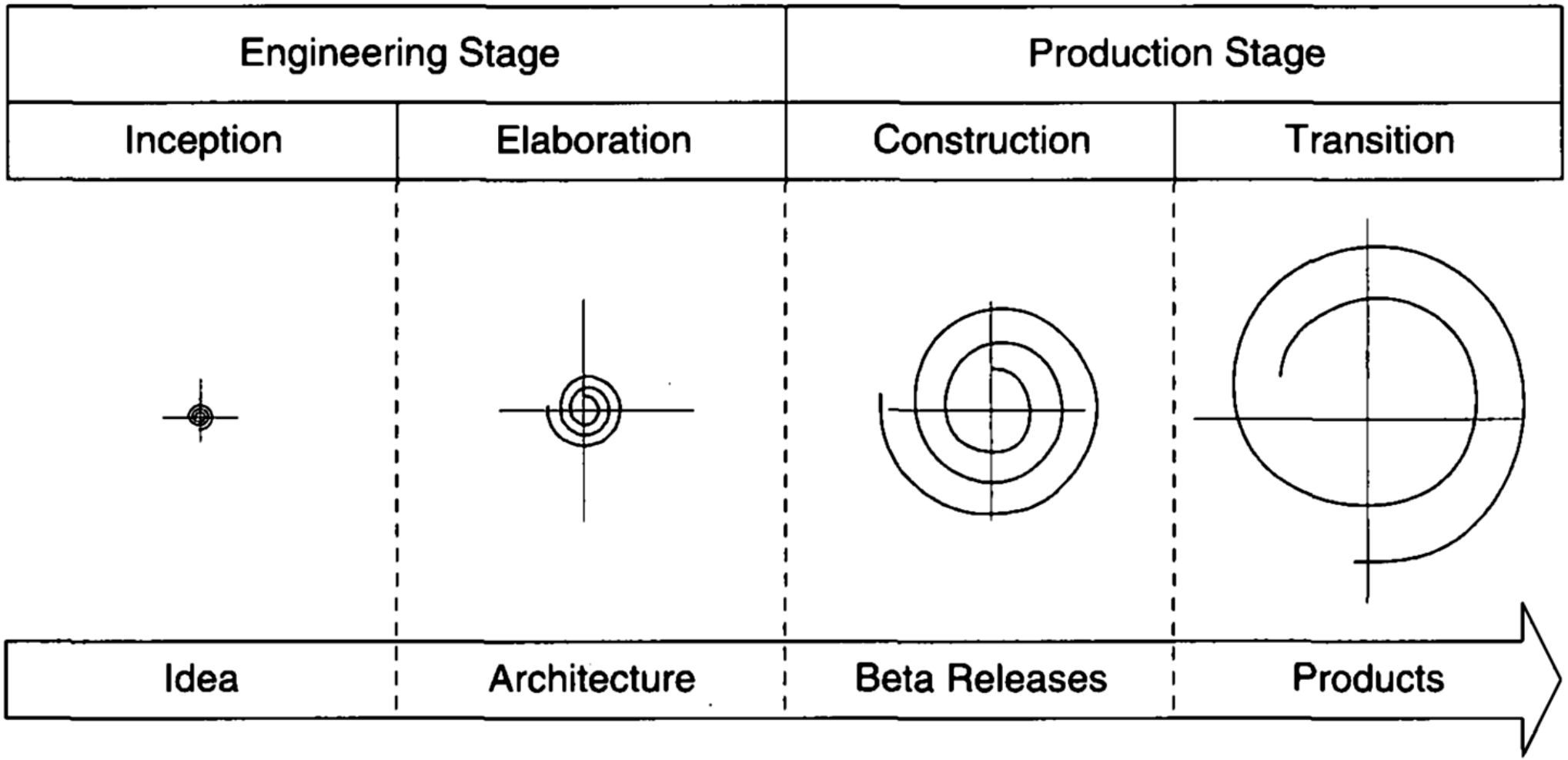
Two stages of the life-cycle :

1. *The engineering stage* – driven by smaller teams doing design and synthesis activities
2. *The production stage* – driven by larger teams doing construction, test, and deployment activities

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 economics of scale
Management	Planning	Operations

Life-Cycle Phases

Engineering and Production Stages



Life-Cycle Phases

Inception Phase

OVERRIDING GOAL – to achieve concurrence among stakeholders on the life-cycle objectives

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)

Life-Cycle Phases

Inception Phase

ESSENTIAL ACTIVITIES:

- ❑ *Formulating the scope of the project* (capturing the requirements and operational concept in an information repository)
- ❑ *Synthesizing the architecture* (design trade-offs, problem space ambiguities, and available solution-space assets are evaluated)
- ❑ *Planning and preparing a business case* (alternatives for risk management, iteration planes, and cost/schedule/profitability trade-offs are evaluated)

Life-Cycle Phases

Inception Phase

PRIMARY 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

Life-Cycle Phases

Elaboration Phase

During the elaboration phase, an executable architecture prototype is built

PRIMARY OBJECTIVES

- ❑ Baselining the architecture as rapidly as practical (establishing a configuration-managed snapshot in which all changes are rationalized, tracked, and maintained)
- ❑ Baselining the vision
- ❑ Baselining 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

Life-Cycle Phases

Elaboration Phase

ESSENTIAL ACTIVITIES :

- ❑ Elaborating the vision (establishing a high-fidelity understanding of the critical use cases that drive architectural or planning decisions)
- ❑ Elaborating the process and infrastructure (establishing the construction process, the tools and process automation support)
- ❑ Elaborating the architecture and selecting components (lessons learned from these activities may result in redesign of the architecture)

Life-Cycle Phases

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?

Life-Cycle Phases

Construction Phase

During the construction phase :

- ❑ All remaining components and application features are integrated into the application
- ❑ All features are thoroughly tested

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

Life-Cycle Phases

Construction Phase

ESSENTIAL ACTIVITIES :

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

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?

Life-Cycle Phases

Transition Phase

- The transition phase is entered when baseline is mature enough to be deployed in the end-user domain
- This phase could include beta testing, conversion of operational databases, and training of users and maintainers

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

Life-Cycle Phases

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

EVALUATION CRITERIA

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

2.2 Various Elements of the Software Process (Management, Engineering and Pragmatics)

Artifacts of the Process

- To make the development of a complete software system manageable, distinct collections of information are organized into artifact sets.
- Artifact represents cohesive information that typically is developed and reviewed as a single entity.
- Life-cycle software artifacts are organized into five distinct sets that are roughly partitioned by the underlying language of the set: **management** (ad hoc textual formats), **requirements** (organized text and models of the problem space), **design** (models of the solution space), **implementation** (human-readable programming language and associated source files), and **deployment** (machine-process able languages and associated files).

Artifacts of the Process

THE ARTIFACT SETS

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 executable	1.Integrated product executable baselines 2.Associated run-time files 3.User manual

Management Set

Planning Artifacts

1.Work breakdown structure
2.Bussines case
3.Release specifications
4.Software development plan

Operational Artifacts

5.Release descriptions
6.Status assessments
7.Software change order database
8.Deployment documents
9.Environment

Artifacts of the Process

Management Set

The Management Set :

The management set captures the artifacts associated with process planning and execution. These artifacts use ad hoc notations, including text, graphics, or whatever representation is required to capture the "contracts" among project personnel (project management, architects, developers, testers, marketers, administrators), among stakeholders (funding authority, user, software project manager, organization manager, regulatory agency), and between project personnel and stakeholders. Specific artifacts included in this set are the work breakdown structure (activity breakdown and financial tracking mechanism), the business case (cost, schedule, profit expectations), the release specifications (scope, plan, objectives for release baselines), the software development plan (project process instance), the release descriptions (results of release baselines), the status assessments (periodic snapshots of project progress), the software change orders (descriptions of discrete baseline changes), the deployment documents (cutover plan, training course, sales rollout kit), and the environment (hardware and software tools, process automation, & documentation).

Artifacts of the Process

Management Set

The Management Set:

Management set artifacts are evaluated, assessed, and measured through a combination of the following:

- Relevant stakeholder review
- Analysis of changes between the current version of the artifact and previous versions
- Major milestone demonstrations of the balance among all artifacts and, in particular, the accuracy of the business case and vision artifacts

Artifacts of the Process

Engineering Set

THE ENGINEERING SET :

The engineering sets consist of the requirements set, the design set, the implementation set, and the deployment set.

REQUIREMENTS SET

Requirements artifacts are evaluated, assessed, and measured through a combination of the following:

- ❑ Analysis of consistency with the release specifications of the management set
- ❑ Analysis of consistency between the vision and the requirements models
- ❑ Mapping against the design, implementation, and deployment sets to evaluate the consistency and completeness and the semantic balance between information in the different sets
- ❑ Analysis of changes between the current version of requirements artifacts and previous versions (scrap, rework, and defect elimination trends)
- ❑ Subjective review of other dimensions of quality

Artifacts of the Process

Engineering Set

DESIGN SET

UML notation is used to engineer the design models for the solution. The design set contains varying levels of abstraction that represent the components of the solution space (their identities, attributes, static relationships, dynamic interactions). The design set is evaluated, assessed, and measured through a combination of the following:

- ❑ Analysis of the internal consistency and quality of the design model
- ❑ Analysis of consistency with the requirements models
- ❑ Translation into implementation and deployment sets and notations (for example, traceability, source code generation, compilation, linking) to evaluate the consistency and completeness and the semantic balance between information in the sets
- ❑ Analysis of changes between the current version of the design model and previous versions (scrap, rework, and defect elimination trends)
- ❑ Subjective review of other dimensions of quality

Artifacts of the Process

Engineering Set

IMPLEMENTATION SET

- The implementation set includes source code (programming language notations) that represents the tangible implementations of components (their form, interface, and dependency relationships).
- Implementation sets are human-readable formats that are evaluated, assessed, and measured through a combination of the following:
  - Analysis of consistency with the design models
  - Translation into deployment set notations (for example, compilation and linking) to evaluate the consistency and completeness among artifact sets
  - Assessment of component source or executable files against relevant evaluation criteria through inspection, analysis, demonstration, or testing
  - Execution of stand-alone component test cases that automatically compare expected results with actual results
  - Analysis of changes between the current version of the implementation set and previous versions (scrap, rework, and defect elimination trends)
  - Subjective review of other dimensions of quality

Artifacts of the Process

Engineering Set

DEPLOYMENT SET

- The deployment set includes user deliverables and machine language notations, executable software, and the build scripts, installation scripts, and executable target specific data necessary to use the product in its target environment.
- Deployment sets are evaluated, assessed, and measured through a combination of the following:
  - Testing against the usage scenarios and quality attributes defined in the requirements set to evaluate the consistency and completeness and the~ semantic balance between information in the two sets
  - Testing the partitioning, replication, and allocation strategies in mapping components of the implementation set to physical resources of the deployment system (platform type, number, network topology)
  - Testing against the defined usage scenarios in the user manual such as installation, user-oriented dynamic reconfiguration, mainstream usage, and anomaly management
  - Analysis of changes between the current version of the deployment set and previous versions (defect elimination trends, performance changes)
  - Subjective review of other dimensions of quality

Artifacts of the Process

Management and Engineering Set

Most of today's software development tools map closely to one of the five artifact sets.

- **Management**: scheduling, workflow, defect tracking, change management, documentation, spreadsheet, resource management, and presentation tools
- **Requirements**: requirements management tools
- **Design**: visual modeling tools
- **Implementation**: compiler/debugger tools, code analysis tools, test coverage analysis tools, and test management tools
- **Deployment**: test coverage and test automation tools, network management tools, commercial components (operating systems, GUIs, RDBMS, networks, middleware), and installation tools.

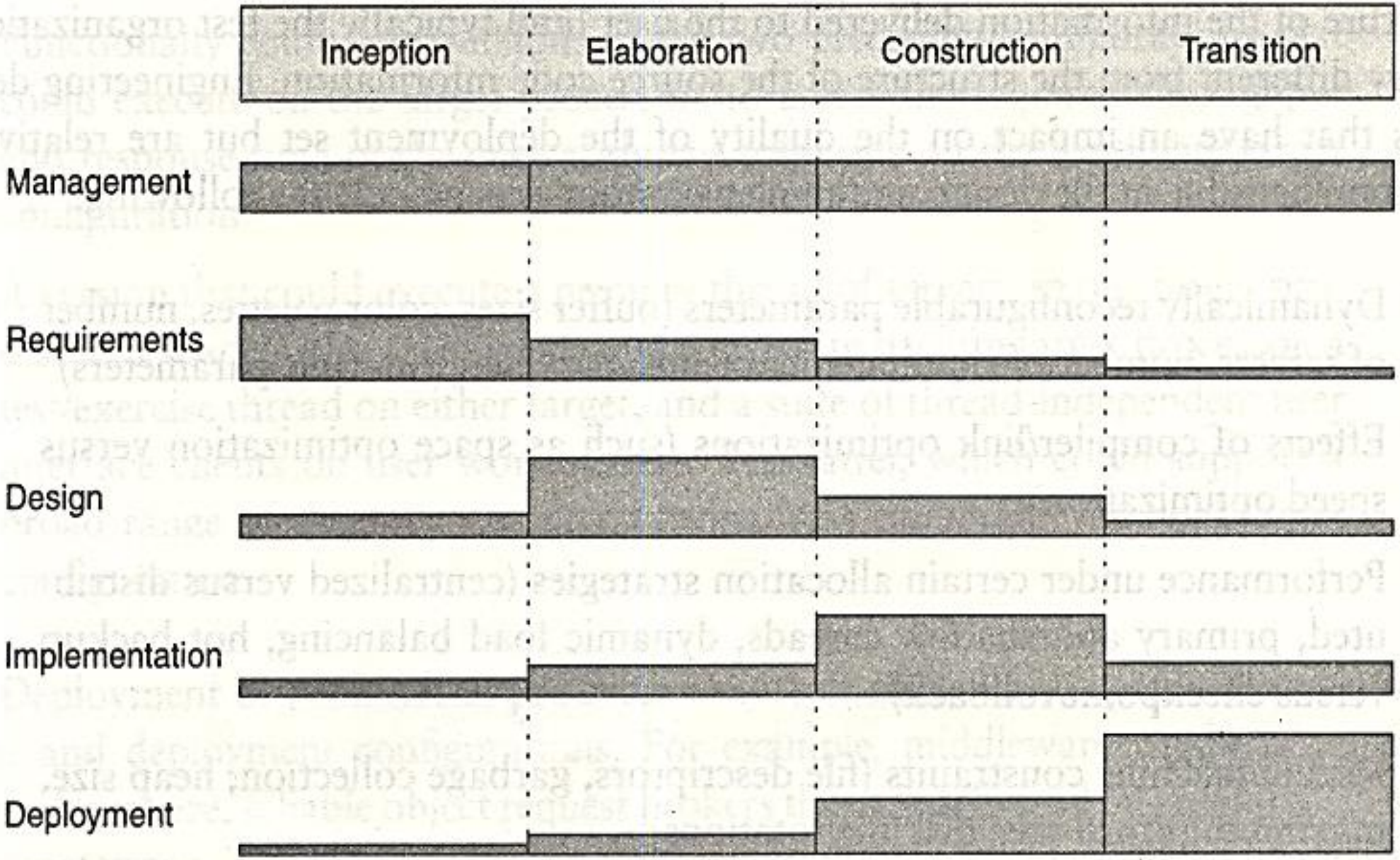


FIGURE 6-2. *Life-cycle focus on artifact sets*

ARTIFACT EVOLUTION OVER THE LIFE CYCLE

Each state of development represents a certain amount of precision in the final system description. Early in the life cycle, precision is low and the representation is generally high. Eventually, the precision of representation is high and everything is specified in full detail. Each phase of development focuses on a particular artifact set. At the end of each phase, the overall system state will have progressed on all sets, as illustrated in Figure below:

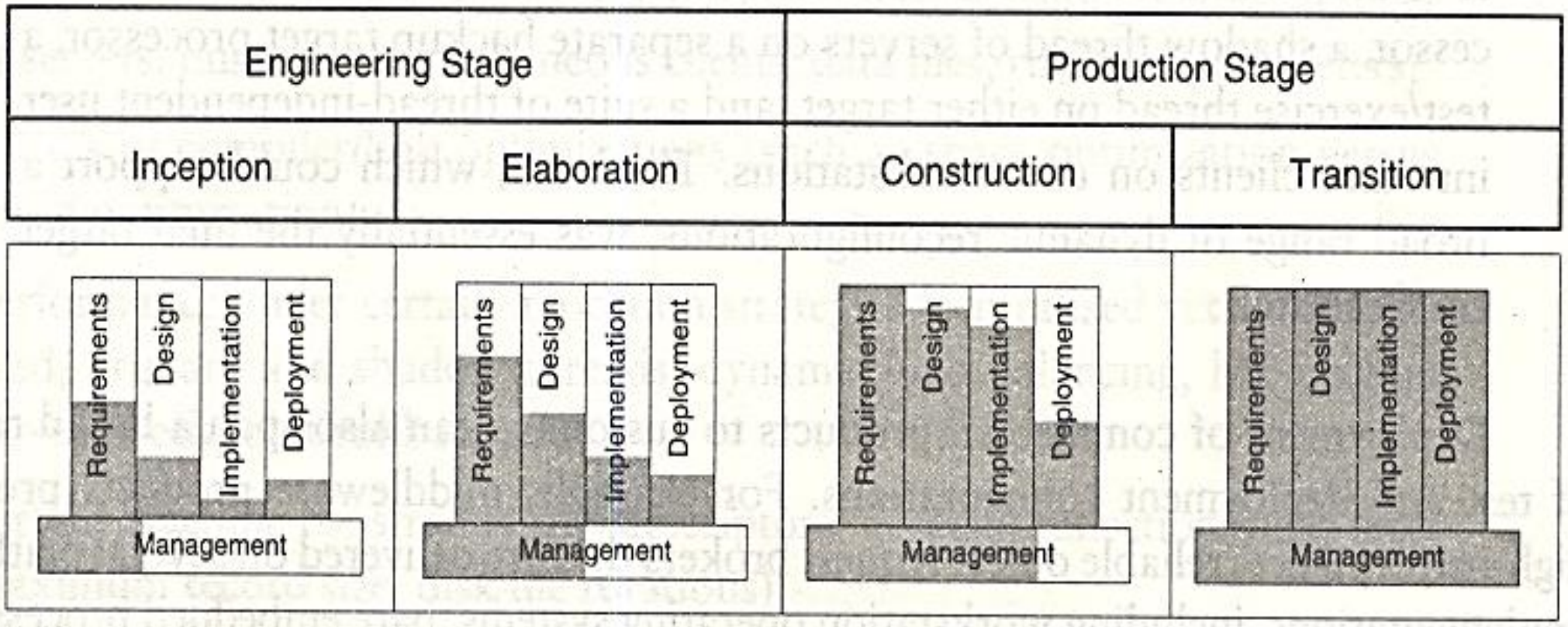


FIGURE 6-3. *Life-cycle evolution of the artifact sets*

TEST ARTIFACTS

- The test artifacts must be developed concurrently with the product from inception through deployment.
- Testing is a full-life-cycle activity, not a late life-cycle activity.
- The test artifacts are communicated, engineered, and developed within the same artifact sets as the developed product.
- The test artifacts are implemented in programmable and repeatable formats (as software programs).
- The test artifacts are documented in the same way that the product is documented.
- Developers of the test artifacts use the same tools, techniques, and training as the software engineers developing the product.

TYPES OF TEST ARTIFACTS

Management set. The release specifications and release descriptions capture the objectives, evaluation criteria, and results of an intermediate milestone. These artifacts are the test plans and test results negotiated among internal project teams. The software change orders capture test results (defects, testability changes, requirements ambiguities, enhancements) and the closure criteria associated with making a discrete change to a baseline.

Requirements set. The system-level use cases capture the operational concept for the system and the acceptance test case descriptions, including the expected behavior of the system and its quality attributes. The entire requirement set is a test artifact because it is the basis of all assessment activities across the life cycle.

TYPES OF TEST ARTIFACTS

Design set. A test model for non-deliverable components needed to test the product baselines is captured in the design set. These components include such design set artifacts as a seismic event simulation for creating realistic sensor data; a "virtual operator" that can support unattended, after-hours test cases; specific instrumentation suites for early demonstration of resource usage; transaction rates or response times; and use case test drivers and component stand-alone test drivers.

Implementation set. Self-documenting source code representations for test components and test drivers provide the equivalent of test procedures and test scripts. These source files may also include human-readable data files representing certain statically defined data sets that are explicit test source files. Output files from test drivers provide the equivalent of test reports.

Deployment set. Executable versions of test components, test drivers, and data files are provided.

Artifacts of the Process

Management Artifacts

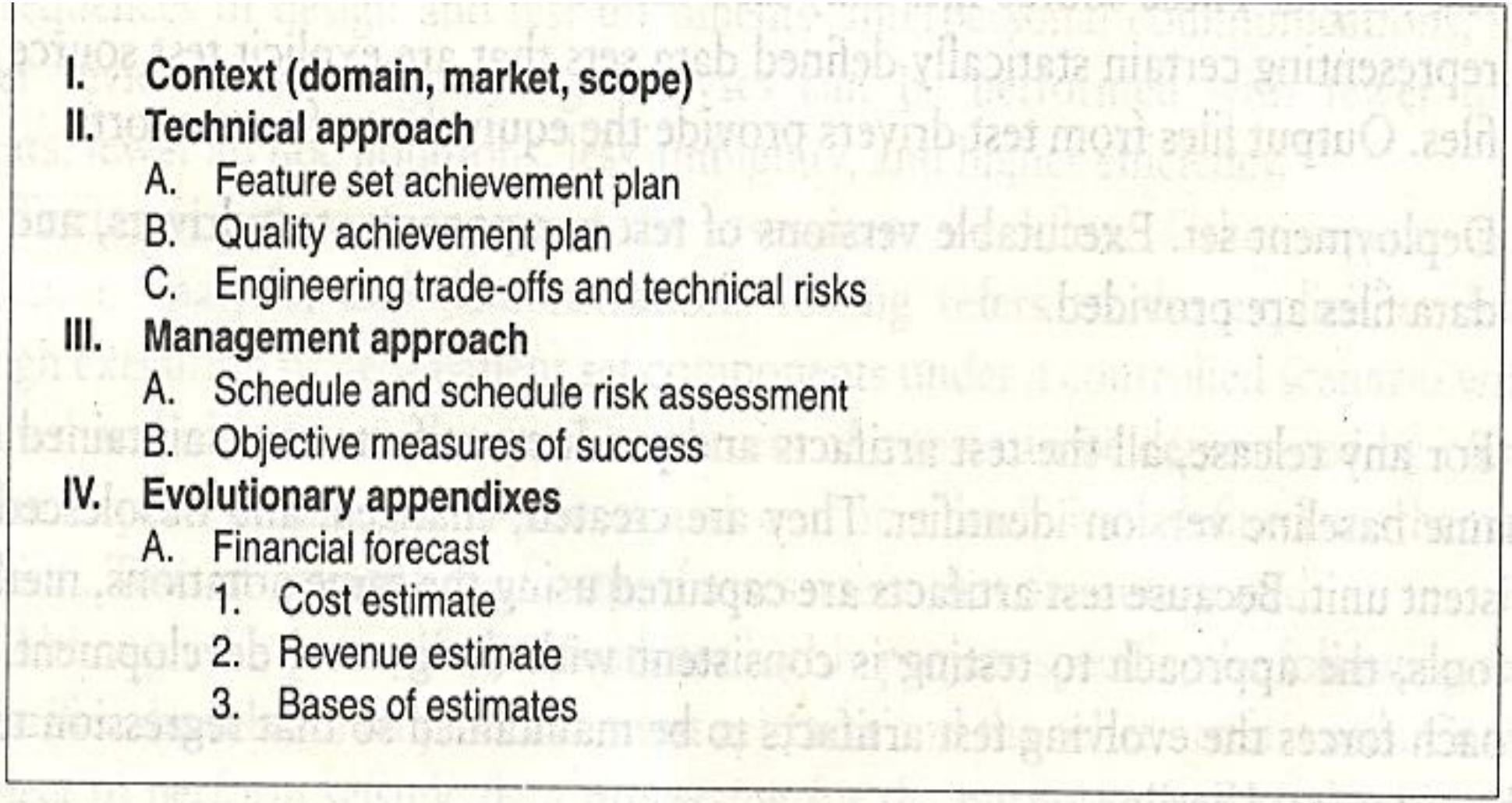
The management set includes several artifacts that capture intermediate results and ancillary information necessary to document the product/process legacy, maintain the product, improve the product, and improve the process.

Business Case

The business case artifact 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, and backup data necessary to demonstrate the risks and realism of the plans. The main purpose is to transform the vision into economic terms so that an organization can make an accurate ROI assessment. The financial forecasts are evolutionary, updated with more accurate forecasts as the life cycle progresses. Figure 6-4 provides a default outline for a business case.

Artifacts of the Process

Management Artifacts



I. **Context (domain, market, scope)**
II. **Technical approach**
    A. Feature set achievement plan
    B. Quality achievement plan
    C. Engineering trade-offs and technical risks
III. **Management approach**
    A. Schedule and schedule risk assessment
    B. Objective measures of success
IV. **Evolutionary appendixes**
    A. Financial forecast
        1. Cost estimate
        2. Revenue estimate
        3. Bases of estimates

FIGURE 6-4. *Typical business case outline*

Artifacts of the Process

Management Artifacts

SOFTWARE DEVELOPMENT PLAN

The software development plan (SDP) elaborates the process framework into a fully detailed plan. Two indications of a useful SDP are periodic updating (it is not stagnant shelfware) and understanding and acceptance by managers and practitioners alike. Figure 6-5 provides a default outline for a software development plan.

WORK BREAKDOWN STRUCTURE

Work breakdown structure (WBS) is the vehicle for budgeting and collecting costs. To monitor and control a project's financial performance, the software project man¬1ger must have insight into project costs and how they are expended. The structure of cost accountability is a serious project planning constraint.

SOFTWARE CHANGE ORDER DATABASE

Managing change is one of the fundamental primitives of an iterative development process. With greater change freedom, a project can iterate more productively. This flexibility increases the content, quality, and number of iterations that a project can achieve within a given schedule. Change freedom has been achieved in practice through automation, and today's iterative development environments carry the burden of change management. Organizational processes that depend on manual change management techniques have encountered major inefficiencies.

I. **Context (scope, objectives)**
II. **Software development process**
   A. Project primitives
      1. Life-cycle phases
      2. Artifacts
      3. Workflows
      4. Checkpoints
   B. Major milestone scope and content
   C. Process improvement procedures
III. **Software engineering environment**
   A. Process automation (hardware and software resource configuration)
   B. Resource allocation procedures (sharing across organizations, security access)
IV. **Software change management**
   A. Configuration control board plan and procedures
   B. Software change order definitions and procedures
   C. Configuration baseline definitions and procedures
V. **Software assessment**
   A. Metrics collection and reporting procedures
   B. Risk management procedures (risk identification, tracking, and resolution)
   C. Status assessment plan
   D. Acceptance test plan
VI. **Standards and procedures**
   A. Standards and procedures for technical artifacts
VII. **Evolutionary appendixes**
   A. Minor milestone scope and content
   B. Human resources (organization, staffing plan, training plan)

FIGURE 6-5. *Typical software development plan outline*

Artifacts of the Process

Management Artifacts

RELEASE SPECIFICATIONS

The scope, plan, and objective evaluation criteria for each baseline release are derived from the vision statement as well as many other sources (make/buy analyses, risk management concerns, architectural considerations, shots in the dark, implementation constraints, quality thresholds). These artifacts are intended to evolve along with the process, achieving greater fidelity as the life cycle progresses and requirements understanding matures. Figure 6-6 provides a default outline for a release specification

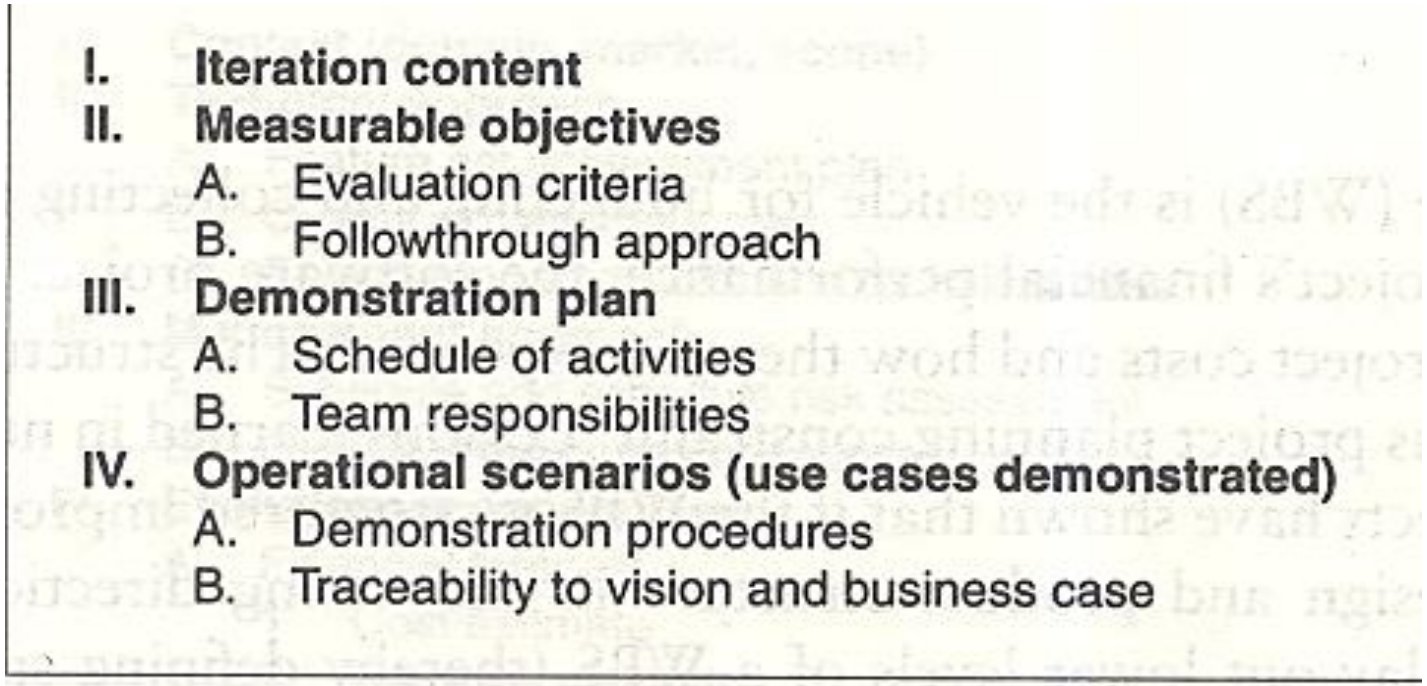


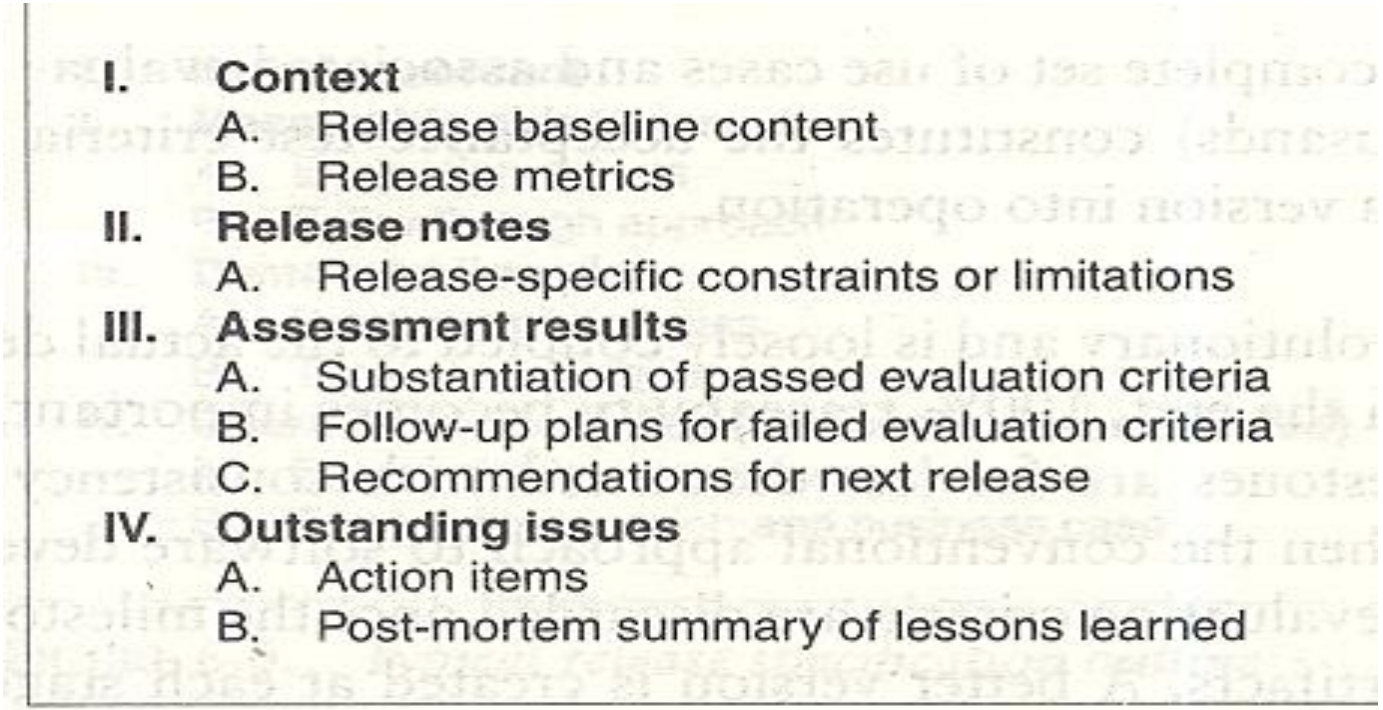
FIGURE 6-6. *Typical release specification outline*

Artifacts of the Process

Management Artifacts

RELEASE DESCRIPTIONS

Release description documents describe the results of each release, including performance against each of the evaluation criteria in the corresponding release specification. Release baselines should be accompanied by a release description document that describes the evaluation criteria for that configuration baseline and provides substantiation (through demonstration, testing, inspection, or analysis) that each criterion has been addressed in an acceptable manner. Figure 6-7 provides a default outline for a release description.



I. **Context**
   - A. Release baseline content
   - B. Release metrics

II. **Release notes**
   - A. Release-specific constraints or limitations

III. **Assessment results**
   - A. Substantiation of passed evaluation criteria
   - B. Follow-up plans for failed evaluation criteria
   - C. Recommendations for next release

IV. **Outstanding issues**
   - A. Action items
   - B. Post-mortem summary of lessons learned

FIGURE 6-7. *Typical release description outline*

Artifacts of the Process

Management Artifacts

STATUS ASSESSMENTS

Status assessments provide periodic snapshots of project health and status, including the software project manager's risk assessment, quality indicators, and management indicators. Typical status assessments should include a review of resources, personnel staffing, financial data (cost and revenue), top 10 risks, technical progress (metrics snapshots), major milestone plans and results, total project or product scope & action items

ENVIRONMENT

An important emphasis of a modern approach is to define the development and maintenance environment as a first-class artifact of the process. A robust, integrated development environment must support automation of the development process. This environment should include requirements management, visual modeling, document automation, host and target programming tools, automated regression testing, and continuous and integrated change management, and feature and defect tracking.

Artifacts of the Process

Management Artifacts

DEPLOYMENT

A deployment document can take many forms. Depending on the project, it could include several document subsets for transitioning the product into operational status. In big contractual efforts in which the system is delivered to a separate maintenance organization, deployment artifacts may include computer system operations manuals, software installation manuals, plans and procedures for cutover (from a legacy system), site surveys, and so forth. For commercial software products, deployment artifacts may include marketing plans, sales rollout kits, and training courses.

MANAGEMENT ARTIFACT SEQUENCES

In each phase of the life cycle, new artifacts are produced and previously developed artifacts are updated to incorporate lessons learned and to capture further depth and breadth of the solution. Figure 6-8 identifies a typical sequence of artifacts across the life-cycle phases.

Artifacts of the Process

Engineering Artifacts

Most of the engineering artifacts are captured in rigorous engineering notations such as UML, programming languages, or executable machine codes. Three engineering artifacts are explicitly intended for more general review, and they deserve further elaboration.

VISION DOCUMENT

The vision document provides a complete vision for the software system under development and. supports the contract between the funding authority and the development organization. A project vision is meant to be changeable as understanding evolves of the requirements, architecture, plans, and technology. A good vision document should change slowly. Figure 6-9 provides a default outline for a vision document.

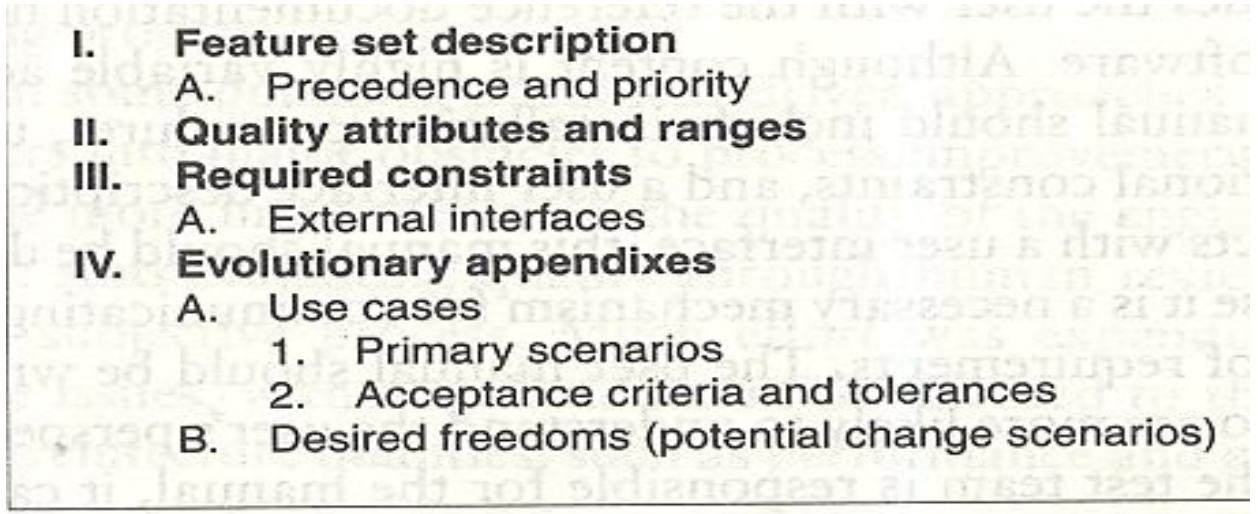


FIGURE 6-9. *Typical vision document outline*

Artifacts of the Process

Engineering Artifacts

ARCHITECTURE DESCRIPTION

The architecture description provides an organized view of the software architecture under development. It is extracted largely 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. The breadth of the architecture description will vary from project to project depending on many factors. Figure 6-10 provides a default outline for an architecture description.

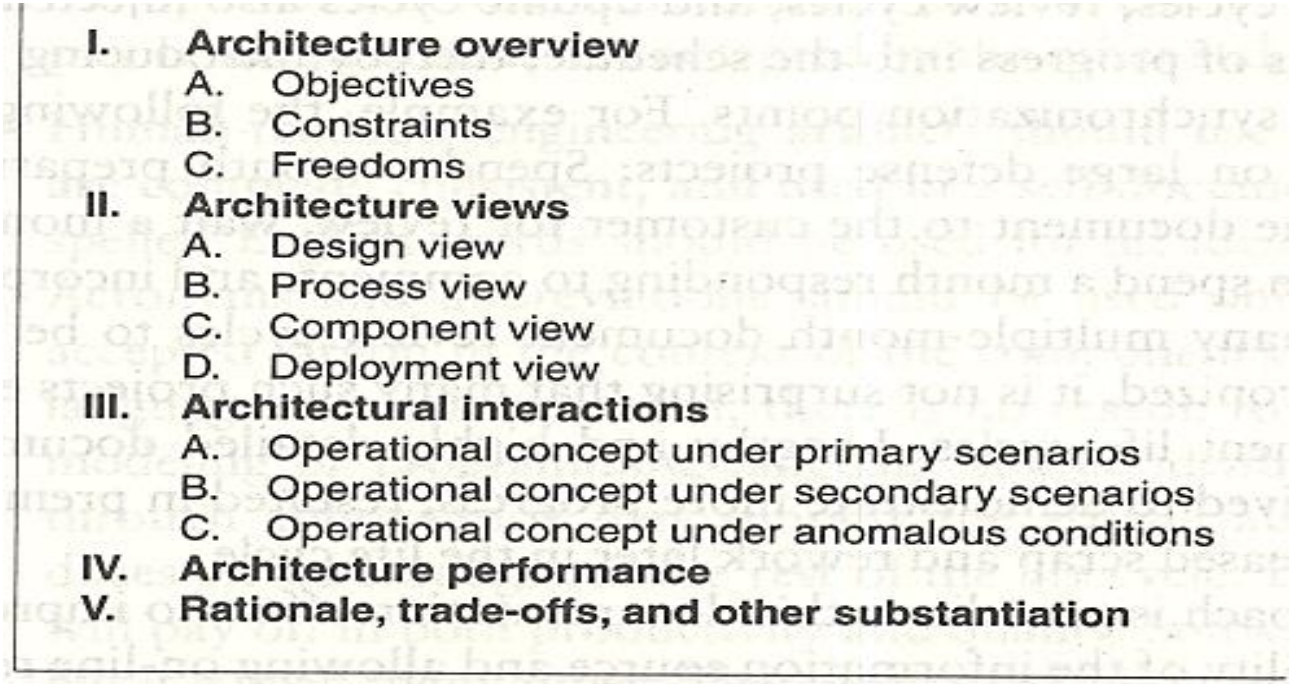


FIGURE 6-10. *Typical architecture description outline*

Artifacts of the Process

Engineering Artifacts

SOFTWARE USER MANUAL

The software user manual provides the user with the reference documentation necessary to support the delivered software. Although content is highly variable across application domains, the user manual should include installation procedures, usage procedures and guidance, operational constraints, and a user interface description, at a minimum. For software products with a user interface, this manual should be developed early in the life cycle because it is a necessary mechanism for communicating and stabilizing an important subset of requirements. The user manual should be written by members of the test team, who are more likely to understand the user's perspective than the development team.

Artifacts of the Process

Pragmatic Artifacts

People want to review information but don't understand the language of the artifact.

Many interested reviewers of a particular artifact will resist having to learn the engineering language in which the artifact is written. It is not uncommon to find people (such as veteran software managers, veteran quality assurance specialists, or an auditing authority from a regulatory agency) who react as follows: "I'm not going to learn UML, but I want to review the design of this software, so give me a separate description such as some flowcharts and text that I can understand."

Artifacts of the Process

Pragmatic Artifacts

People want to review the information but don't have access to the tools.

It is not very common for the development organization to be fully tooled; it is extremely rare that the/other stakeholders have any capability to review the engineering artifacts on-line. Consequently, organizations are forced to exchange paper documents. Standardized formats (such as UML, spreadsheets, Visual Basic, C++, and Ada 95), visualization tools, and the Web are rapidly making it economically feasible for all stakeholders to exchange information electronically.

Artifacts of the Process

Pragmatic Artifacts

Human-readable engineering artifacts should use rigorous notations that are complete, consistent, and used in a self-documenting manner.

Properly spelled English words should be used for all identifiers and descriptions. Acronyms and abbreviations should be used only where they are well-accepted jargon in the context of the component's usage. Readability should be emphasized and the use of proper English words should be required in all engineering artifacts. This practice enables understandable representations, browse able formats (paperless review), more-rigorous notations, and reduced error rates.

Artifacts of the Process

Pragmatic Artifacts

Useful documentation is self-defining: It is documentation that gets used.

Paper is tangible; electronic artifacts are too easy to change.
On-line and Web-based artifacts can be changed easily and are viewed with more skepticism because of their inherent volatility.

2.3 Technical and Management Perspective of Software Architecture

Model-Based Software Architectures

A Management Perspective

- The most critical technical product of a software project is its architecture: the infrastructure, control, and data interfaces that permit software components to cooperate as a system and software designers to cooperate efficiently as a team.
- When the communications media include multiple languages and intergroup literacy varies, the communications problem can become extremely complex and even unsolvable.
- If a software development team is to be successful, the inter project communications, as captured in the software architecture, must be both accurate and precise

Model-Based 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 opposed to design of a component.
- An ***architecture baseline*** (the tangible artifacts) is a slice 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.

Model-Based Software Architectures

A Management Perspective

The number of views and the level of detail in each view can vary widely. The importance of software architecture and its close linkage with modern software development processes can be summarized as follows:

- Achieving a stable software architecture represents a significant project milestone at which the critical make/buy decisions should have been resolved.
- Architecture representations provide a basis for balancing the trade-offs between the problem space (requirements and constraints) and the solution space (the operational product).
- The architecture and process encapsulate many of the important (high-payoff or high-risk) communications among individuals, teams, organizations, and stakeholders.
- 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; they require human innovation and cannot be automated.

Model-Based Software Architectures

A Technical Perspective

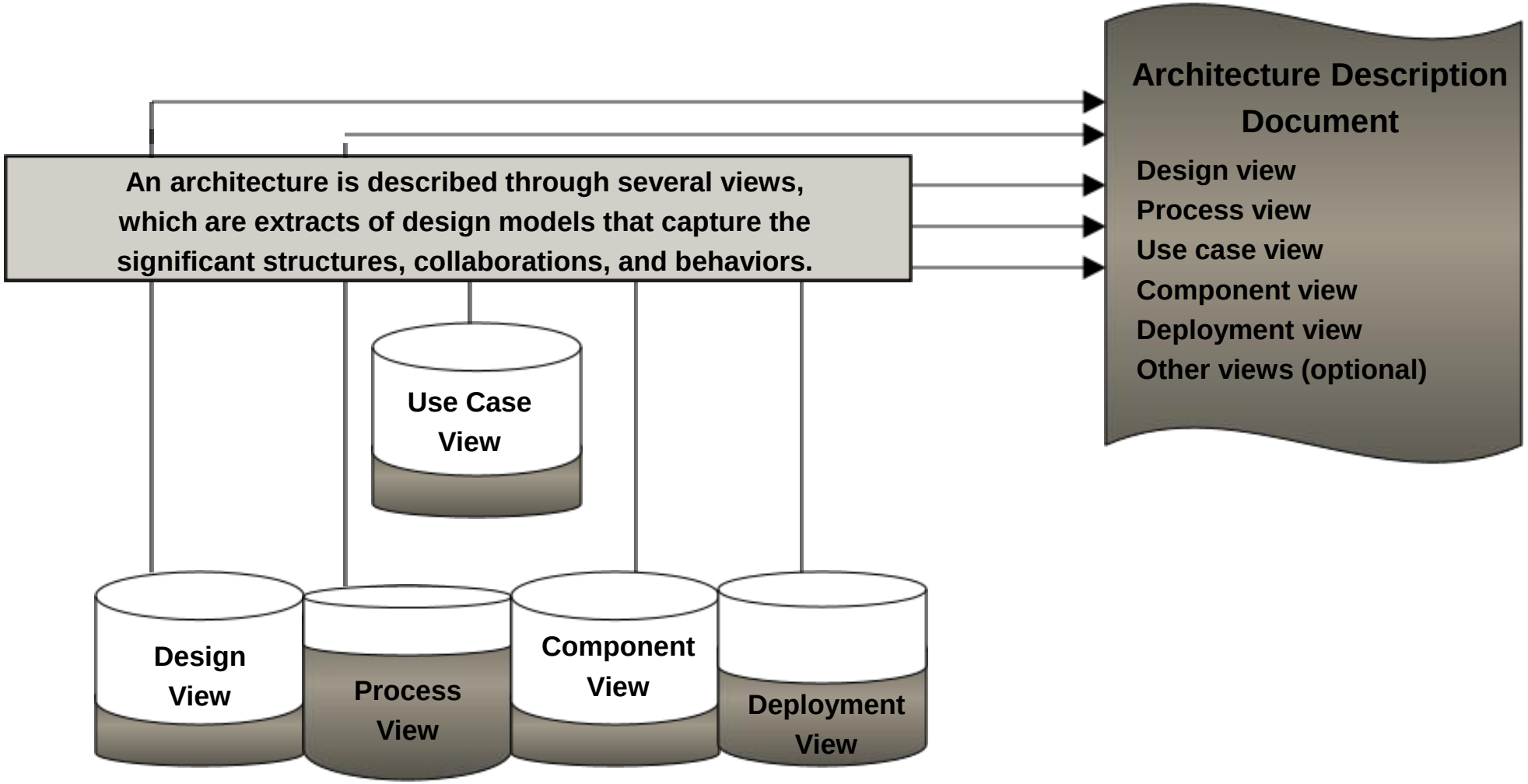
An architecture framework is defined in terms of views that are abstractions of the UML models in the design set. The design model includes the full breadth and depth of information. An architecture view is an abstraction of the design model; it contains only the architecturally significant information. Most real-world systems require four views: design, process, component, and deployment.

- **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

Model-Based Software Architectures

A Technical 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 :



Model-Based Software Architectures

A Technical Perspective

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

Model-Based Software Architectures

A Technical Perspective

- 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 (allocation to processes and threads of control), interprocess communication, and state management. This view is modeled statically using deployment diagrams, and dynamically using any of the UML behavioral diagrams.
- 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.
- The deployment view addresses the executable realization of the system, including the allocation of logical processes in the distribution view (the logical software topology) to physical resources of the deployment network (the physical system topology). It is modeled statically using deployment diagrams, and dynamically using any of the UML behavioral diagrams.

2.4 Software Process Workflow and Iteration Workflow

Workflows of the Process

Software Process Workflows

The term WORKFLOWS is used to mean a thread of cohesive and mostly sequential activities. Workflows are mapped to product artifacts 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

Workflows of the Process

Software Process Workflows

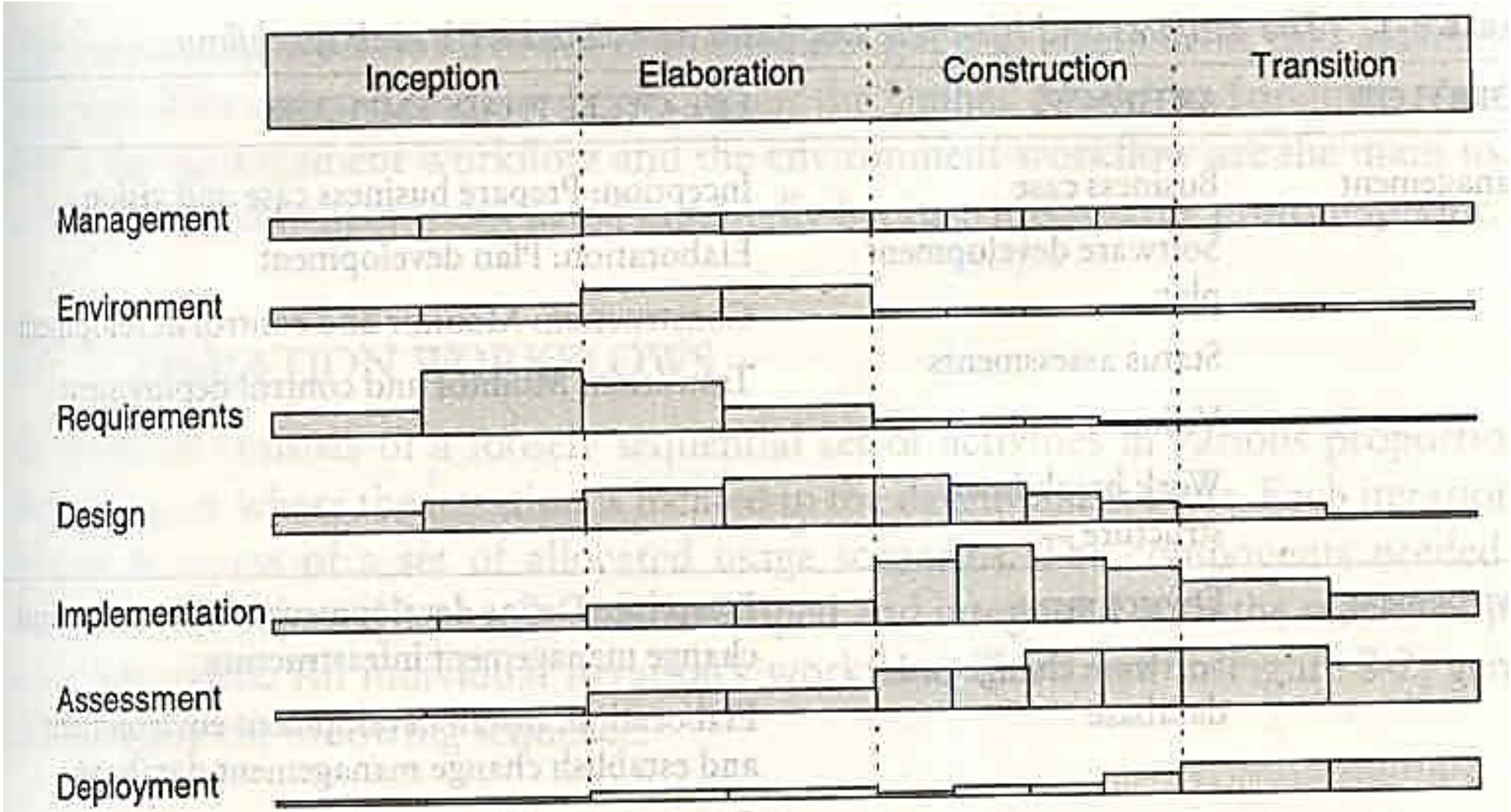


FIGURE 8-1. *Activity levels across the life-cycle phases*

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

Workflows of the Process

Software Process Workflows

Four Basic Key Principles:

1. ***Architecture-first approach***: **implementing and testing the architecture must precede full-scale development and testing and must precede the downstream focus on completeness and quality of the product features.**
2. ***Iterative life-cycle process***: **the activities and artifacts of any given workflow may require more than one pass to achieve adequate results.**
3. ***Roundtrip engineering***: **Raising the environment activities to a first-class workflow is critical; the environment is the tangible embodiment of the project's process and notations for producing the artifacts.**
4. ***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.**

Workflows of the Process

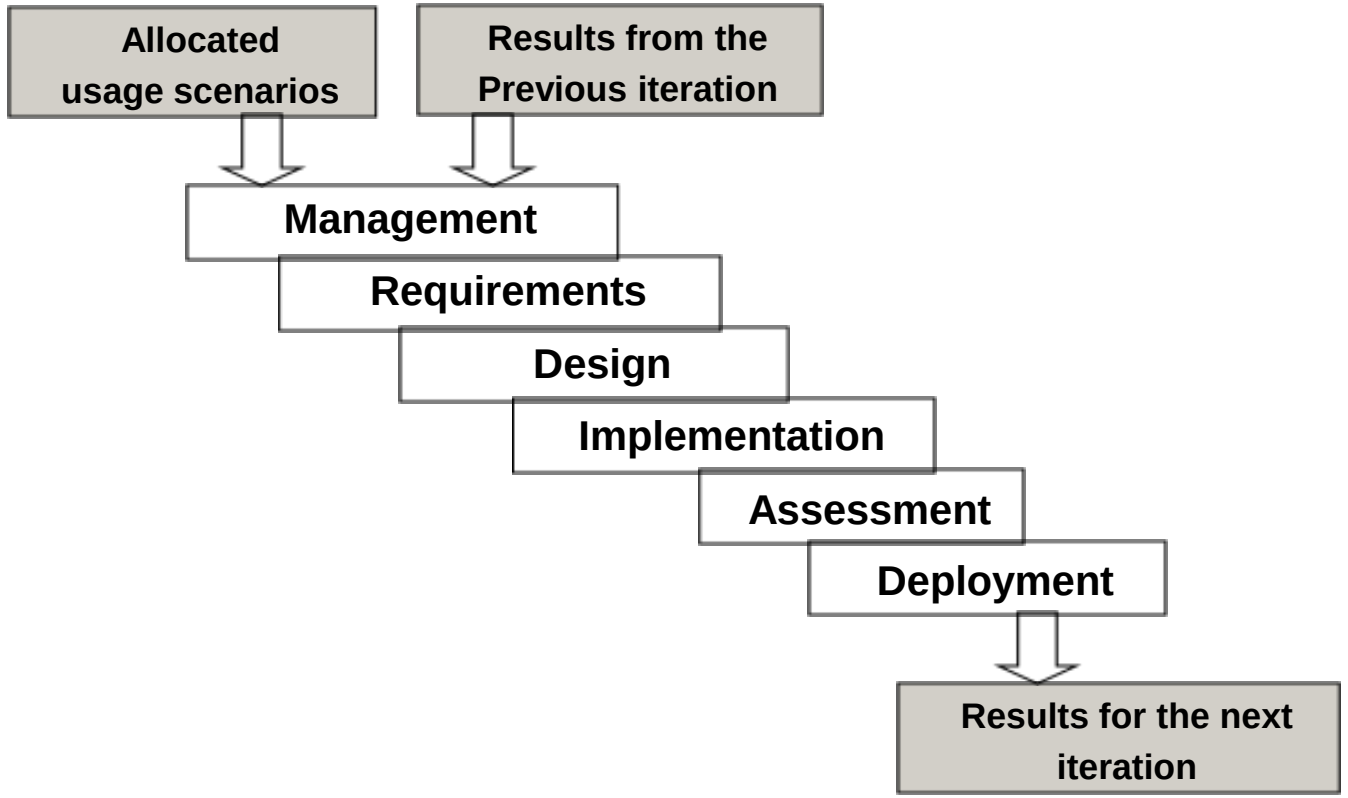
Iteration Workflows

- An iteration consist of sequential set of activities in various proportions, depending on where the iteration is located in the development cycle.
- 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.
- An individual iteration's workflow, illustrated in Figure 8-2, generally includes the following sequence:

Workflows of the Process

Iteration Workflows

An individual iteration's workflow:



Workflows of the Process

Iteration Workflows

- **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

Workflows of the Process

Iteration Workflows

- **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)

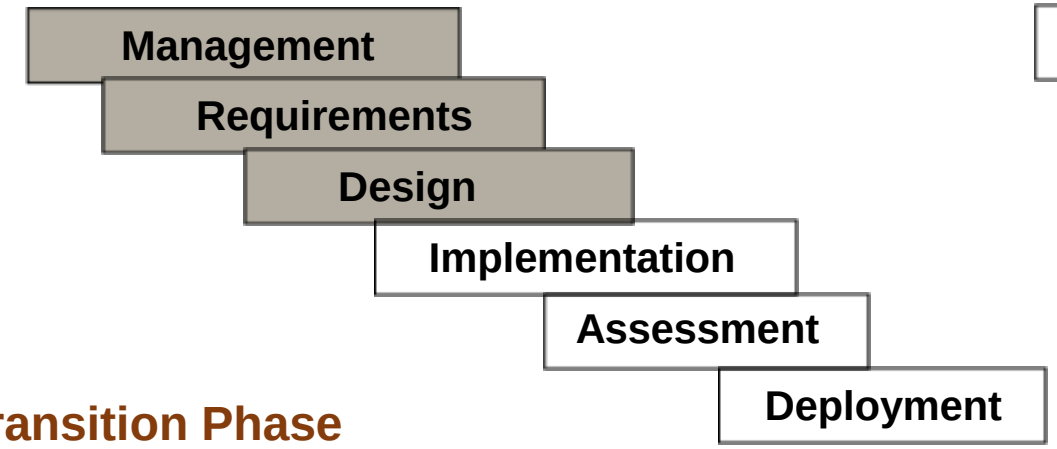
- **Assessment**: evaluating the results of the iteration, including compliance with the allocated evaluation criteria and the quality of the current baselines; identifying any rework required and determining whether it should be performed before deployment of this release or allocated to the next release; 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

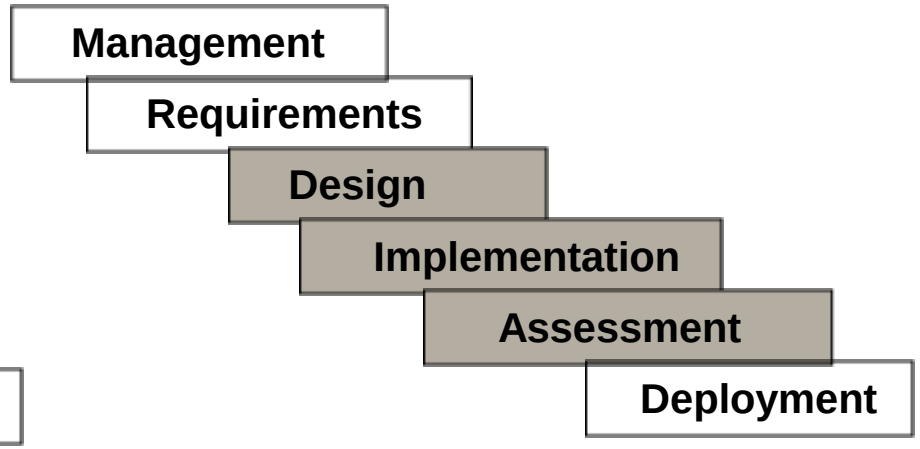
Workflows of the Process

Iteration Workflows

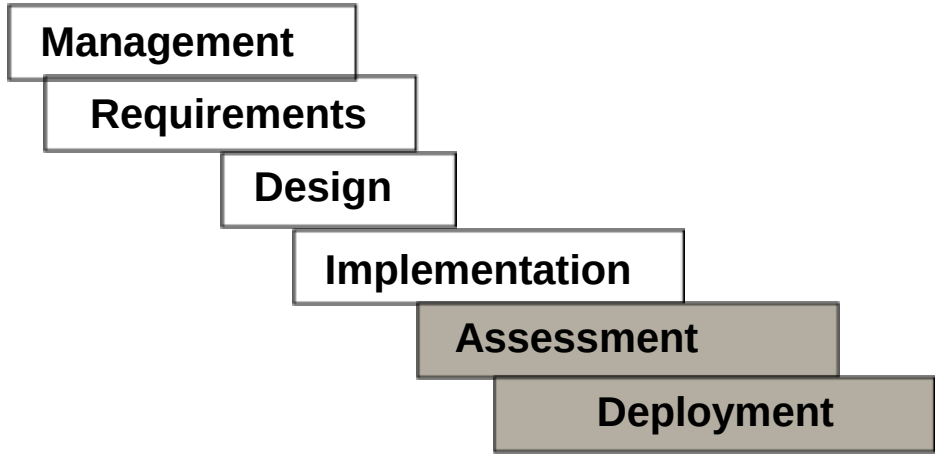
Inception and Elaboration Phases



Construction Phase



Transition Phase



Workflows of the Process

Iteration vs. Increment

- An iteration represents the state of the overall architecture and the complete deliverable system.
- An increment represents the current work in progress that will be combined with the preceding iteration to form the next iteration.

2.5 Status Monitoring - Software Process Checkpoints and Milestones

Checkpoints of the Process

- It is important to have visible milestones in the life cycle , where various stakeholders meet to discuss progress and planes.
- The purpose of this events is to:
  - Synchronize stakeholder expectations and achieve concurrence on the requirements, the design, and the plan.
  - Synchronize related artifacts into a consistent and balanced state
  - Identify the important risks, issues, and out-of-rolerance conditions
  - Perform a global assessment for the whole life-cycle.

Checkpoints of the Process

Three types of joint management reviews are conducted throughout the process:

1. **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.

2. **Minor milestones.** These iteration-focused events are conducted to review the content of an iteration in detail and to authorize continued work.

3. **Status assessments.** These periodic events provide management with frequent and regular insight into the progress being made.

Checkpoints of the Process

Figure 9-1 illustrates a typical sequence of project checkpoints for a relatively large project.

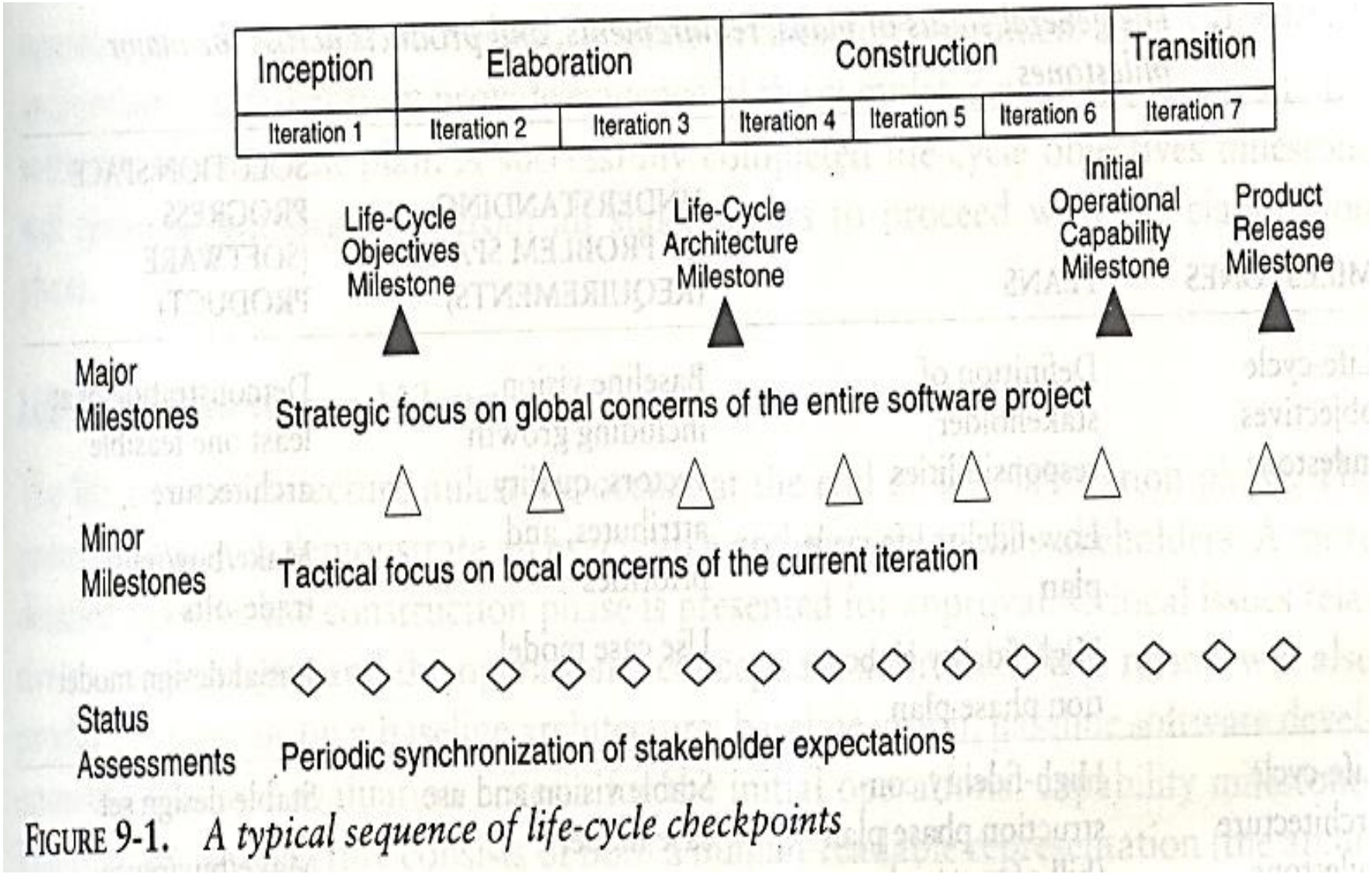


FIGURE 9-1. *A typical sequence of life-cycle checkpoints*

Major Milestones

- The four major milestones occur at the transition points between life-cycle phases.
- They can be used in many different process models, including the conventional waterfall model.
- In an iterative model, the major milestones are used to achieve concurrence among all stakeholders on the current state of the project.
- Different stakeholders have very different concerns:
  - Customers
  - Users
  - Architects and systems engineers
  - Developers
  - Maintainers
  - Others

Major Milestones

- **Customers**: schedule and budget estimates, feasibility, risk assessment, requirements understanding, progress, product line compatibility
- **Users**: consistency with requirements and usage scenarios, potential for accommodating growth, quality attributes
- **Architects and systems engineers**: product line compatibility, requirements changes, trade-off analyses, completeness and consistency, balance among risk, quality, and usability
- **Developers**: sufficiency of requirements detail and usage scenario descriptions, . frameworks for component selection or development, resolution of development risk, product line compatibility, sufficiency of the development environment
- **Maintainers**: sufficiency of product and documentation artifacts, understandability, interoperability with existing systems, sufficiency of maintenance environment
- **Others**: possibly many other perspectives by stakeholders such as regulatory agencies, independent verification and validation contractors, venture capital investors, subcontractors, associate contractors, and sales and marketing teams

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

Major Milestones

Life-Cycle Objectives Milestone

The life-cycle objectives milestone occurs at the end of the inception phase. The goal is to present to all stakeholders a recommendation on how to proceed with development, including a plan, estimated cost and schedule, and expected benefits and cost savings. A successfully completed life-cycle objectives milestone will result in authorization from all stakeholders to proceed with the elaboration phase.

Life-Cycle Architecture Milestone

The life-cycle architecture milestone occurs at the end of the elaboration phase. The primary goal is to demonstrate an executable architecture to all stakeholders. The baseline architecture consists of both a human-readable representation (the architecture document) and a configuration-controlled set of software components captured in the engineering artifacts. A successfully completed life-cycle architecture milestone will result in authorization from the stakeholders to proceed with the construction phase.

I. **Requirements**
   A. Use case model
   B. Vision document (text, use cases)
   C. Evaluation criteria for elaboration (text, scenarios)

II. **Architecture**
   A. Design view (object models)
   B. Process view (if necessary, run-time layout, executable code structure)
   C. Component view (subsystem layout, make/buy/reuse component identification)
   D. Deployment view (target run-time layout, target executable code structure)
   E. Use case view (test case structure, test result expectation)
      1. Draft user manual

III. **Source and executable libraries**
   A. Product components
   B. Test components
   C. Environment and tool components

FIGURE 9-2. *Engineering artifacts available at the life-cycle architecture milestone*

Presentation Agenda

- **I. Scope and objectives**
  - A. Demonstration overview
- **II. Requirements assessment**
  - A. Project vision and use cases
  - B. Primary scenarios and evaluation criteria
- **III. Architecture assessment**
  - A. Progress
    1. Baseline architecture metrics (progress to date and baseline for measuring future architectural stability, scrap, and rework)
    2. Development metrics baseline estimate (for assessing future progress)
    3. Test metrics baseline estimate (for assessing future progress of the test team)
  - B. Quality
    1. Architectural features (demonstration capability summary vs. evaluation criteria)
    2. Performance (demonstration capability summary vs. evaluation criteria)
    3. Exposed architectural risks and resolution plans
    4. Affordability and make/buy/reuse trade-offs
- **IV. Construction phase plan assessment**
  - A. Iteration content and use case allocation
  - B. Next iteration(s) detailed plan and evaluation criteria
  - C. Elaboration phase cost/schedule performance
  - D. Construction phase resource plan and basis of estimate
  - E. Risk assessment

Demonstration Agenda

- **I. Evaluation criteria**
- **II. Architecture subset summary**
- **III. Demonstration environment summary**
- **IV. Scripted demonstration scenarios**
- **V. Evaluation criteria results and follow-up items**

FIGURE 9-3. *Default agendas for the life-cycle architecture milestone*

Major Milestones

Initial Operational Capability Milestone

The initial operational capability milestone occurs late in the construction phase. The goals are to assess the readiness of the software to begin the transition into customer/user sites and to authorize the start of acceptance testing. Acceptance testing can be done incrementally across multiple iterations or can be completed entirely during the transition phase is not necessarily the completion of the construction phase.

Product Release Milestone

The product release milestone occurs at the end of the transition phase. The goal is to assess the completion of the software and its transition to the support organization, if any. The results of acceptance testing are reviewed, and all open issues are addressed. Software quality metrics are reviewed to determine whether quality is sufficient for transition to the support organization.

Minor Milestones

For most iterations, which have a one-month to six-month duration, only two minor milestones are needed: the iteration readiness review and the iteration assessment review.

Iteration Readiness Review.
This informal milestone is conducted at the start of each iteration to review the detailed iteration plan and the evaluation criteria that have been allocated to this iteration .

Iteration Assessment Review.
This informal milestone is conducted at the end of each iteration to assess the degree to which the iteration achieved its objectives and satisfied its evaluation criteria, to review iteration results, to review qualification test results (if part of the iteration), to determine the amount of rework to be done, and to review the impact of the iteration results on the plan for subsequent iterations.

Minor Milestones

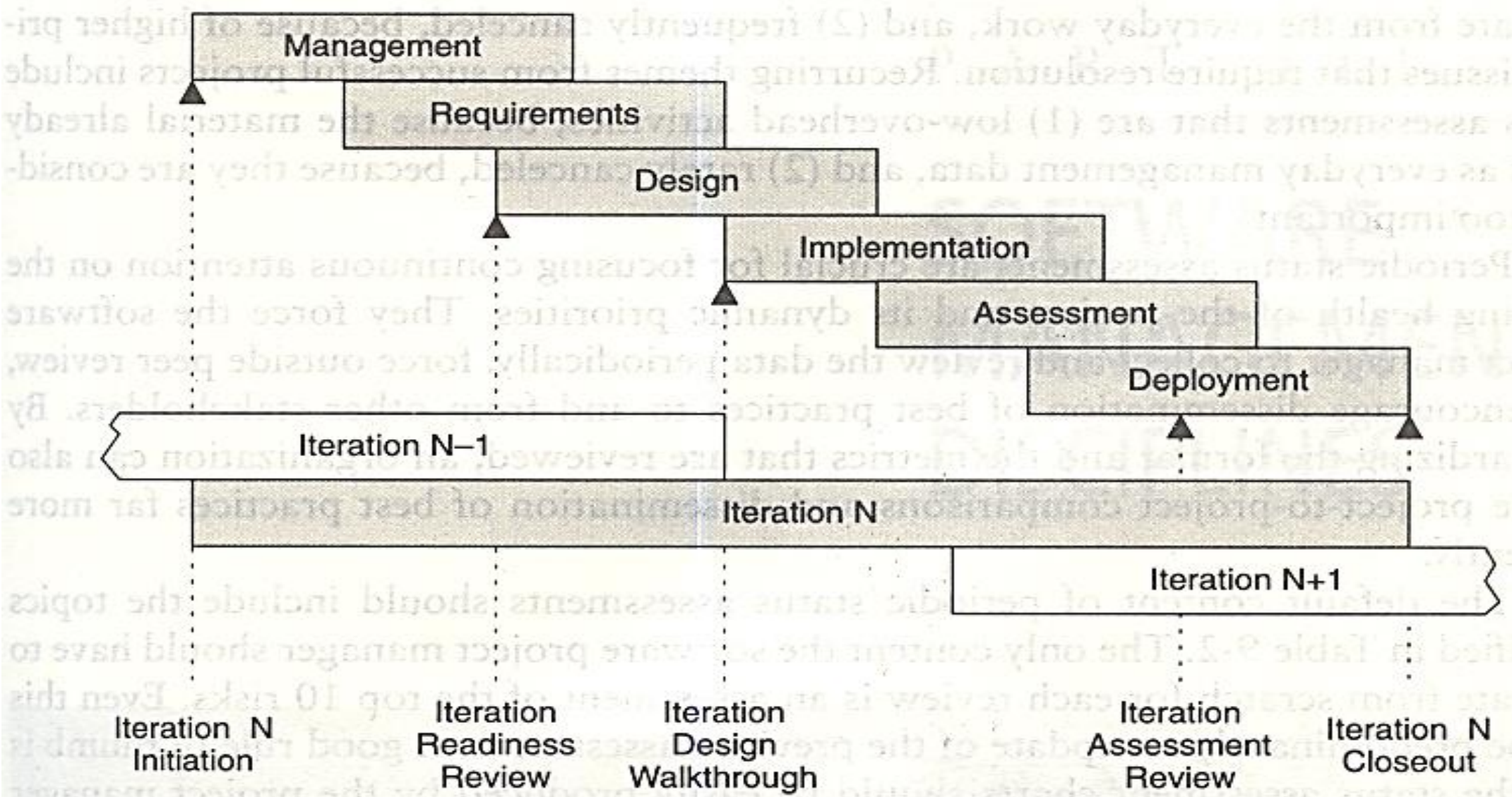


FIGURE 9-4. *Typical minor milestones in the life cycle of an iteration*

Unit 2: Important Questions

1.	Explain briefly two stages of the life cycle engineering and production.
2.	Explain different phases of the life cycle process?
3.	Explain the goal of Inception phase, Elaboration phase, Construction phase and Transition phase.
4.	Explain the overview of the artifact set
5.	Write a short note on (a) Management Artifacts (b) Engineering Artifacts (c) Pragmatic Artifacts
6.	Define Model-Based software architecture?
7.	Explain various process workflows?
8.	Define typical sequence of life cycle checkpoints?
9.	Explain general status of plans, requirements and product across the major milestones.


Any Questions
?

Book References:

1. Software Project Management – A Unifies Framework, Walker Royce, 1998, Addison Wesley
2. Software Project Management – From Concept to Deployment, Conway, K., 2001.
3. Software Project Management, Bob Hughes and Mike Cotterell, Latest Publication
4. Software Project Management, Rajeev Chopra, 2009
5. Software Engineering – A Practitioner's approach, Roger S. Pressman Latest Plublication
6. Managing Global Software Projects, Ramesh, 2001, TMH


Thank You