

## 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.

---

### INTRODUCTION TO WORKFLOW

Workflow is the definition, execution and automation of business processes where tasks, information or documents are passed from one participant to another for action, according to a set of procedural rules.

Organizations use workflows to coordinate tasks between people and synchronize data between systems, with the ultimate goal of improving organizational efficiency, responsiveness and profitability.

A fully functional digital workflow provides four key benefits to a company:

- ✓ Improved productivity
- ✓ Process transparency
- ✓ Faster business reaction time
- ✓ Improved accountability

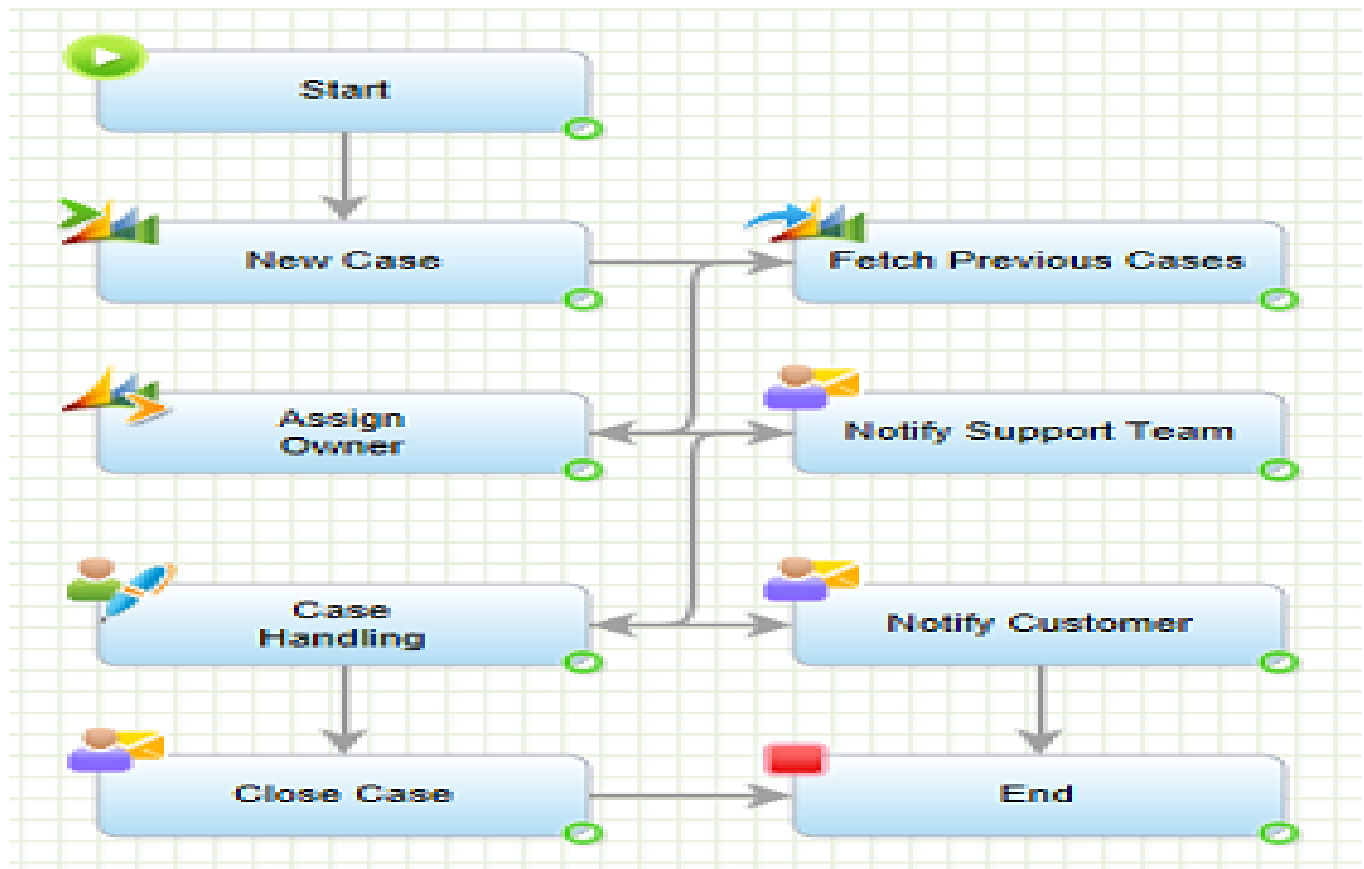


Figure 3-1: Workflow Process

There are seven top-level of software process 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

### **Checkpoints of the process**

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.

### **Software Process Workflows**

Previous chapters introduced a life-cycle macro process and the fundamental sets of artifacts. The macro process 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 micro processes, 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.

### **Iteration Workflows**

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

As with any sequence of a software development workflow, many of the activities occur concurrently. For example, requirements analysis is not done all in one continuous lump; it intermingles with management, design, implementation, and so forth.

Iterations in the inception and elaboration phases focus on management, requirements, and design activities. Iterations in the construction phase focus on design, implementation, and assessment. Iterations in the transition phase focus on assessment and deployment. Figure 8-3 shows the emphasis on different activities across the life cycle.

These descriptions are pretty simplistic. In practice, the various sequences and overlaps among iterations become more complex. The terms iteration and increment deal with some of the pragmatic considerations. 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, 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.

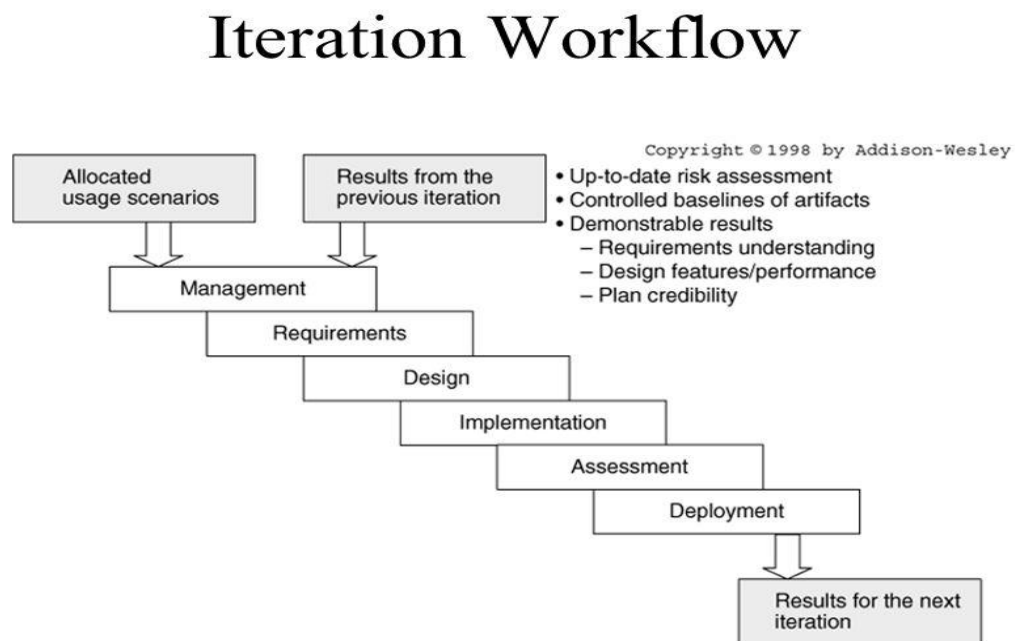


Figure: 3-2: Iteration Workflow

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 undergo 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, stakeholder-approved evaluation 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.

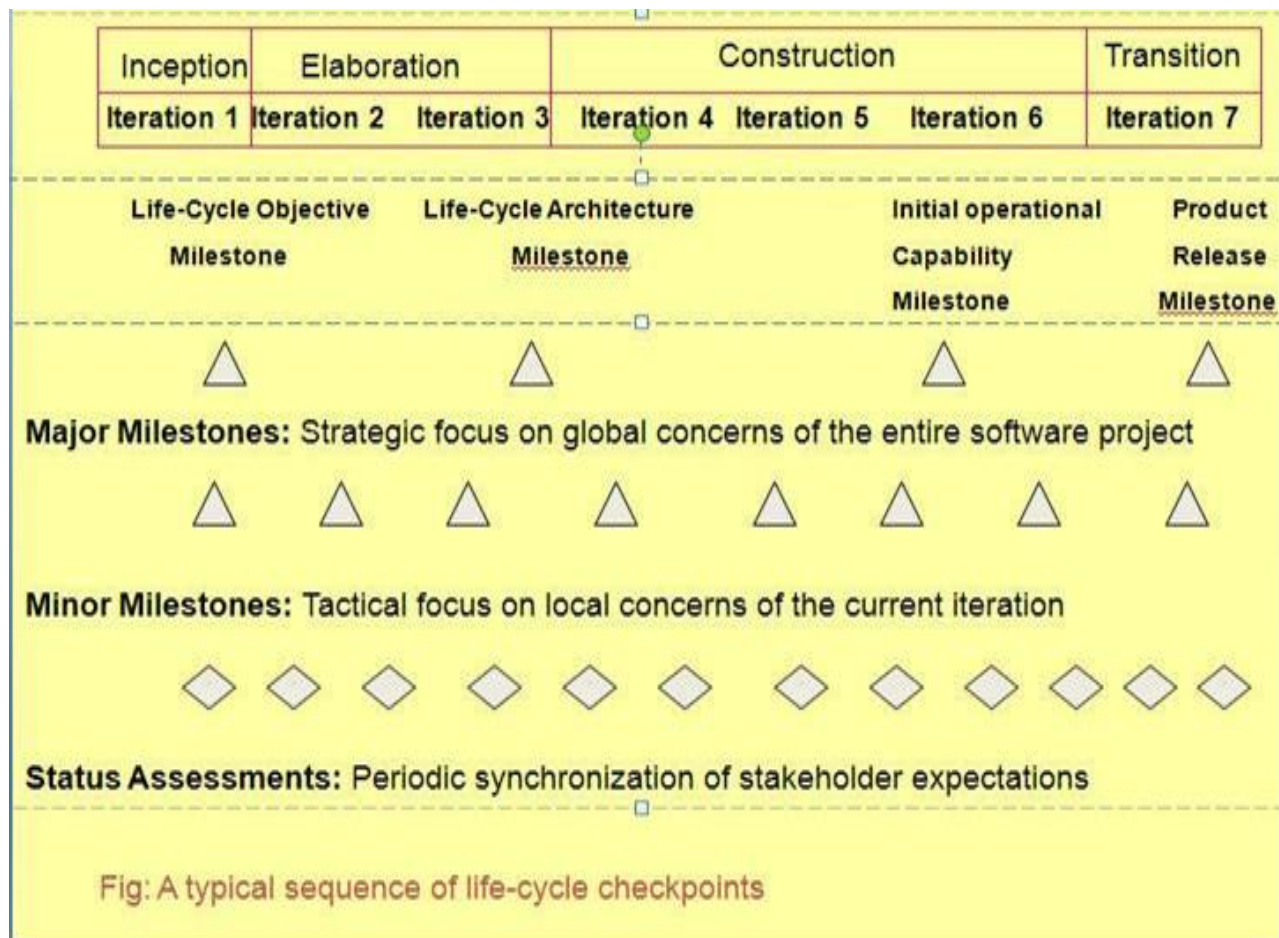


Figure 3-3: Sequence of life-cycle checkpoints

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



**Table 3-1. 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                                  | Baseline vision, including growth vectors, quality attributes, and priorities | Demonstration of at least one feasible architecture |
|  | Low-fidelity life-cycle plan  | Use case model  | Make/buy/reuse trade-offs                           |
|  | High-fidelity elaboration phase plan  |   | Initial design model                                |
| Life-cycle architecture milestone        | High-fidelity construction phase plan (bill of materials, labor allocation) | Stable vision and use case model  | Stable design set                                   |
|  | Low-fidelity transition phase plan  | Evaluation criteria for construction releases, initial operational capability | Make/buy/reuse decisions                            |
|  |   | Draft user manual   | Critical component prototypes                       |
| Initial operational capability milestone | High-fidelity transition phase plan   | Acceptance criteria for product release                                       | Stable implementation set                           |
|  |   | Releasable user manual  | 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                                   |

|   |
|---|
| <b>I. Requirements</b>  |
| A. Use case model   |
| B. Vision document (text, use cases)  |
| C. Evaluation criteria for elaboration (text, scenarios)                      |
| <b>II. Architecture</b>   |
| 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  |
| <b>III. Source and executable libraries</b>                                   |
| A. Product components   |
| B. Test components  |
| C. Environment and tool components  |

**Figure 3-4: Engineering artifacts available at the life-cycle architecture milestone**

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

**Figure 3-5. Default agendas for the life-cycle architecture milestone**

**Initial Operational Capability Milestone:** These milestones occur 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.

- **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. An iteration take 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.

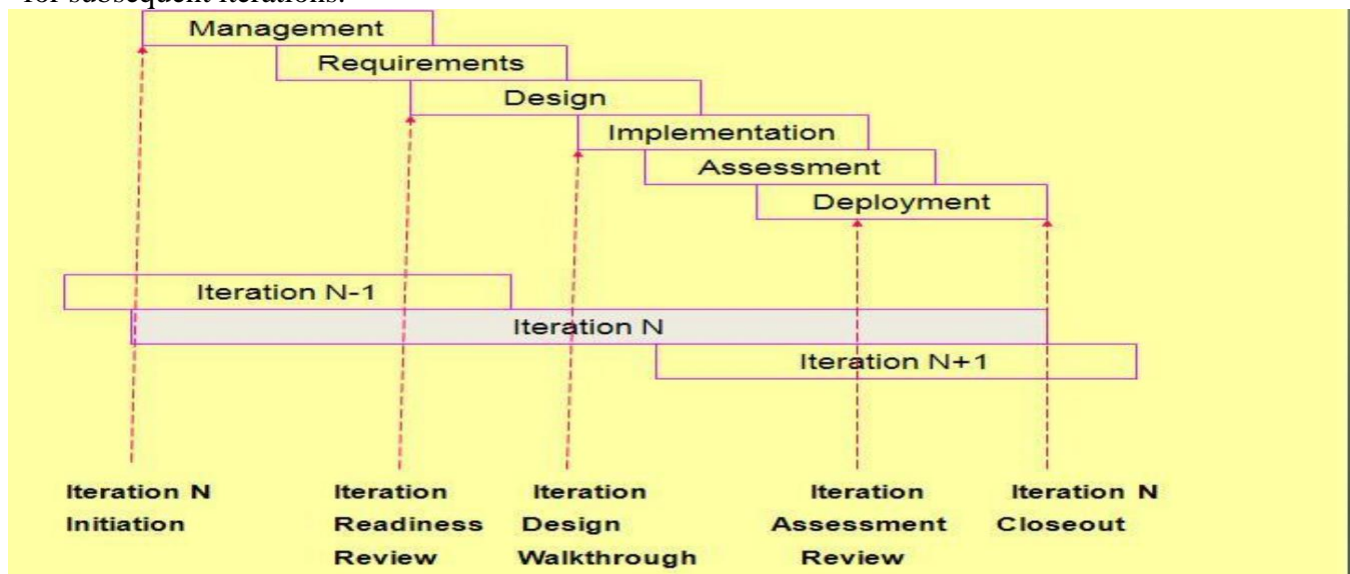


Figure 3-6: Typical minor milestones in the life cycle of an iteration

### 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 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 mechanism for openly addressing, communicating, and resolving management issues, technical issues, and project risks
- Objective data derived 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

Periodic status assessments are crucial for focusing continuous attention on the evolving health of the project and its dynamic priorities. They force the software project manager to collect and review the data periodically, force outside peer review, and encourage dissemination of best practices to and from other stakeholders.

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

**Table 3-2: Default content of status assessment reviews**

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

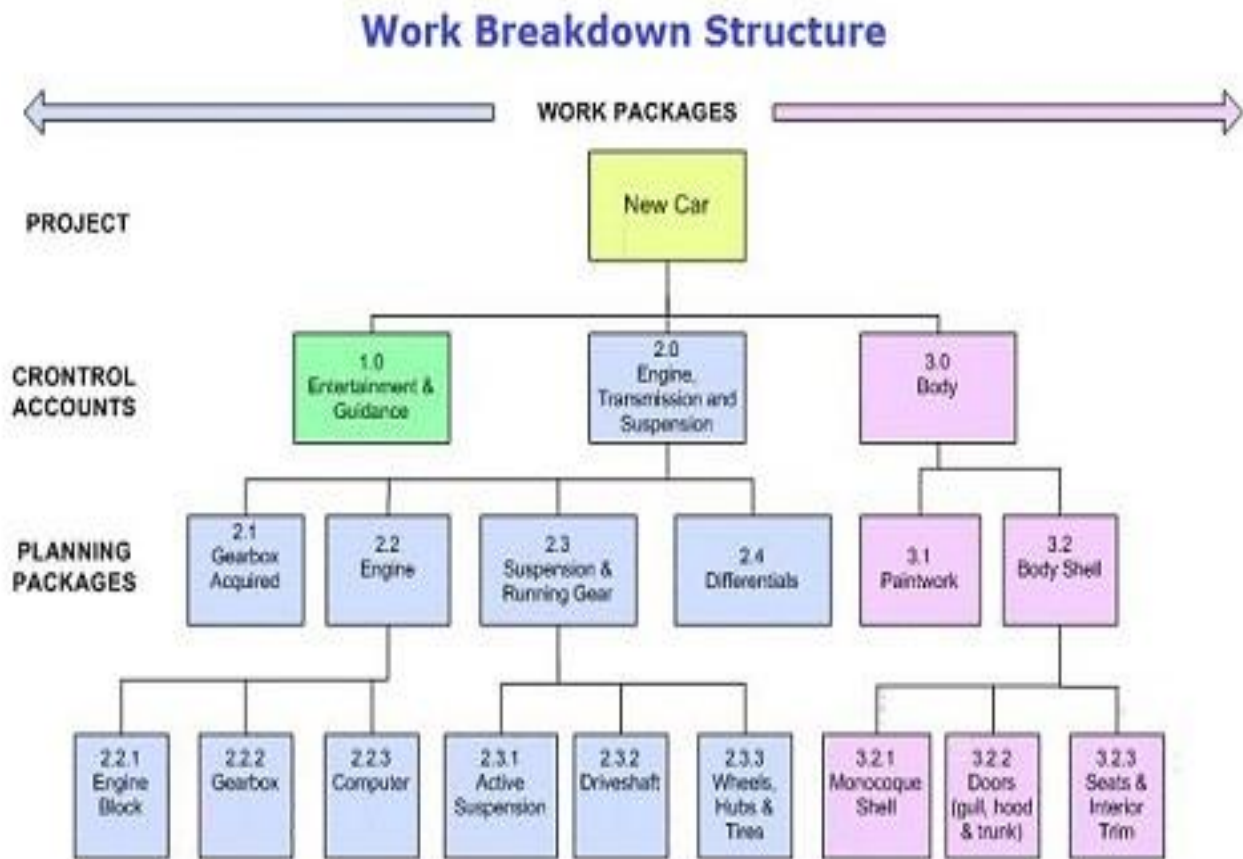
### **Work Breakdown Structures**

A good work breakdown structure and its synchronization with the process framework are critical factors in software project success. Development of a work breakdown structure dependent on the project management style, organizational culture, customer preference, financial constraints, and several other hard-to-define, project-specific parameters.

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

Many parameters can drive the decomposition of work into discrete tasks: product subsystems, components, functions, organizational units, life-cycle phases, even geographies. Most systems have a first-level decomposition by subsystem. Subsystems are then decomposed into their components, one of which is typically the software.



**Figure 3-7: Work Breakdown structure**

### Conventional WBS Issues

Conventional work breakdown structures frequently suffer from three fundamental flaws.

1. They are prematurely structured around the product design.
2. They are prematurely decomposed, planned, and budgeted in either too much or too little detail.
3. They are project-specific, and cross-project comparisons are usually difficult or impossible.

Conventional work breakdown structures are prematurely structured around the product design. Figure shows a typical conventional WBS that has been structured primarily around the subsystems of its product architecture, then further decomposed into the components of each subsystem. A WBS is the architecture for the financial plan.

Conventional work breakdown structures are prematurely decomposed, planned, and budgeted in either too little or too much detail. Large software projects tend to be over planned and small projects tend to be under planned. The basic problem with planning too much detail at the outset is that the detail does not evolve with the level of fidelity in the plan.

Conventional work breakdown structures are project-specific, and cross-project comparisons are usually difficult or impossible. With no standard WBS structure, it is extremely difficult to compare plans, financial data, schedule data, organizational efficiencies, cost trends, productivity trends, or quality trends across multiple projects.

## **Conventional work breakdown structure, following the product hierarchy**

### **Management**

**System requirement and design**

#### **Subsystem 1**

##### **Component 11**

**Requirements**

**Design Code Test**

**Documentation**

**...(similar structures for other components)**

#### **Component 1N Requirements Design**

**Code Test Documentation**

**...(similar structures for other subsystems)**

#### **Subsystem M Component M1**

**Requirements**

**Design Code Test**

**Documentation**

**...(similar structures for other components)**

#### **Component MN**

**Requirements**

**Design Code Test**

**Documentation**

### **Integration and test**

**Test planning**

**Test procedure preparation**

**Testing**

**Test reports**

### **Other support areas**

**Configuration control**

**Quality assurance**

**System administration**

## **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, requirements, design, implementation, assessment, and deployment).
- Second-level elements are defined for each phase of the life cycle (inception, elaboration, construction, and transition).
- Third-level elements are defined for the focus of activities that produce the artifacts of each phase.

A default WBS consistent with the process framework (phases, workflows, and artifacts) is shown in Figure. This recommended structure provides one example of how the elements of the process

framework can be integrated into a plan. It provides a framework for estimating the costs and schedules of each element, allocating them across a project organization, and tracking expenditures.

The structure shown is intended to be merely a starting point. It needs to be tailored to the specifics of a project in many ways.

- **Scale.** Larger projects will have more levels and substructures.
- **Organizational structure.** Projects that include subcontractors or span multiple organizational entities may introduce constraints that necessitate different WBS allocations.
- **Degree of custom development.** Depending on the character of the project, there can be very different emphases in the requirements, design, and implementation workflows.
- **Business context.** Projects developing commercial products for delivery to a broad customer base may require much more elaborate substructures for the deployment element.
- **Precedent experience.** Very few projects start with a clean slate. Most of them are developed as new generations of a legacy system (with a mature WBS) or in the context of existing organizational standards (with preordained WBS expectations).

The WBS decomposes the character of the project and maps it to the life cycle, the budget, and the personnel. Reviewing a WBS provides insight into the important attributes, priorities, and structure of the project plan.

Another important attribute of a good WBS is that the planning fidelity inherent in each element is commensurate with the current life-cycle phase and project state. Figure 10-3 illustrates this idea. One of the primary reasons for organizing the default WBS the way I have is to allow for planning elements that range from planning packages (rough budgets that are maintained as an estimate for future elaboration rather than being decomposed into detail) through fully planned activity networks (with a well-defined budget and continuous assessment of actual versus planned expenditures).

### **Default work breakdown structure**

#### **A Management**

##### **AA Inception phase management**

**AAA Business case development**

**AAB Elaboration phase release specifications**

**AAC Elaboration phase WBS specifications**

**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
    - CCA 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
- D Design
  - DA Inception phase architecture prototyping
  - DB Elaboration phase architecture baselining
    - DBA Architecture design modeling
    - DBB Design demonstration planning and conduct
    - DBC Software architecture description
  - DC Construction phase design modeling
    - DCA Architecture design model maintenance
    - DCB Component design modeling
  - DD Transition phase design maintenance
- E Implementation
  - EA Inception phase component prototyping
  - EB Elaboration phase component implementation
    - EBA Critical component coding demonstration integration
  - EC Construction phase component implementation
    - ECA Initial release(s) component coding and stand-alone testing
    - ECB Alpha release component coding and stand-alone testing
    - ECC Beta release component coding and stand-alone testing
    - ECD Component maintenance
- F Assessment
  - FA Inception phase assessment
  - FB Elaboration phase assessment
    - FBA Test modeling
    - FBB Architecture test scenario implementation

- FBC Demonstration assessment and release descriptions
- FC Construction phase assessment
  - FCA Initial release assessment and release description
  - FCB Alpha release assessment and release description
  - FCC Beta release assessment and release description
- FD Transition phase assessment
  - FDA Product release assessment and release description
- G Deployment
  - GA Inception phase deployment planning
  - GB Elaboration phase deployment planning
  - GC Construction phase deployment
    - GCA User manual baselining
  - GD Transition phase deployment
    - GDA Product transition to user

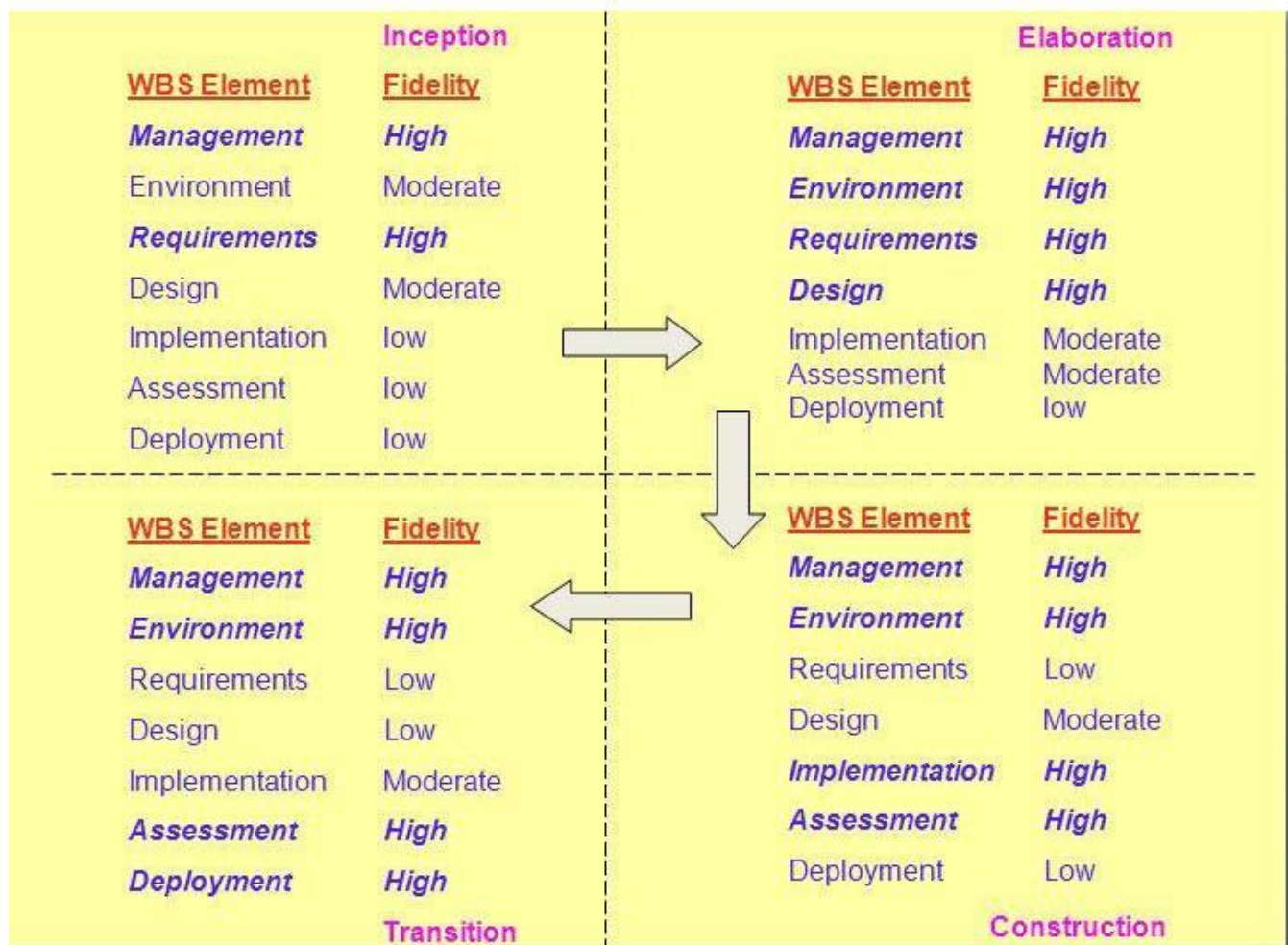


Figure 3-8: Evolution of planning fidelity in the WBS over the life cycle

## 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 circumstances. Two simple planning guidelines should be considered when a project plan is being initiated or assessed. The first guideline, detailed in Table, prescribes a default allocation of costs among the first-level WBS elements. The second guideline, detailed in Table, prescribes the 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%           |

### 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:

1. The software project manager (and others) develops a characterization of the overall size, process, environment, people, and quality required for the project.
2. A macro-level estimate of the total effort and schedule is developed using a software cost estimation model.
3. The software project manager partitions the estimate for the effort into a top-level WBS using guidelines such as those in Table 10-1.
4. 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, and 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:

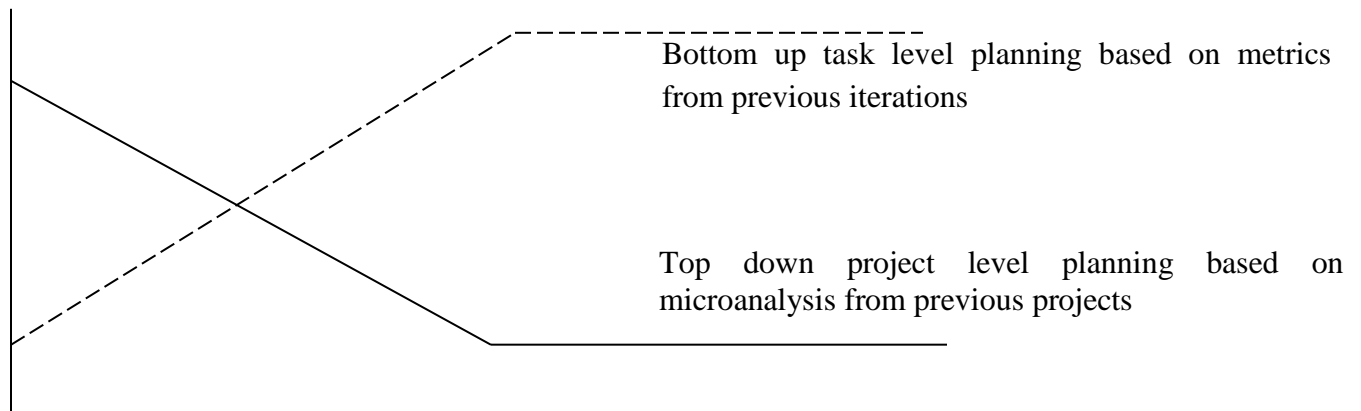
1. The lowest level WBS elements are elaborated into detailed tasks
2. Estimates are combined and integrated into higher level budgets and milestones.
3. Comparisons are made with the top-down budgets and schedule milestones.

Milestone scheduling or budget allocation through top-down estimating tends to exaggerate the project management biases and usually results in an overly optimistic plan. Bottom-up estimates usually exaggerate the performer biases and result in an overly pessimistic plan.

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 bottom-up planning.

During the production stage, there should be enough precedent experience and planning fidelity that the bottom-up planning perspective will dominate. Top-down approach should be well tuned to the project-specific parameters, so it should be used more as a global assessment technique. Figure illustrates this life-cycle planning balance.

#### **Planning balance throughout the life cycle**



| Engineering Stage |             | Production Stage |            |
|-------------------|-------------|------------------|------------|
| Inception         | Elaboration | Construction     | Transition |

Feasibility iteration    Architecture iteration    Usable iteration    Product Releases

| Engineering Stage Planning Emphasis   | Production Stage Planning Emphasis                                   |
|---|--|
| Macro level task estimation for production stage artifacts  | Micro level task estimation for production stage artifacts           |
| Micro level task estimation for engineering artifacts   | Macro level task estimation for maintenance of engineering artifacts |
| Stakeholder concurrence   | Stakeholder concurrence  |
| Coarse grained variance analysis of actual vs planned expenditures                                    | Fine 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  |  |

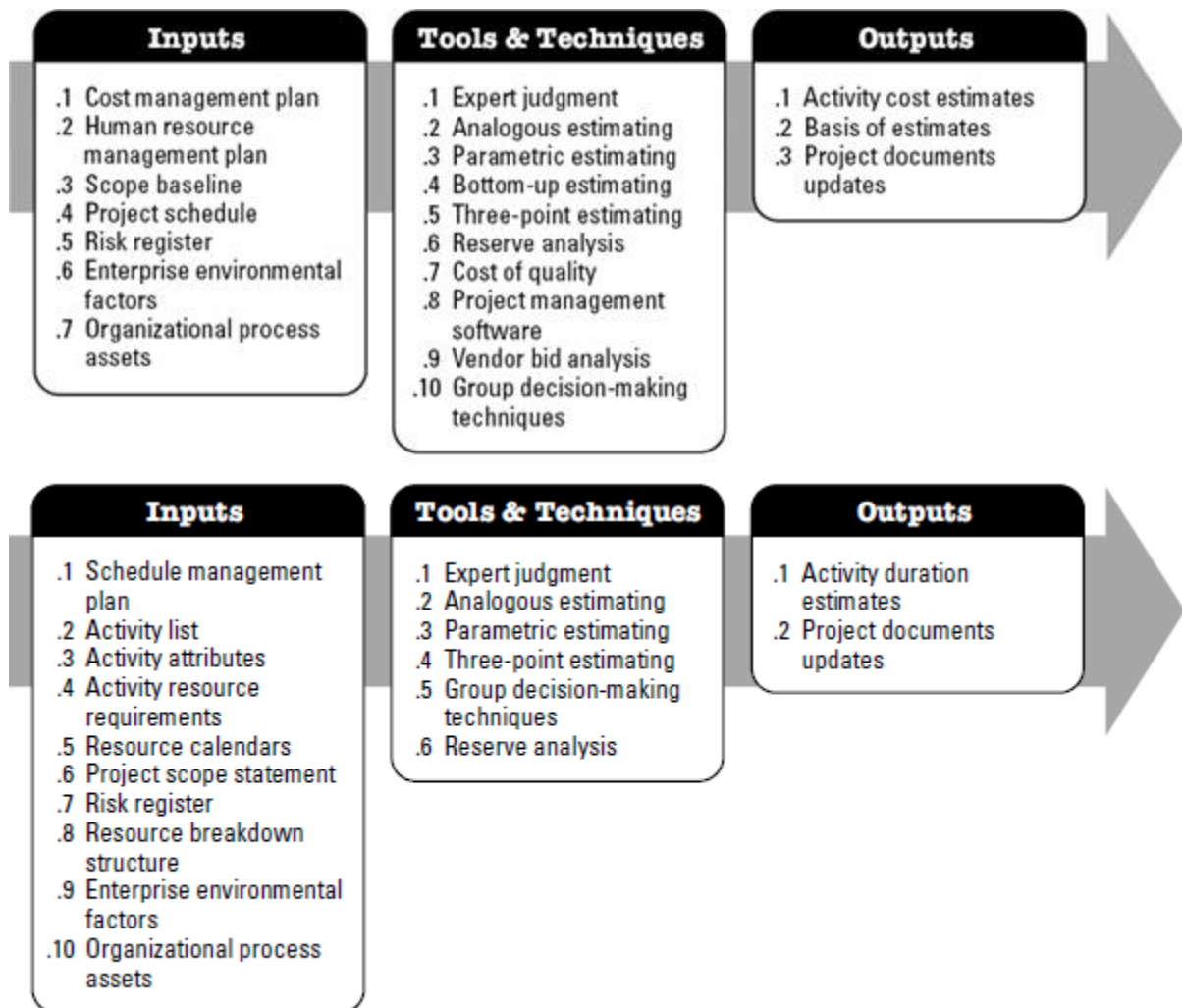


Figure 3-9: Cost & Schedule Estimate process

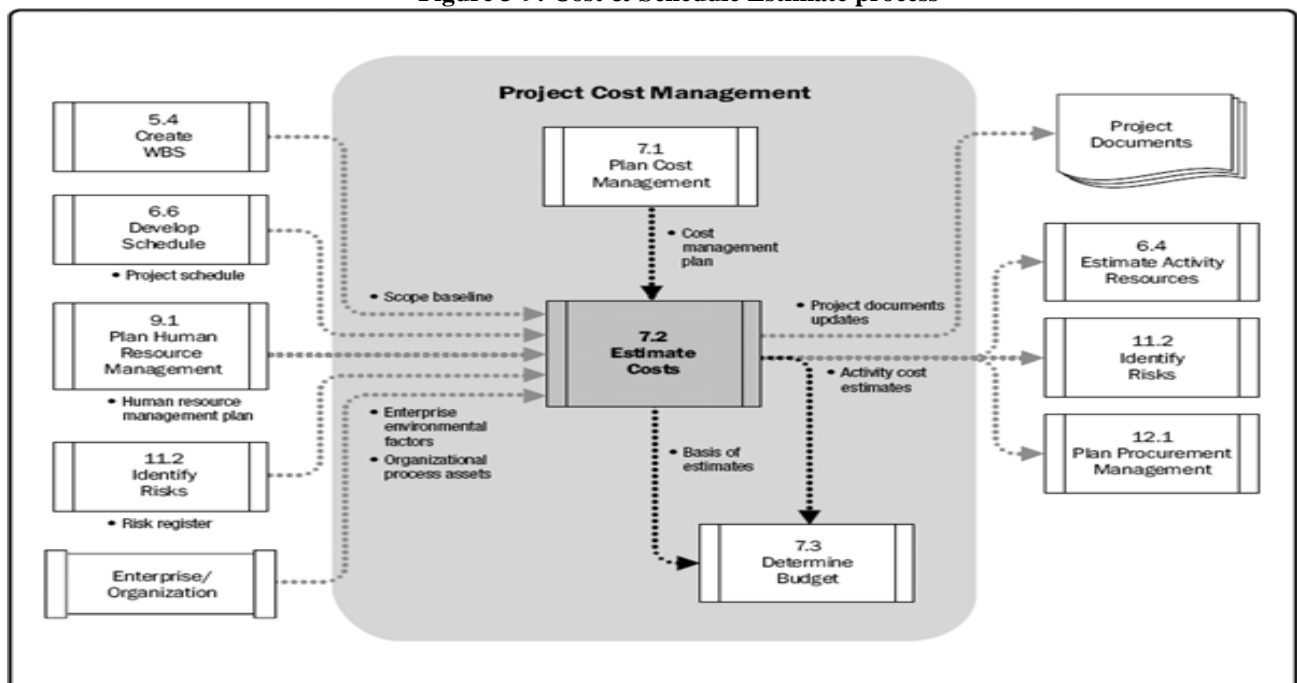


Figure 3-10: Estimate Cost Data Flow Diagram



## **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 a 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 a 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.

## **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 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.

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 stakeholders, and it must be used. Plans are not just for managers. 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.