ESPM UNIT - III

Model Based Software Architectures: A Management Perspective and Technical Perspective.
Work Flows of the Process: Software Process Workflows, Iteration Workflows.
Checkpoints of the Process: Major Mile Stones, Minor Milestones, Periodic Status Assessments.

ARCHITECTURE: A MANAGEMENT PERSPECTIVE

- The most critical technical product of a software project is its architecture: the infrastructure, control and date interfaces that permit software components to co-operate as a system and software designers to co-operate efficiently as a tem. When the communications media include multiple languages and inter group 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.
- From a management perspective, there are three difference aspects of architecture.
    - ***An architecture*** (the intangible design concept) is the design of a software system this includes all engineering necessary to specify a complete bill of materials.
    - ***An architecture baseline*** (the tangible artifacts) is a slice of information across the engineering artifact sets sufficient to satisfy all stakeholders that the vision (function and quality) can be achieved within the parameters of the business case (cost, profit, time, technology and people).
    - ***An architecture description*** (a human-readable representation of an architecture, which is one of the components of an architecture baseline) is an organized subset of information extracted form the design set model(s). The architecture description communicates how the intangible concept is realized in the tangible artifacts.

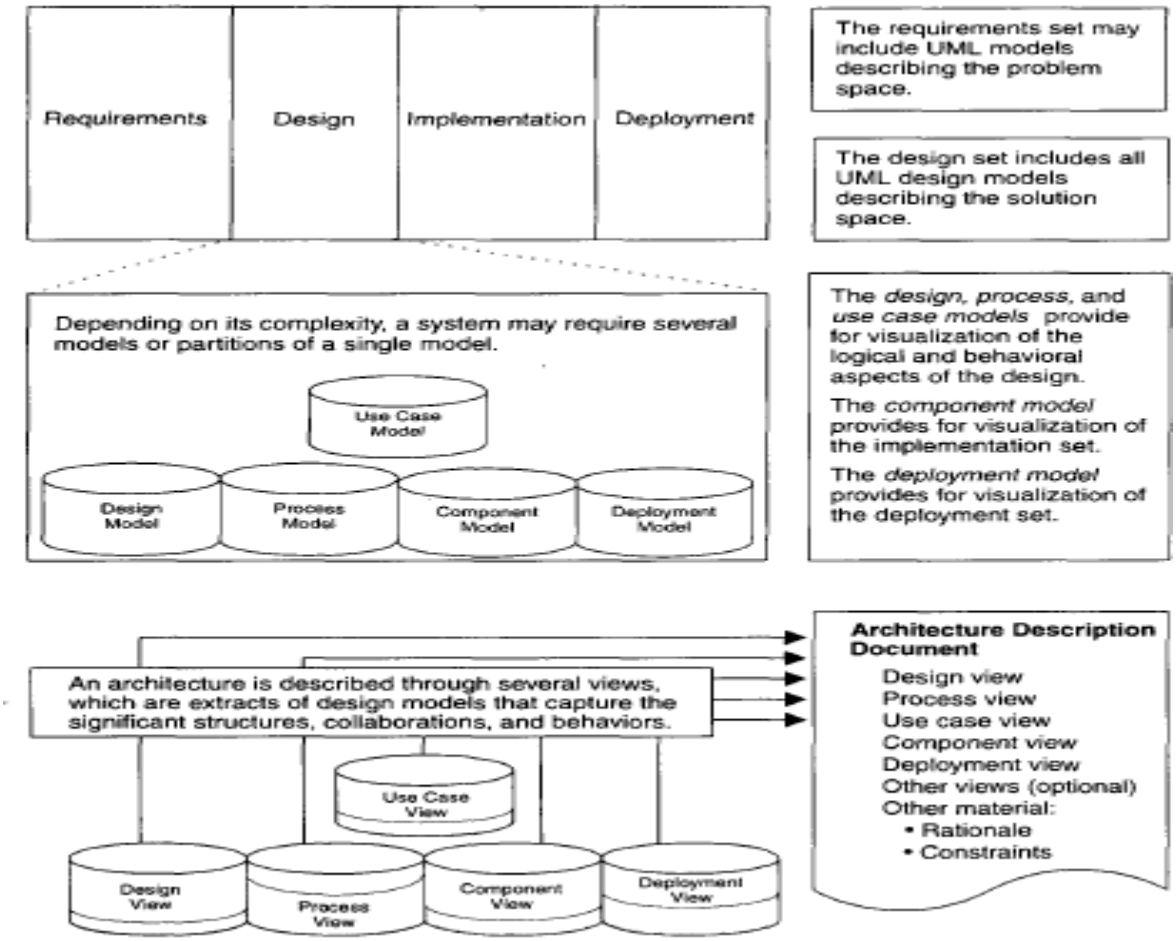
The importance of software architecture and its close linkage with modern software development processes can be summarized as follows:

- Achieving 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 fro 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**.**

ARCHITECTURE: A TECHNICAL PERSPECTIVE

- An architecture framework is defined in the 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 relationship among the design, component and deployment views.
    - ***Component:*** Describes the structure of the implementation set.
    - ***Deployment:*** Describes the structures of the deploy.

The following Figure Summarizes the artifacts of the design set, including the architecture views and architecture description:



Architecture, an organized and abstracted view into the design models

The ***requirements model*** addresses the behavior of the system as seen by its end users, analysts, and testers. This view is modeled statically using use case and class diagrams and dynamically using sequence, collaboration, state chart and activity diagrams.

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 calls and object diagrams and dynamically using any of the UML behavioral diagrams.

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 process and threads of control), inter process 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.

Generally, an architecture baseline should including 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
-

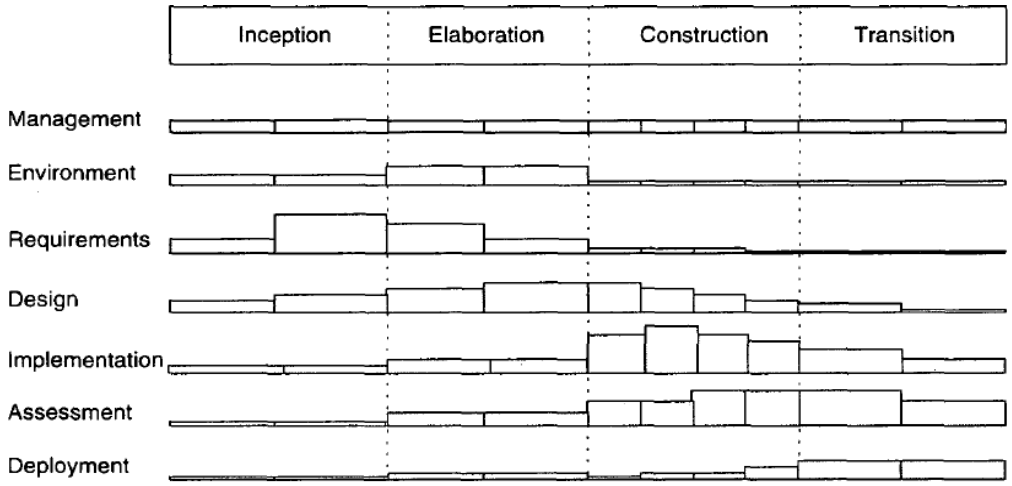
Implementation: source component inventory and bill of materials (number, name, purpose, cost) of all primitive components

- Development: executable components sufficient to demonstrate the critical us cases and the risk associated with achieving the system qualities.

SOFTWARE PROCESS WORKFLOWS

The term ***workflow*** is used to mean a thread of cohesive and mostly sequential activities; Workflows are mapped to product artifacts. There are seven top-level workflows:

1. Management workflow: controlling the process and ensuring win conditions for all stakeholders.
2. Environment workflow: automating the process and evolving the maintenance environment.
3. Requirements workflow: analyzing the problem space and evolving the requirements artifacts.
4. Design workflow: modeling the solution and evolving the architecture and designartifacts.
5. Implementation workflow: programming components & evolving the implementation and deployment artifacts.
6. Assessment workflow: assessing the trends in process and product quality.
7. Deployment workflow: transitioning the end products to the user.



Activity levels across the life-cycle phases

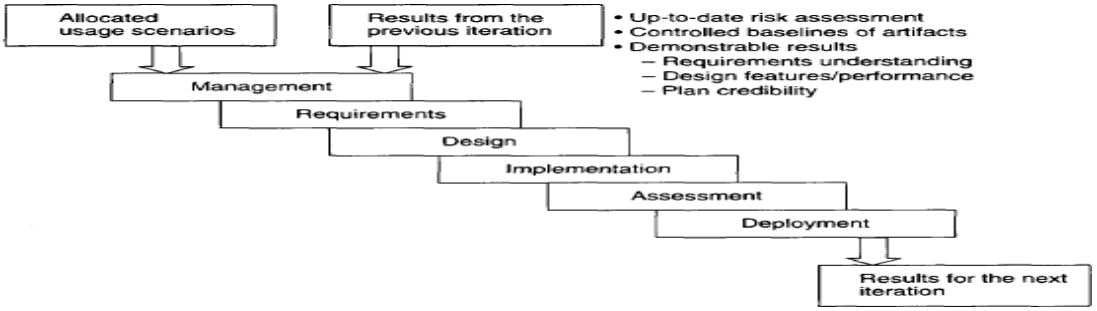
1. Architecture –first approach: Extensive requirements analysis, design, implementation and assessment activities are performed before the construction phase when full-scale implementation is the focus.
2. Iterative life-cycle process: Some projects may require only one iteration in a phase, others may require several iterations. The point is that the activities and artifacts of any given workflow may require more than one pass to achieve results
3. Round-trip engineering: Raising the environment activities to a first-class workflow is critical. The environment is the tangible embodiment of the projects process, methods and notations for producing the artifacts.
4. Demonstration-based approach: Implementation and assessment activities are initiated early in the life cycle, reflecting the emphasis on constructing executable subsets of the evolving architecture.

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 WORKFLOWS

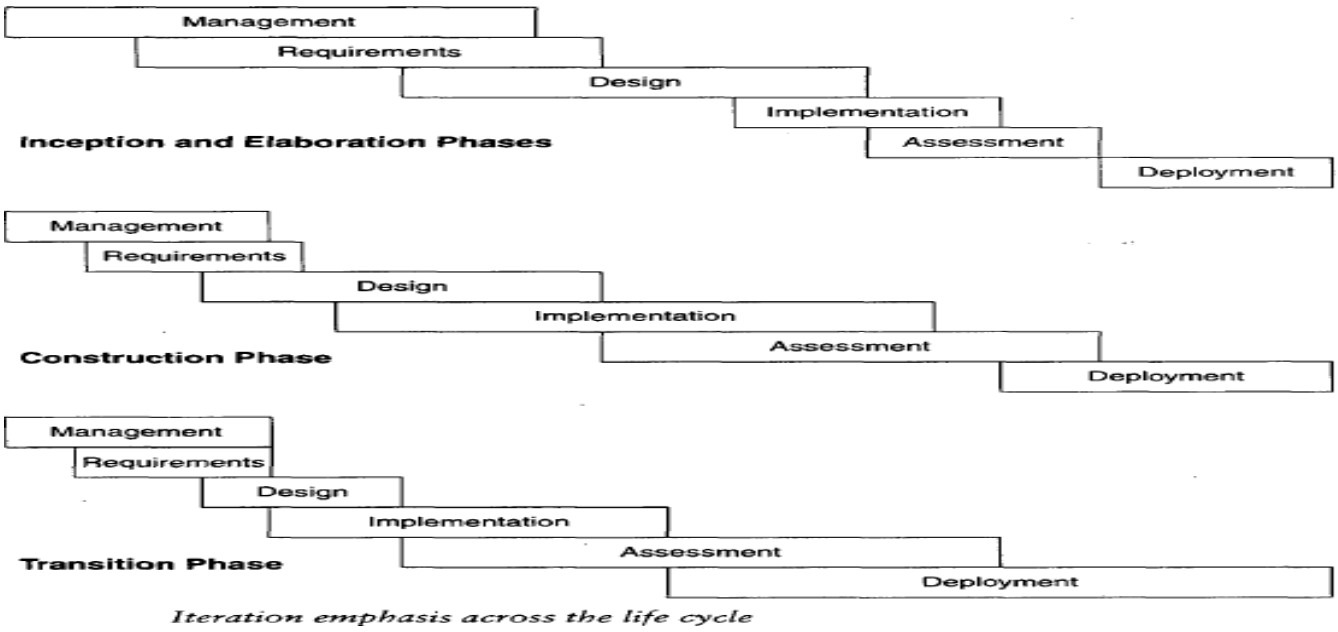
Iteration consists of a loosely sequential set of activities in various proportions, depending on where the iteration is located in the development cycle. Each iteration is defined in terms of a set of allocated usage scenarios.



The workflow of an iteration

An individual iteration's workflow generally includes the following sequence:

- **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 detem1ining 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.



Iteration emphasis across the life cycle

CHECKPOINTS OF THE PROCESS

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

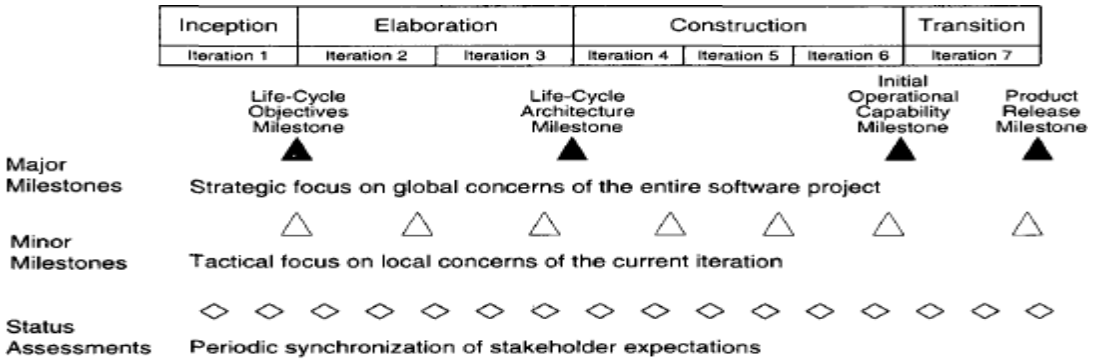
- **Major Milestones**: these system wide events are held ant the end of each development phase. They provide visibility to system wide issues synchronize the management and engineering perspectives and verify that the aims of the phase have been achieved.
- **Minor Milestones**: theses iteration-focused 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.

Each of the four phases-inceptions, elaboration, construction and transition consists of one or more iterations and concludes with a major milestone when a planned technical capability is produced in demonstrable form.

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 eth 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.

- **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 sale and marketing teams.



A typical sequence of life-cycle checkpoints

The following Table summarizes the balance of information across them major milestones.

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

Life-Cycle Objectives Milestone

The life-cycle objectives milestone occurs at tile 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 and a configuration-controlled set of software components captured in the engineering artifacts.

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

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**

Default agendas for the life-cycle architecture milestone

MINOR MILESTONES

- Minor milestones are sometimes called as inch-pebbs.
- Minor milestones mainly focus on local concerns of current iteration.
- These iterative focused events are used to review iterative content in a detailed manner & authorize continued work.

Minor Milestone in the life cycle of Iteration: The number of iteration specific milestones is dependent on the iteration length and the content. A one month to six month iterative period requires only two minor milestones

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

b) **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.

PERIODIC STATUS ASSESSMENTS

- Periodic status assessments are management reviews conducted at regular intervals (monthly, quarterly) to address progress and quality indicators, ensure continuous attention to project dynamics, and maintain open communications among all stakeholders.
- Periodic status assessments are crucial for focusing continuous attention on the evolving health of the project and its dynamic priorities.
- Periodic status assessments serve as project snapshots. While the period may vary, the recurring event forces the project history to be captured and documented. Status assessments provide the following:
  a) A mechanismfor openly addressing, communicating and resolving management issues technical issues and project risks.
  b) Objective data derived directly from on-going activities and evolving product configurations
  c) A mechanism for disseminating process, progress, quality trends, practices, and experience information to and from all stakeholders in an open forum.

Default content of status assessment reviews

TOPIC	CONTENT
Personnel	Staffing plan vs. actuals
	Attritions, additions
Financial trends	Expenditure plan vs. actuals for the previous, current, and next major milestones
	Revenue forecasts
Top 10 risks	Issues and criticality resolution plans
	Quantification (cost, time, quality) of exposure
Technical progress	Configuration baseline schedules for major milestones
	Software management metrics and indicators
	Current change trends
	Test and quality assessments
Major milestone plans and results	Plan, schedule, and risks for the next major milestone
	Pass/fail results for all acceptance criteria
Total product scope	Total size, growth, and acceptance criteria perturbations