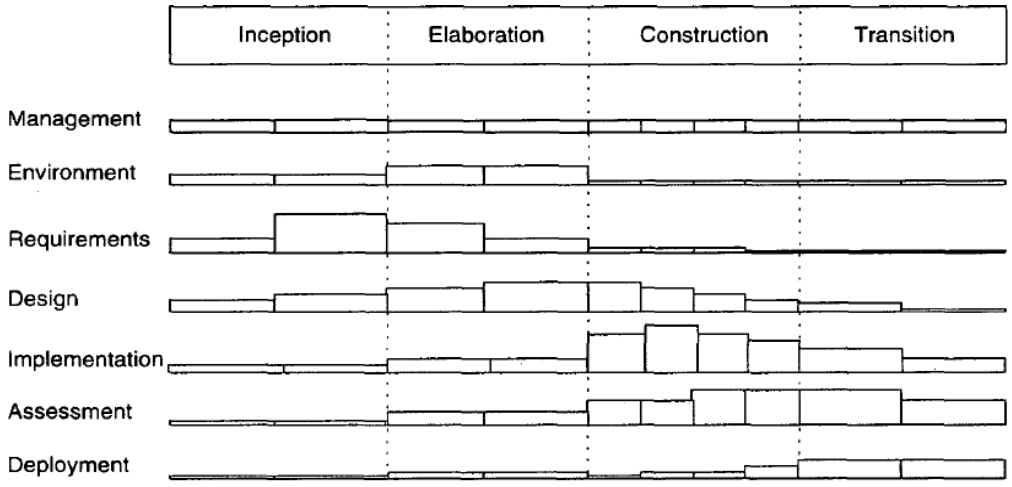
SPM UNIT III

Work Flows of the Process: Software Process Workflows. Inter Trans Workflows.
Checkpoints of the Process: Major Mile Stones, Minor Milestones, Periodic Status Assessments.
Interactive Process Planning: Work Breakdown Structures, Planning Guidelines, Cost and Schedule Estimating. Interaction Planning Process. Pragmatic Planning.

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

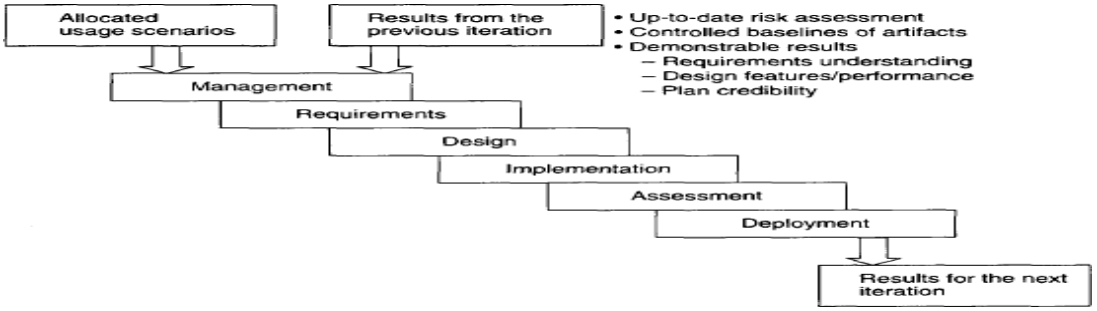
1. Architecture –first approach: Extensive requirements analysis, design, implementation and assessment activities are performed before the construction phase when full-scale implementation is the focus.
2. Iterative life-cycle process: Some projects may require only one iteration in a phase, others 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

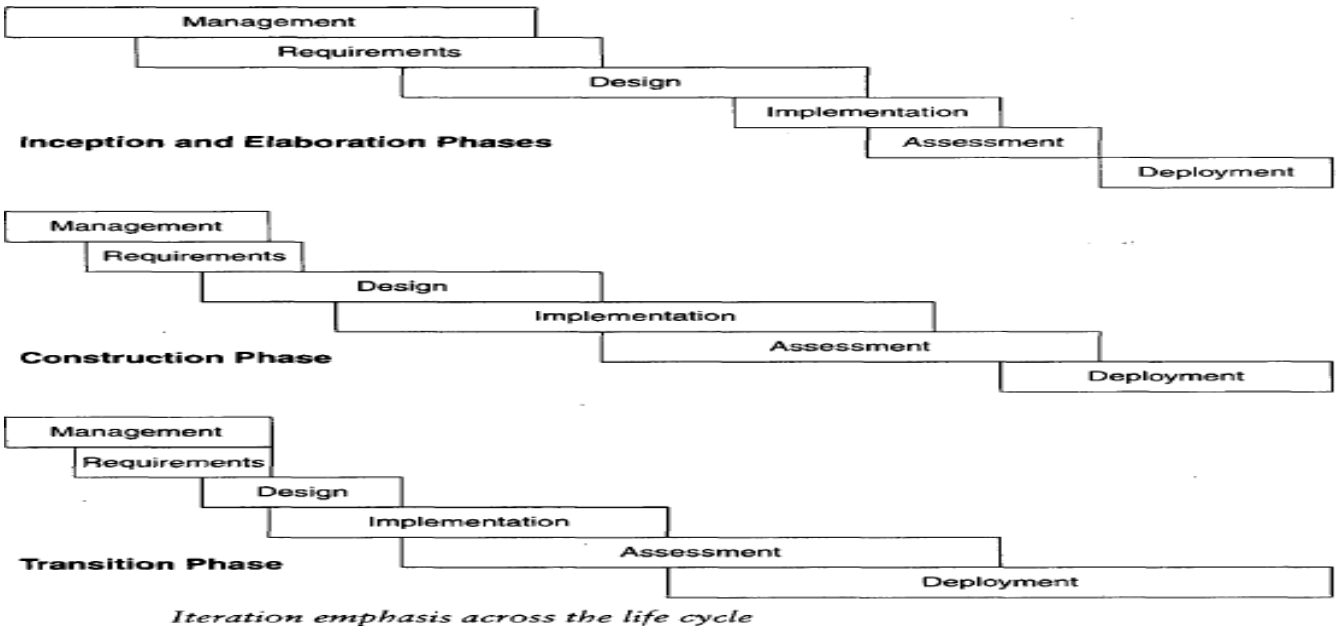
Iteration consists of a loosely sequential set of activities in various proportions, depending on where the iteration is located in the development cycle. Each iteration is defined in terms of a set of allocated usage scenarios.



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:

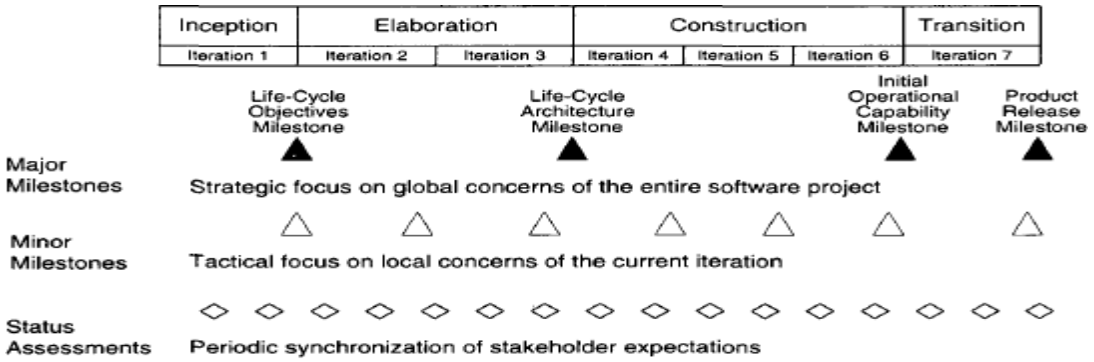
- **Major Milestones**: these system wide events are held ant the end of each development phase. They provide visibility to system wide issues synchronize the management and engineering perspectives and verify that the aims of the phase have been achieved.
- **Minor Milestones**: theses iteration-focused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

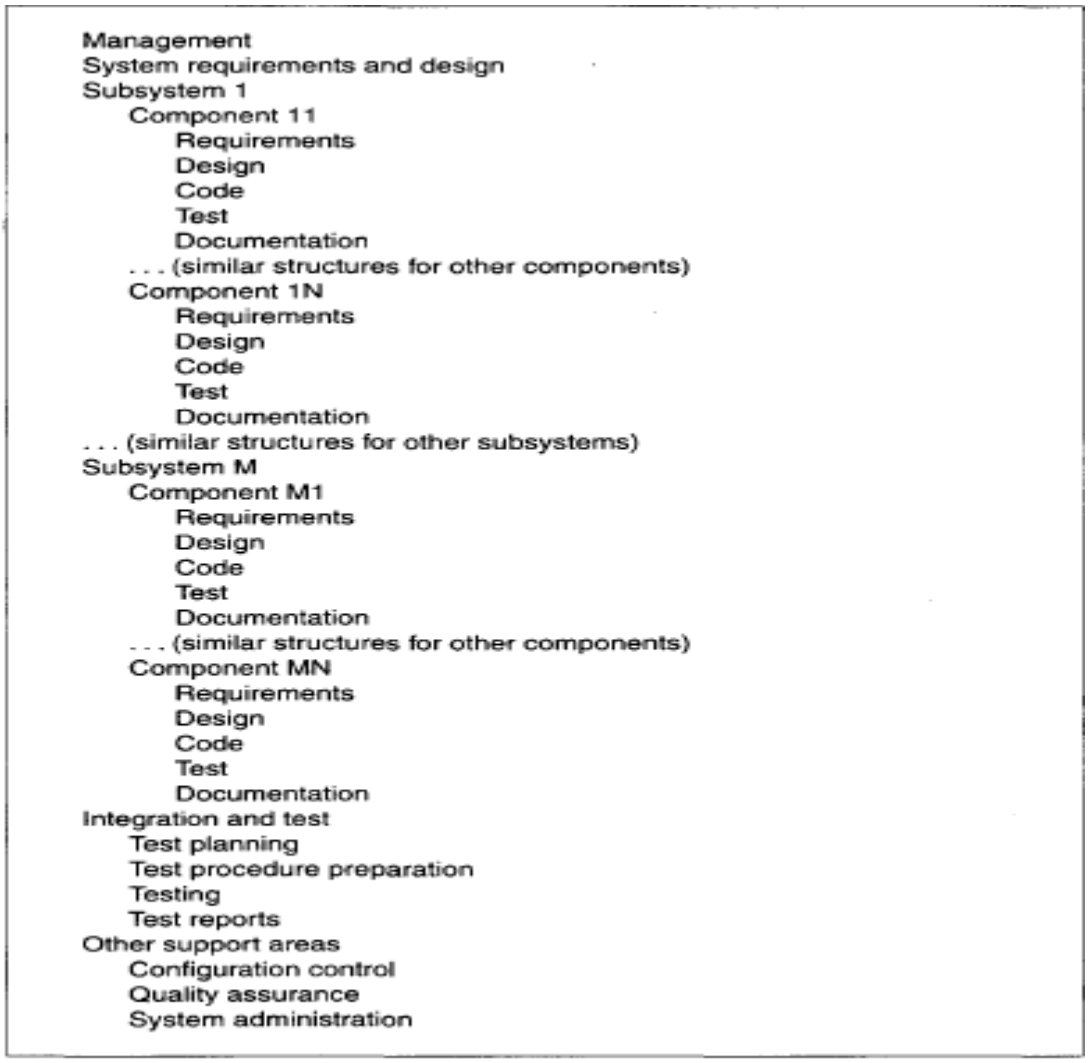
WORK BREAKDOWN STRUCTURES (WBS)

- 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) A delineation of all significant work
  b) A clear task decomposition for assignment of responsibilities
  c) A framework for scheduling, budgeting, and expenditure tracking

CONVENTIONAL WBS ISSUES

Conventional work breakdown structures frequently suffer from three fundamental flaws.

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



Conventional work breakdown structure, following the product hierarchy

- *Conventional work breakdown structures are prematurely structured around the product design.* The above Figure shows a typical conventional WBS that has been structured primarily around the subsystems of its product architecture, and then further decomposed into the components of each subsystems. 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.

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:

a) First-level WBS elements are the workflows (management, environment, requirements, design, implementation, assessment, and deployment).

b) Second-level elements are defined for each phase of life cycle (inception, elaboration, construction, and transition).

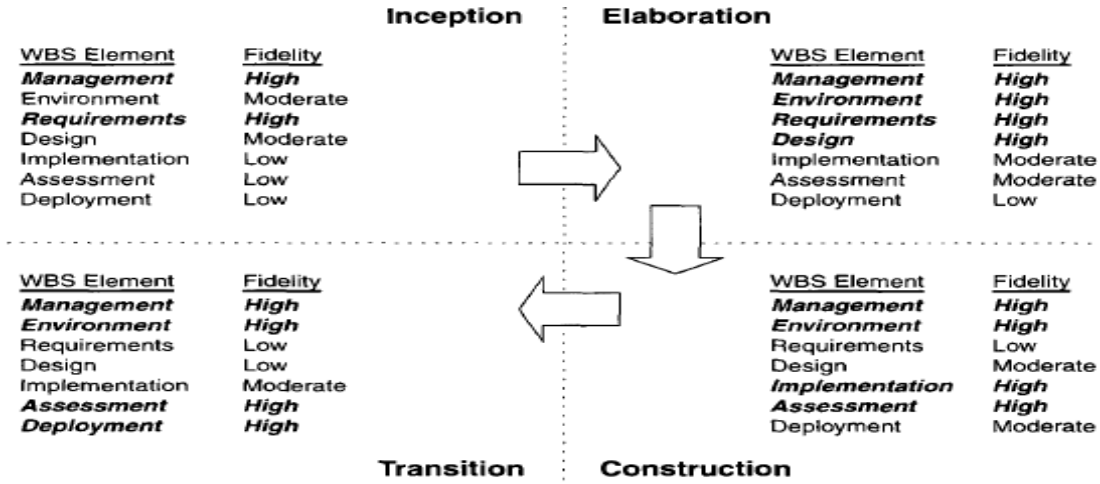
c) Third-level elements are defined for the focus of activities that produce the artifacts of each phase.

A default WBS consistent with the process framework (phases, workflows, and artifacts) is shown in Figure:

- 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
- 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
  - ED Transition phase component maintenance
- F Assessment
  - FA Inception phase assessment planning
  - 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 descriptions
- 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

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.



Evolution of planning fidelity in the WBS over the life cycle

The below table prescribes a default allocation of costs among the first-level WBS elements.

WBS budgeting defaults

FIRST-LEVEL WBS ELEMENT	DEFAULT BUDGET
Management	10%
Environment	10%
Requirements	10%
Design	15%
Implementation	25%
Assessment	25%
Deployment	5%
Total	100%

The below table prescribes allocation of effort and schedule across the lifecycle phases.

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.
- A macro-level estimate of the total effort and schedule is developed using a software cost estimation model.
- The software project manager partitions the estimate for the effort into top-level WBS using guidelines.

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.

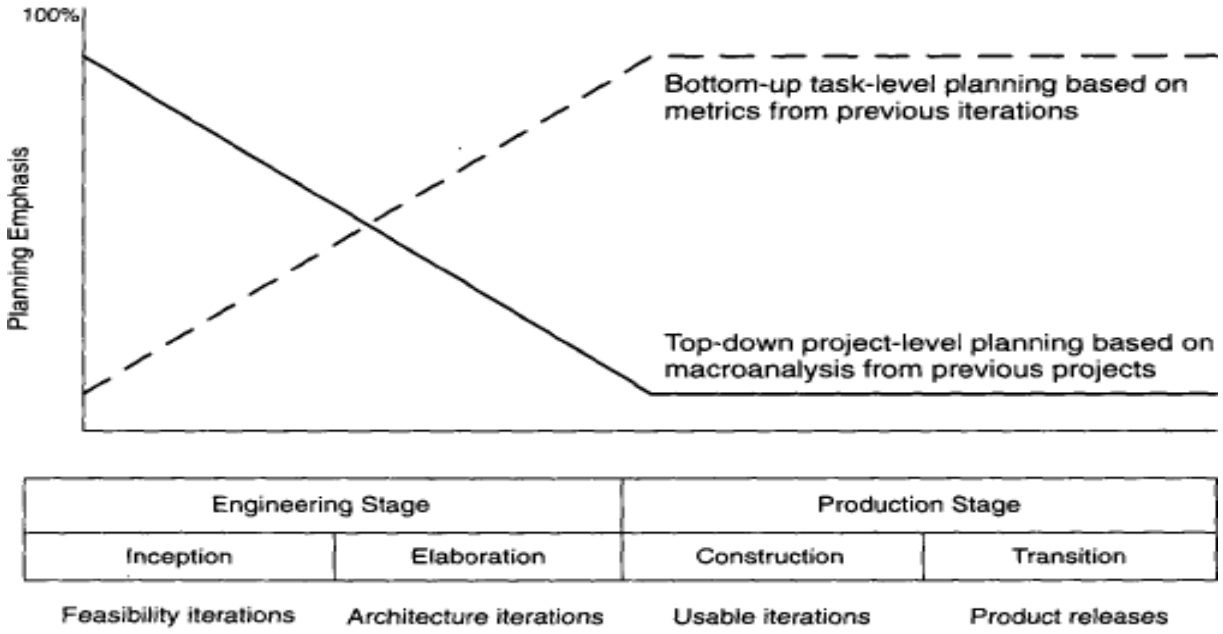
The second perspective is a ***backward-looking***, bottom-up approach. We start with the end in mind, analyze the micro-level budgets and schedules, and then sum all these elements into the higher level budgets and intermediate milestones. This approach tends to define and populate the WBS from the lowest levels upward. From this perspective, the following planning sequence would occur:

a) The lowest level WBS elements are elaborated into detailed tasks

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

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

During the engineering stage top down approach dominates bottom up approach. During the production stage bottom approach dominates top down approach.



Engineering 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

Production stage planning emphasis:

- 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

Planning balance throughout the life cycle

THE ITERATION PLANNING PROCESS

- Planning is concerned with defining the actual sequence of intermediate results.
- 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.
  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 (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.