UNIT - III

Workflows and Checkpoints of process, Software process workflows, Iteration workflows, Major milestones, minor milestones, periodic status assessments. Process Planning Work breakdown structures, Planning guidelines, cost and schedule estimating process, iteration planning process, Pragmatic planning.

SOFTWARE MANAGEMENT PROCESS FRAMEWORK:

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 with 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 design artifacts
5. Implementation workflow: programming the components and 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

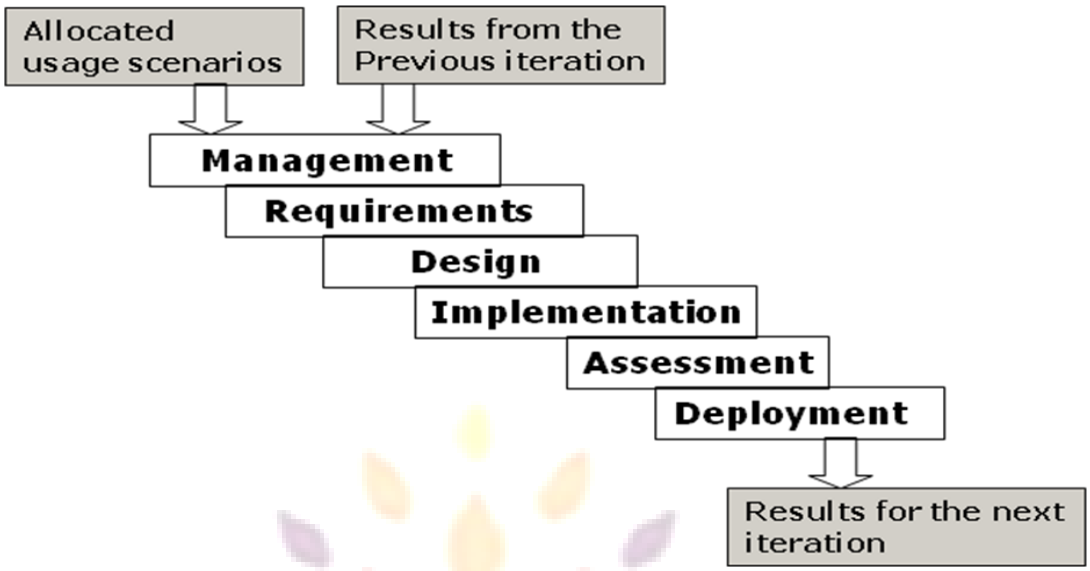
Four basic key principles of the modern process frame work:

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.

Iterative life-cycle process: the activities and artifacts of any given workflow may require more than one pass to achieve adequate results.

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.

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.

Explain in detail about the iteration workflows of the software process?

Iteration consists of 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 se t of allocated usage scenarios. The components needed to implement all selected scenarios are developed and integrated with the results of previous iterations. An individual iteration's workflow illustrated in 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 the demonstrated at the end of the iteration and their evaluation criteria.

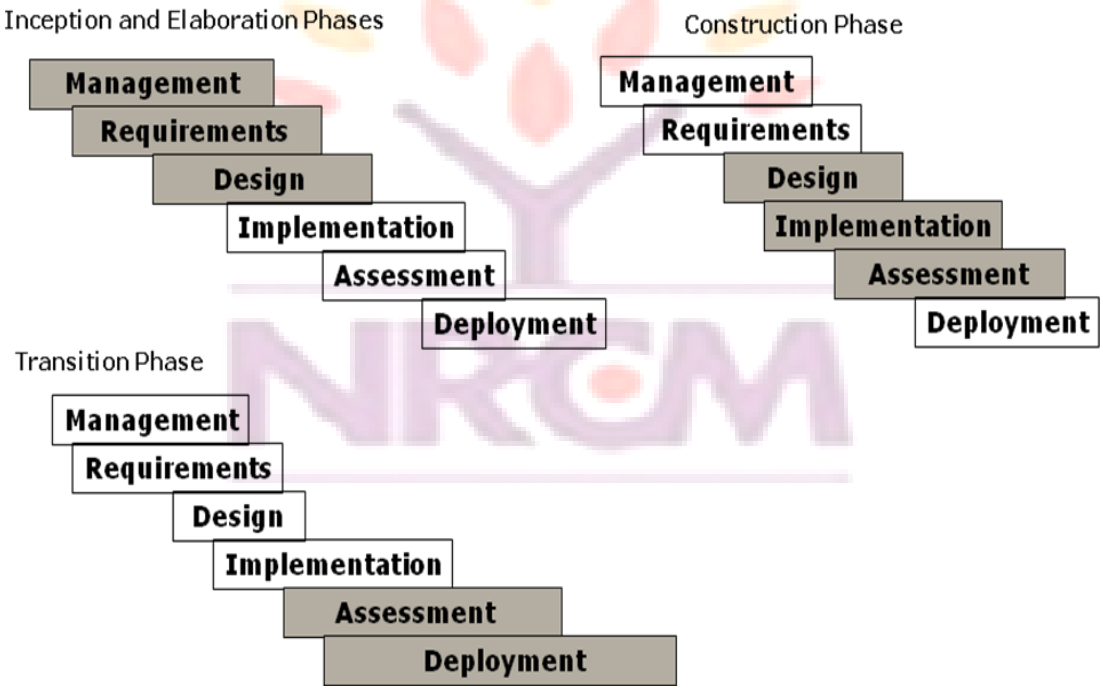
Design: Evolving the baseline architecture and the baseline design set artifacts to elaborate fully the design model and test model components necessary to demonstrate against the evolution criteria allocated to this iteration.

Implementation: developing any new components, and enhancing or modifying any existing components, to demonstrate the evolution criteria allocated to this iteration

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.

Deployment: transitioning the released either to an external organization or to internal closure by conducting a post mortem so that lessons learned can be captured and reflected in the next iteration.

The following is an example of a simple development life cycle, illustrates the difference between iterations and increments. This example also illustrates a typical build sequence from the perspective of an abstract layered architecture.



Iteration emphasis across the life cycle

CHECK POINTS 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.

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.

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

Major milestones –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 – iteration-focused events, conducted to review the content of iteration in detail and to authorize continued work.

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

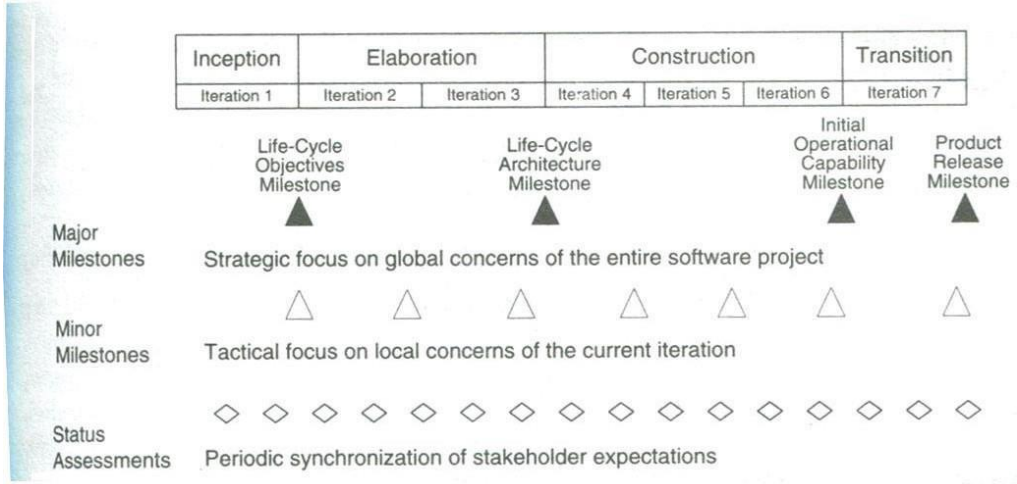


FIGURE 9-1. *A typical sequence of life-cycle checkpoints*

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: 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.

Architectures and systems engineers: product line compatibility, requirements change, tradeoff analyses, completeness and consistency, balance among risk, quality, and usability.

Developers: sufficiency of requirements detail and usage scenario descriptions, frameworks for component selection of development, resolution of development risk, 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.

The milestones may be conducted as one continuous meeting of all concerned parties or incrementally through mostly on-line review of the various artifacts. There are considerable differences in the levels of ceremony for these events depending on several factors.

The essence of each major milestone is to ensure that the requirements understanding, the life-cycle plans, and the product's form, function, and quality are evolving in balanced levels of detail and to ensure consistency among the various artifacts. The following table summarizes the balance of information across the major milestones.

MINOR MILESTONES

All iterations are not created equal. An iteration can take on very different forms and priorities, depending on where the project is in the life cycle. Early iterations focus on analysis and design with substantial elements of discovery, experimentation, and risk assessment. Later iterations focus much more on completeness, consistency, usability, and change management.

Iteration readiness review: this informal milestone is conducted at the start of each iteration to review the detailed iteration plan the evolution 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 of which the iteration achieved its objectives and satisfied its evaluation criteria, to review iteration achieved its objectives and satisfied its evaluation criteria, to review iteration results, to review qualification test results, to determine the amount of rework to be done, and to review the impact of the iteration results on the plan for subsequent iterations.

PERIODIC STATUS ASSESSMENTS

Periodic stats assessments are management reviews conducted at regular intervals to address progress and quality indicators, ensure continuous attention to project dynamics, and maintain open communications among all stakeholders.

Status assessments provide the following:

A mechanism for openly addressing, communicating, and resolving management issues, technical issues, and project risks

Objective data directly from on-going activities and evolving product configurations

A mechanism for disseminating process, progress quality trends, practices and experience information to and from all stakeholders in an open forum.

The default content of periodic status assessments should include the topics identified in the following ta

ITERATIVE PROCESS PLANNING

A WBS is simply a hierarchy of elements that decomposes the project plan into the discrete work tasks.

A WBS provides the following information structure:

A delineation of all significant work A clear task decomposition for assignment of responsibilities

A framework for scheduling, budgeting, and expenditure tracking.

The development of a work breakdown structure is dependent on the project management style, organizational culture, customer preference, financial constraints and several other hard- to-define parameters.

Conventional WBS Issues:

Conventional WBS frequently suffer from three fundamental flaws:

Conventional WBS are prematurely structured around the productdesign:

Once this structure is ingrained in the WBS and then allocated to responsible managers with budgets, schedules and expected deliverables, a concrete planning foundation has been set that is difficult and expensive to change.

Conventional WBS are prematurely decomposed, planned, and budgeted in wither too much or too little detail:

Large software projects tend to be over planned and small projects tend to be under planned. The WBS shown in the above figure is overly simplistic for most large-scale systems, where size or more levels of WBS elements are commonplace.

Conventional WBS are project-specific, and cross-project comparisons are usually difficult or impossible:

Most organizations allow individual projects to define their own project-specific structure tailored to the project manager's style, the customer's demands, or other project-specific preferences.

It is extremely difficult to compare plans, financial data, schedule data, organizational efficiencies, cost trends, productivity tends, or quality tends across multiple projects.

Some of the following simple questions, which are critical to any organizational process improvement program, cannot be answered by most project teams that use conventional WBS.

What is the ratio of productive activities to overhead activities?

What is the percentage of effort expanded in rework activities?

What is the percentage of cost expended in software capital equipment?

What is the ration of productive testing versus integration?

What is the cost of release?

Evolutionary Work Breakdown Structures:

An evolutionary WBS should organize the planning elements around the process framework rather than the product framework. The basic recommendation for the WBS is to organize the hierarchy as follows:

First level WBS elements are the workflows (Management, environment, requirement, design,

Seco boration,

Third phase.

A de is shown

The specifics

- A Management
  - AA Inception phase management
    - AAA Business case development
    - AAB Elaboration phase release specifications
    - AAC Elaboration phase WBS baselining
    - AAD Software development plan
    - AAE Inception phase project control and status assessments
  - AB Elaboration phase management
    - ABA Construction phase release specifications
    - ABB Construction phase WBS baselining
    - ABC Elaboration phase project control and status assessments
  - AC Construction phase management
    - ACA Deployment phase planning
    - ACB Deployment phase WBS baselining
    - ACC Construction phase project control and status assessments
  - AD Transition phase management
    - ADA Next generation planning
    - ADB Transition phase project control and status assessments
- B Environment
  - BA Inception phase environment specification
  - BB Elaboration phase environment baselining
    - BBA Development environment installation and administration
    - BBB Development environment integration and custom toolsmithing
    - BBC SCO database formulation
  - BC Construction phase environment maintenance
    - BCA Development environment installation and administration
    - BCB SCO database maintenance
  - BD Transition phase environment maintenance
    - BDA Development environment maintenance and administration
    - BDB SCO database maintenance
    - BDC Maintenance environment packaging and transition
- C Requirements
  - CA Inception phase requirements development
    - CAA Vision specification
    - CAB Use case modeling
  - CB Elaboration phase requirements baselining
    - CBA Vision baselining
    - CBB Use case model baselining
  - CC Construction phase requirements maintenance
  - CD Transition phase requirements maintenance

PLANNING GUIDELINES

- Software projects span a broad range of application domains. It is valuable but risky to make specific planning recommendations independent of project context. Project-independent planning advice is also risky. There is the risk that the guidelines may be adopted blindly without being adapted to specific project circumstance. Two simple planning guidelines should be considered when a project plan is being initiated or assessed. The first guideline, detailed in Table 10-1, prescribes a default allocation of costs among the first-level WBS elements. The second guideline, detailed in Table 10-25, prescribes allocation of effort and schedule across the lifecycle phases.

Web budgeting defaults

First Level WBS Element	Default Budget
Management	10%
Environment	10%
Requirement	10%
Design	15%
Implementation	25%
Assessment	25%
Deployment	5%
Total	100%

Table 10-2 Default distributions of effort and schedule by phase

Domain	Inception	Elaboration	Construction	Transition
Effort	5%	20%	65%	10%
Schedule	10%	30%	50%	10%

THE COST AND SCHEDULE ESTIMATING PROCESS

- Project plans need to be derived from two perspectives. The first is a forward-looking, top-down approach. It starts with an understanding of the general requirements and constraints, derives a macro-level budget and schedule, then decomposes these elements into lower level budgets and intermediate milestones. From this perspective, the following planning sequence would occur:
  - The software project manager (and others) develops a characterization of the overall size, process, environment, people, and quality required for the project.
  - The software project manager partitions the estimate for the effort into top-level WBS using guidelines such as those in Table 10-1.
  - At this point, subproject managers are given the responsibility for decomposing each of the WBS elements into lower levels using their top-level allocation, staffing profile, and major milestone dates as constraints.
- The second perspective is a backward-looking, bottom-up approach. We start with the end in mind, analyze the micro-level budgets and schedules, then sum all these elements into the higher level budgets and intermediate milestones. This approach tends to define and populate the WBS from the lowest levels upward. From this perspective, the following planning sequence would occur:
  1. The lowest level WBS elements are elaborated into detailed tasks
  2. Estimates are combined and integrated into higher level budgets and milestones.
  3. Comparisons are made with the top-down budgets and schedule milestones.

Engineering Stage		**Production Stage**	
Inception	Elaboration	Construction	Transition
Engineering stage planning emphasis:		**Production stage planning emphasis:**	

▪ Macro level task estimation for production stage artifacts ▪ Micro level task estimation for engineering artifacts ▪ Stakeholder concurrence ▪ Coarse grained variance analysis of actual Vs planned expenditures ▪ Tuning the top down project independent planning guidelines into project specific planning guidelines ▪ WBS definition and elaboration	▪ Micro level task estimation for production stage artifacts ▪ Macro level task estimation for maintenance of engineering artifacts ▪ Stakeholder concurrence ▪ Fine grained variance analysis of actual Vs planned expenditures

THE ITERATION PALNNING PROCESS

- Planning is concerned with defining the actual sequence of intermediate results. An evolutionary build plan is important because there are always adjustments in build content and schedule as early conjecture evolves into well-understood project circumstance. Iteration is used to mean a complete synchronization across the project, with a well-orchestrated global assessment of the entire project baseline.

Inception Iterations: the early prototyping activities integrate the foundation components of candidate architecture and provide an executable framework for elaborating the critical use cases of eth system. This framework includes existing components, commercial components and custom prototypes sufficient to demonstrate candidate architecture and sufficient requirements understanding to establish a credible business case, vision and software development plan

- **Elaboration Iteration:** These iterations result in architecture, including a complete framework and infrastructure for execution. Upon completion of the architecture iteration, a few critical use cases should be demonstrable: (1) initializing the architecture (2) injecting a scenario to drive the worst-case data processing flow through the system (for example, the

peak transaction throughput or peak loan scenario) and (3) injecting a scenario to drive the worst-case control flow through the system (for example, orchestrating the fault-tolerance use cases).

- **Construction Iterations:** Most projects require at least two major construction iterations: an alpha release and a beta release.
- **Transition Iterations:** Most projects use a single iteration to transition a beta release into the final product.
- The general guideline is that most projects will use between four and nine iteration. The typical project would have the following six-iteration profile:
    - **One iteration in inception:** an architecture prototype
    - **Two iterations in elaboration:** architecture prototype and architecture baseline
    - **Two iterations in construction:** alpha and beta releases
    - **One iteration in transition:** product release

PRAGMATIC PLANNING

- Even though good planning is more dynamic in an iterative process, doing it accurately is far easier. While executing iteration N of any phase, the software project manager must be monitoring and controlling against a plan that was initiated in iteration N-1 and must be planning iteration N+1. the art of good project management is to make trade-offs in the current iteration plan and the next iteration plan based on objective results in the current iteration and previous iterations. Aside form bad architectures and misunderstood requirement, inadequate planning (and subsequent bad management) is one of the most common reasons for project failures. Conversely, the success of every successful project can be attributed in part to good planning.
- A project's plan is a definition of how the project requirements will be transformed into a product within the business constraints. It must be realistic, it must be current, it must be a team product, it must be understood by the stake holders, and it must be used. Plans are not just for mangers. The more open and visible the planning process and results, the more ownership there is among the team members who need to execute it. Bad, closely held plans cause attrition. Good, open plans can shape cultures and encourage teamwork.