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.

Part –I Work flows and check points of process

Workflows of the Process:

- In most of the cases a process is a sequence of activities why because of easy to understand, represent, plans and conduct.
- But the simplistic activity sequences are not realistic why because it includes many teams, making progress on many artifacts that must be synchronized, cross-checked, homogenized, merged and integrated.
- In order to manage complex software's the workflow of the software process is to be changed that is distributed.
- Modern software process avoids the life-cycle phases like inception, elaboration, construction and transition. It tells only the state of the project rather than a sequence of activities in each phase.
- The activities of the process are organized in to seven major workflows:

1) Management 2) Environment 3) Requirements
4) Design 5) Implementation 6) Assessment
7) Deployment

- These activities are performed concurrently, with varying levels of effort and emphasis as a project progresses through the life cycle.
- The management workflow is concerned with three disciplines:

1) Planning 2) Project control 3) Organization

Software Process Workflows

Previous chapters introduced a life-cycle macroprocess and the fundamental sets of artifacts. The macroprocess comprises discrete phases and iterations, but not discrete activities. A continuum of activities occurs in each phase and iteration. The next-level process description is the microprocesses, or workflows, that produce the artifacts. The term workflow is used to mean a thread of cohesive and mostly sequential activities. Workflows are mapped to product artifacts as described in Chapter 6 and to project teams as described in Chapter
11. 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 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

Figure 8-1 illustrates the relative levels of effort expected across the phases in each of the top-level workflows. It represents one of the key signatures of a modern process framework and provides a viewpoint from which to discuss several of the key principles introduced in Chapter 4.

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. This early life-cycle focus on implementing and testing the architecture must proceed full-scale

2. **Iterative life-cycle process.**

"Each phase portrays at least two iterations of each workflow. This default is intended to be descriptive, **not prescriptive**. 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 … core discipline may require more than one pass to achieve adequate results."

3. Round-trip Engineering

"Raising the environment activities to a first-class … Core Supporting Discipline is critical. The environment is the tangible embodiment of the project's 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.

Iteration Workflows

- 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.

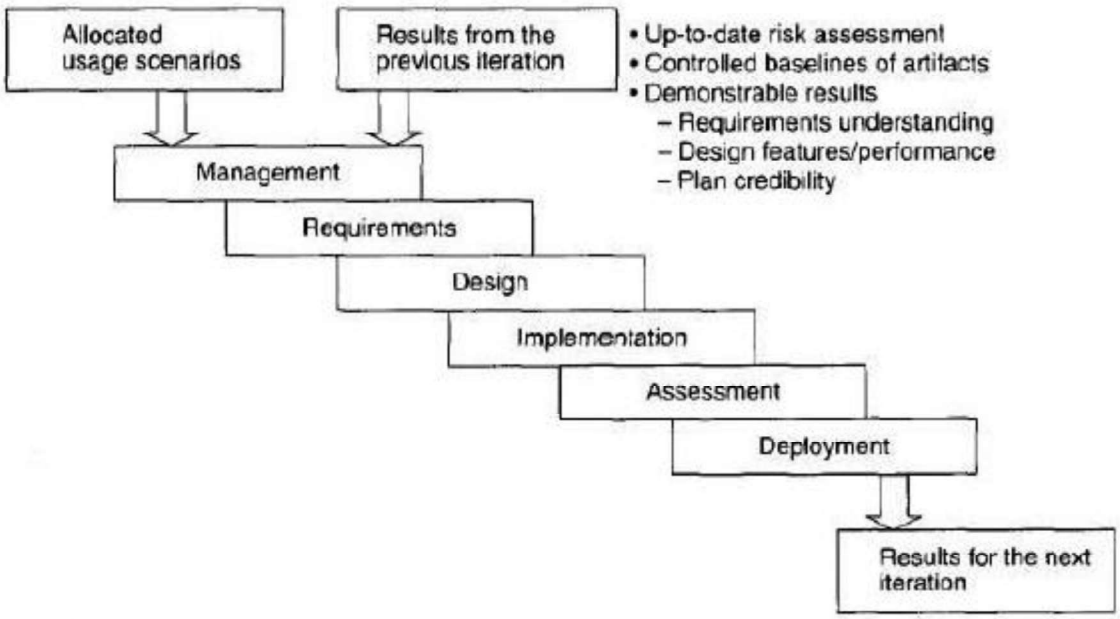


FIGURE 8-2. *The workflow of an iteration*

Workflow of an iteration

Example of usage scenario(ATM banking for the week)

- sally places her bank card into the ATM.
- sally successfully logs into the ATM using her personal identification number.

- sally deposits her weekly paycheck of $350 into her savings account.
- sally pays her phone bill of $75, her electric bill of $145, her cable bill of $55, and her water bill of $85 from her savings account
- sally attempts to withdraw $100 from her savings account forthe weekend but discovers that she has insufficient funds
- sally withdraws $40 and gets her card back

Iteration emphasis across life cycle

Iteration Contents

- Management is concerned with the content of the iterations, assigning work, and the contents of anticipated releases.
- Environment is concerned primarily with maintaining the software change database – tracking, prioritizing, addressing any changes that arise.

Requirements: is concerned with looking carefully at the **baseline plan, architecture, and requirement set artifacts** needed to fully expand the use cases that need to be demonstrated at the end of this iteration – as well as their evaluation criteria.

Design: is concerned with **developing the design model and test model** by evolving the baseline architecture and design set artifacts against the **evaluation criteria** for a specific iteration; Also need to update the design set artifacts in response to the activities within this iteration

Implementation: addresses developing (from scratch), **customizing, or acquiring** (purchase, reuse) new components and to test and **integrate** these components into to **current architectural baseline.**

Assessment: concerned with evaluating the results of each iteration to insure compliance with **evaluation criteria** and to identify **rework** needed before deployment of this release or allocated to the next iteration; also, assess the results for this iteration so as to improve the next iteration's **procedure**.

Deployment: concerned with transmitting the release either internally or to an external organization for exercise.

CHECKPOINTS OF THE PROCESS

Check pointing is a technique to add fault tolerance into computing systems. It basically consists of saving a snapshot of the application's state, so that it can restart from that point in case of failure. This is particularly important for long running application that is executed in vulnerable computing system.

- It is important to place visible milestones in the life cycle in order to discuss the progress of the project by the stakeholders and also to achieve,

1) Synchronize stakeholder expectations and achieve agreement among the requirements, the design, and the plan perspectives.

2) Synchronize related artifacts into a consistent and balanced state.

3) Identify the important risks, issues, and out-of-tolerance conditions.

4) Perform a global review for the whole life cycle, not just the current situation of an individual perspective or intermediate product.

Three sequence of project checkpoints are used to synchronize stakeholder expectations throughout the lifecycle:

1) Major milestones 2) Minor milestones 3) Status assessments

- The most important major milestone is usually the event that transitions the project from the elaboration phase into the construction phase.

- The format and content of minor milestones are highly dependent on the project and the organizational culture.

- Periodic status assessments are important for focusing continuous attention on the evolving health of the project and its dynamic priorities.

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

1) Major milestones: These are the system wide events are held at the end of each development phase.

They provide visibility to system wide issues.

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

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

- An iteration represents a cycle of activities. Each of the lifecycle phases undergoes one or more iterations.

Minor milestones capture two artifacts: a release specification and a release description. Major milestones at the end of each phase use formal, stakeholderapprovedevaluation criteria and release descriptions; minor milestones use informal, development-team-controlled versions of these artifacts.

- Most projects should establish all four major milestones. Only in exceptional case you add other major milestones or operate with fewer. For simpler projects, very few or no minor milestones may be necessary to manage intermediate results, and the number of status assessments may be infrequent.

MAJOR MILESTONES:

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 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. Architects and systems engineers:product line compatibility, requirements changes, tradeoff analyses, completeness

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: regulatory agencies, independent verification and validation contractors, venture capital investors, subcontractors, associate contractors, and sales andmarketing teams.

Life-Cycle Objective Milestone: These milestones occur 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.

Life- Cycle Architecture Milestone: These milestones occur at the end of the elaboration phase. Primary goal is to demonstrate an executable architecture to all stakeholders. A more detailed plan for the construction phase is presented for approval. Critical issues relative to requirements and the operational concept are addressed.

Initial Operational Capability Milestone: These milestones occur late in theconstruction 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.

Product Release Milestone: Occur 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 and software quality metrics are reviewed to determine whether quality is sufficient for transition to the support organization.

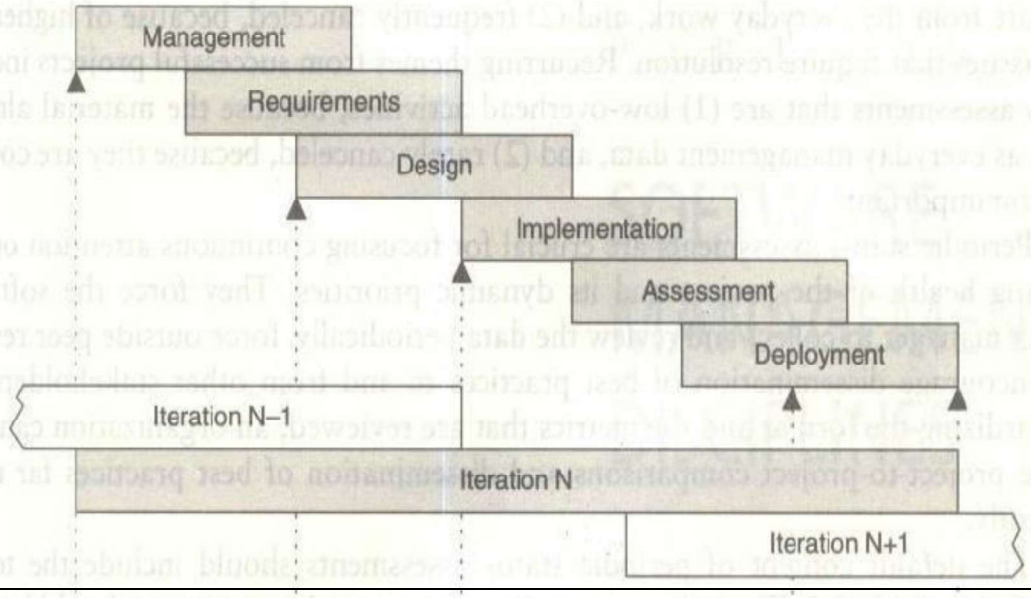
MINOR MILESTONES:

The number of iteration-specific, informal milestones needed depends on the content and length of the iteration. Iterations which have one- month to six-month duration have only two milestones are needed: the iteration readiness review and iteration assessment review. For longer iterations some other intermediate review points are added. All iterations are not created equal. Iteration takes different forms

and priorities, depending on where the project is in the life cycle. Early iterations focus on analysis and design. Later iterations focus 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 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 to review iteration results, test results, to determine amount of rework to be done, to review impact of the iteration results on the plan for subsequent iterations.



PERIODIC STATUS ASSESSMENTS:

- These are management reviews conducted at regular intervals (monthly, quarterly) to address progress and quality of project and maintain open communication among all stakeholders.

The main objective of these assessments is to synchronize all stakeholders expectations and also serve as project snapshots. Also provide,

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

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

3) Objective data derived directly from on-going activities and evolving product configurations.

Iterative Process Planning:

- Like software development, project planning is also an iterative process.

- Like software, plan is also an intangible one. Plans have an engineering stage,during which the plan is developed, and a production stage, where the plan is executed.

Part-II Process Planning

Work breakdown structures

- Work breakdown structure is the "architecture" of the project plan and also an architecture for financial plan.

- A project is said to be in success, if we maintain good work breakdown structure and its synchronization with the process frame work.

- A WBS is simply a hierarchy of elements that decomposes the project plan into discrete work tasks and it provides:

1) A pictorial description of all significant work.

2) A clear task decomposition for assignment of responsibilities.

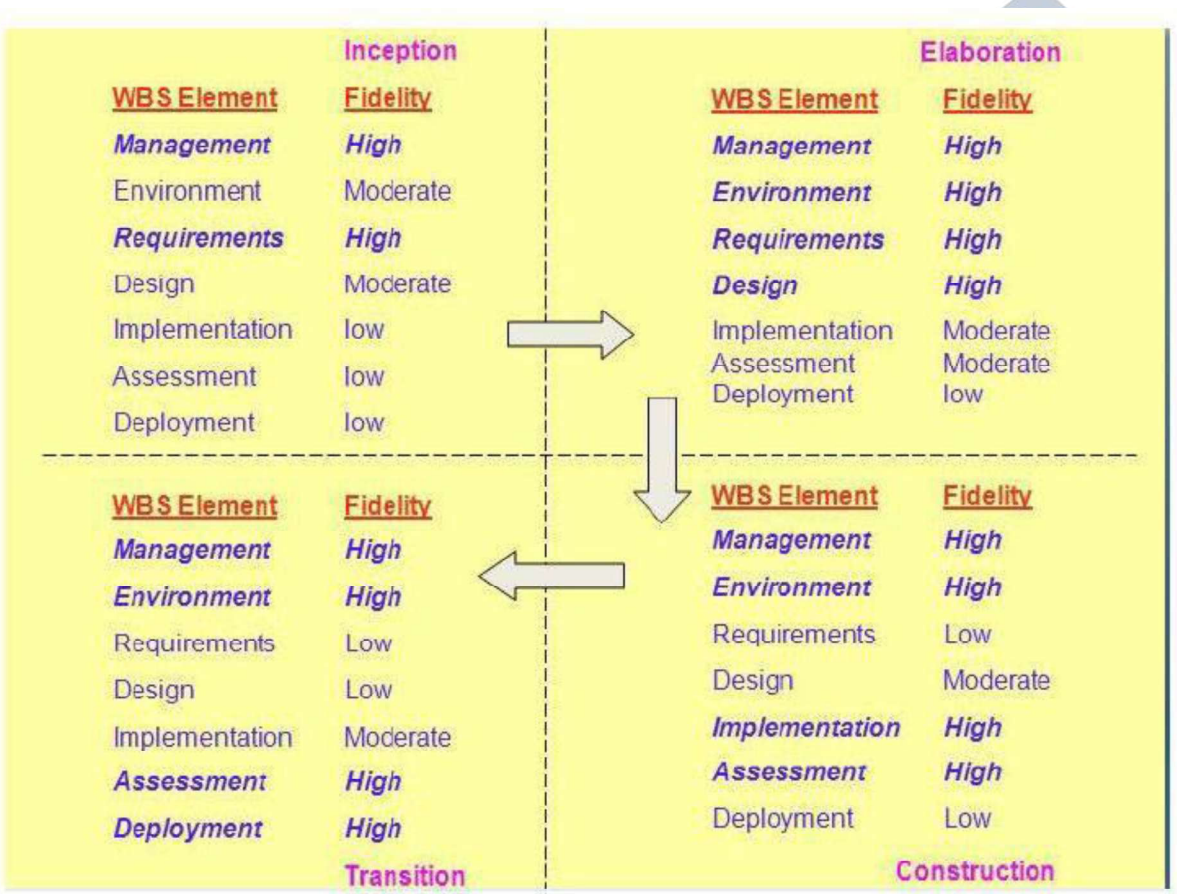
3) A framework for scheduling, budgeting, and expenditure tracking.

1) First-level elements: WBS elements are the workflows and are allocated to single team; provide the structure for the purpose of planning and comparison with the other projects.

2) Second-level elements: elements are defined for each phase of the life cycle.These elements allow the faithfulness of the plan to evolve more naturally with the level of understanding of the requirements and architecture, and the risks therein.

3) Third-level elements:

- These elements are defined for the focus of activities that produce the artifacts of each phase.
- These elements may be the lowest level in the hierarchy that collects the cost of discrete artifacts for a given phase, or they may be decomposed further into several lower level activities that, taken together, produce a single artifact.



PLANNING GUIDELINES:

- Software projects span a broad range of application domains.
- It is valuable but risky to make specific planning suggestions independent of project context.

- Planning provides a skeleton of the project from which the management people

can decide the starting point of the project.

- In order to proper plan it is necessary to capture the planning guidelines from most expertise and experience people.

- Project-independent planning advice is also risky. Adopting the planning guidelines blindly without being adapted to specific project circumstances is risk.

The above table provides default allocation for budgeted costs of each first-level WBS element.

- Sometimes these values may vary across projects but this allocation provides a good benchmark for assessing the plan by understanding the foundation for deviations from these guidelines.

- It is cost allocation table not the effort allocation.

The cost and schedule estimating process

Project plans need to be derived from two perspectives:

1) Forward-looking, top-down approach: It starts with an understanding 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 sequences would occur:

a) The software project manager develops a characterization of the overall size, process, environment, people, and quality required for the project.

b) A macro-level estimate of the total effort and schedule is developed using a software cost estimation model.

c) The software project manager partitions the estimate for the effort into a top level WBS using guidelines (table 10-1) and also partitions the schedule into major milestone dates and partition the effort into a staffing profile using guidelines

(table 10-2).

d) Subproject managers are given the responsibility for decomposing each of the WBS elements into lower levels using their tip-level allocation, staffing profile, and major milestone dates as constraints.

2) 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 higher level budgets and intermediate milestones. This approach tends to define the WBS from the lowest levels upward. From this perspective, the following planning sequences would occur:

a) The lowest level WBS elements are elaborated into detailed tasks. These estimates tend to incorporate the project-specific parameters in an exaggerated way.

b) Estimates are combined and integrated into higher level budgets and milestones.

c) Comparisons are made with the top-down budgets and schedule milestones.

Gross differences are assessed and adjustments are made in order to converge on agreement between the topdown and bottom-up estimates.

- These two planning approaches should be used together, in balance, throughout the life cycle of the project.

- During the engineering stage, the top-down perspective will dominate because there is usually not enough depth of understanding nor stability in the detailed task sequences to perform credible bottomup planning.

- During the production stage, there should be enough precedent experience and planning fidelity that the bottom-up planning perspective will dominate.
- By then, the top-down approach should be well tuned to the project specific parameters, so it should be used more as a global assessment technique.

The iteration planning 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 circumstances.

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 the 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 iterations: 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 load 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 iterations. 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

A very large or unprecedented project with many stakeholders may require additional inception iteration and two additional iterations in construction, for a total of nine iterations.

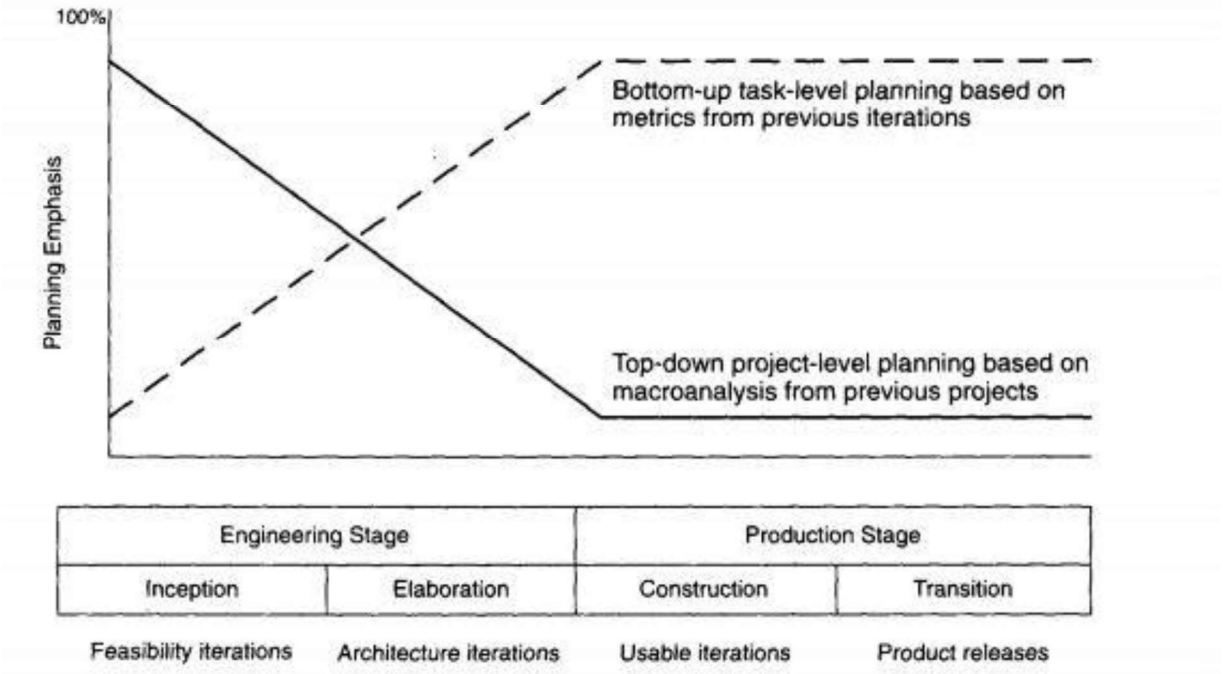


FIGURE 10-4. *Planning balance throughout the life cycle*

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 A side from bad architectures and misunderstood requirements, 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.