UNIT - IV

Iterative Process Planning: Work breakdown structures, planning guidelines, cost and schedule estimating, Iteration planning process, Pragmatic planning.


10. Iterative process planning

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.

10.1.1 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 10-1 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.

**Figure 10-1 Conventional work breakdown structure, following the product hierarchy**

Management
System requirement and design
Subsystem 1
   Component 1
      Requirements
      Design
      Code
      Test
      Documentation
   ...(similar structures for other components)
Component 1N
   Requirements
   Design
   Code
   Test
10.1.2 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 10-2. 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).

**Figure 10-2 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 10-3 Evolution of planning fidelity in the WBS over the life cycle**

<table>
<thead>
<tr>
<th>WBS Element</th>
<th>Fidelity</th>
<th>WBS Element</th>
<th>Fidelity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>High</td>
<td>Management</td>
<td>High</td>
</tr>
<tr>
<td>Environment</td>
<td>Moderate</td>
<td>Environment</td>
<td>High</td>
</tr>
<tr>
<td>Requirement</td>
<td>High</td>
<td>Requirement</td>
<td>High</td>
</tr>
<tr>
<td>Design</td>
<td>Moderate</td>
<td>Design</td>
<td>High</td>
</tr>
<tr>
<td>Implementation</td>
<td>Low</td>
<td>Implementation</td>
<td>Moderate</td>
</tr>
<tr>
<td>Assessment</td>
<td>Low</td>
<td>Assessment</td>
<td>Moderate</td>
</tr>
<tr>
<td>Deployment</td>
<td>Low</td>
<td>Deployment</td>
<td>Low</td>
</tr>
</tbody>
</table>
WBS Element | Fidelity | WBS Element | Fidelity
---|---|---|---
Management | High | Management | High
Environment | High | Environment | High
Requirements | Low | Requirements | Low
Design | Low | Design | Moderate
Implementation | Moderate | Implementation | High
Assessment | High | Assessment | High
Deployment | High | Deployment | Moderate

**Transition**

**Construction**

10.2 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 10-1, prescribes a default allocation of costs among the first-level WBS elements. The second guideline, detailed in Table 10-2, prescribes the allocation of effort and schedule across the lifecycle phases.

---

**10-1 Web budgeting defaults**

<table>
<thead>
<tr>
<th>First Level WBS Element</th>
<th>Default Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>10%</td>
</tr>
<tr>
<td>Environment</td>
<td>10%</td>
</tr>
<tr>
<td>Requirement</td>
<td>10%</td>
</tr>
<tr>
<td>Design</td>
<td>15%</td>
</tr>
<tr>
<td>Implementation</td>
<td>25%</td>
</tr>
<tr>
<td>Assessment</td>
<td>25%</td>
</tr>
<tr>
<td>Deployment</td>
<td>5%</td>
</tr>
<tr>
<td>Total</td>
<td>100%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Domain</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
</table>

10.3 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, 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 10-4 illustrates this life-cycle planning balance.

Figure 10-4 Planning balance throughout the life cycle

<table>
<thead>
<tr>
<th>Effort</th>
<th>5%</th>
<th>20%</th>
<th>65%</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>10%</td>
<td>30%</td>
<td>50%</td>
<td>10%</td>
</tr>
</tbody>
</table>
Top down project level planning based on microanalysis from previous projects

<table>
<thead>
<tr>
<th>Engineering Stage</th>
<th>Production Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
<td>Construction</td>
</tr>
<tr>
<td>Elaboration</td>
<td>Transition</td>
</tr>
<tr>
<td>Feasibility iteration</td>
<td>Usable iteration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Engineering stage planning emphasis</th>
<th>Production stage planning emphasis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro level task estimation for production stage artifacts</td>
<td>Micro level task estimation for production stage artifacts</td>
</tr>
<tr>
<td>Micro level task estimation for engineering artifacts</td>
<td>Macro level task estimation for maintenance of engineering artifacts</td>
</tr>
<tr>
<td>Stakeholder concurrence</td>
<td>Stakeholder concurrence</td>
</tr>
<tr>
<td>Coarse grained variance analysis of actual vs planned expenditures</td>
<td>Fine grained variance analysis of actual vs planned expenditures</td>
</tr>
<tr>
<td>Tuning the top down project independent planning guidelines into project specific planning guidelines</td>
<td></td>
</tr>
<tr>
<td>WBS definition and elaboration</td>
<td></td>
</tr>
</tbody>
</table>
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.

10.5 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.

Unit – Important Questions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Define Model-Based software architecture?</td>
</tr>
<tr>
<td>2.</td>
<td>Explain various process workflows?</td>
</tr>
<tr>
<td>3.</td>
<td>Define typical sequence of life cycle checkpoints?</td>
</tr>
<tr>
<td>4.</td>
<td>Explain general status of plans, requirements and product across the major milestones.</td>
</tr>
<tr>
<td>5.</td>
<td>Explain conventional and Evolutionary work break down structures?</td>
</tr>
<tr>
<td>6.</td>
<td>Explain briefly planning balance throughout the life cycle?</td>
</tr>
</tbody>
</table>

---

Project Organizations and Responsibilities:

- **Organizations** engaged in software Line-of-Business need to support projects with the infrastructure necessary to use a common process.
- **Project** organizations need to allocate artifacts & responsibilities across project team to ensure a balance of global (architecture) & local (component) concerns.
- **The organization** must evolve with the WBS & Life cycle concerns.
- **Software lines of business & product teams have different motivation.**
- **Software lines of business** are motivated by return of investment (ROI), new business discriminators, market diversification & profitability.
- **Project teams** are motivated by the cost, Schedule & quality of specific
deliverables

1) **Line-Of-Business Organizations:**

   The main features of default organization are as follows:
   • Responsibility for process definition & maintenance is specific to a cohesive line of business.
   • Responsibility for process automation is an organizational role & is equal in importance to the process definition role.
   • Organizational role may be fulfilled by a single individual or several different teams.

![Default roles in a Software Line-of-Business Organization](image)

**Software Engineering Process Authority (SEPA)**

The SEPA facilities the exchange of information & process guidance both to & from project practitioners

This role is accountable to General Manager for maintaining a current assessment of the organization’s process maturity & its plan for future improvement

**Project Review Authority (PRA)**

The PRA is the single individual responsible for ensuring that a software project complies with all organizational & business unit software policies, practices & standards

A software Project Manager is responsible for meeting the requirements of a contract or some other project compliance standard

**Software Engineering Environment Authority (SEEA)**
The SEEA is responsible for automating the organization’s process, maintaining the organization’s standard environment, training projects to use the environment & maintaining organization-wide reusable assets. The SEEA role is necessary to achieve a significant ROI for common process.

**Infrastructure**

An organization’s infrastructure provides human resources support, project-independent research & development, & other capital software engineering assets.

2) Project organizations:

- The above figure shows a default project organization and maps project-level roles and responsibilities.
- The main features of the default organization are as follows:
  - **The project management team** is an active participant, responsible for producing as well as managing.
  - **The architecture team** is responsible for real artifacts and for the integration of components, not just for staff functions.
  - **The development team** owns the component construction and maintenance activities.
  - The assessment team is separate from development.
  - **Quality** is everyone’s into all activities and checkpoints.
  - Each team takes responsibility for a different quality perspective.

3) EVOLUTION OF ORGANIZATIONS:
The Process Automation:

**Introductory Remarks:**
The environment must be the first-class artifact of the process.

Process automation & change management is critical to an iterative process. If the change is expensive then the development organization will resist it.

Round-trip engineering & integrated environments promote change freedom & effective evolution of technical artifacts.

Metric automation is crucial to effective project control.

External stakeholders need access to environment resources to improve interaction with the development team & add value to the process.

The three levels of process which requires a certain degree of process automation for the corresponding process to be carried out efficiently.

Metaprocess (Line of business): The automation support for this level is called an infrastructure.
Macroproces (project): The automation support for a project’s process is called an environment.

Microproces (iteration): The automation support for generating artifacts is generally called a tool.

Tools: Automation Building blocks:
Many tools are available to automate the software development process. Most of the core software development tools map closely to one of the process workflows.

<table>
<thead>
<tr>
<th>Workflows</th>
<th>Environment Tools &amp; process Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>Workflow automation, Metrics automation</td>
</tr>
<tr>
<td>Environment</td>
<td>Change Management, Document Automation</td>
</tr>
<tr>
<td>Requirements</td>
<td>Requirement Management</td>
</tr>
<tr>
<td>Design</td>
<td>Visual Modeling</td>
</tr>
<tr>
<td>Implementation</td>
<td>Editors, Compilers, Debugger, Linker, Runtime</td>
</tr>
<tr>
<td>Assessment</td>
<td>Test automation, defect Tracking</td>
</tr>
<tr>
<td>Deployment</td>
<td>Defect Tracking</td>
</tr>
</tbody>
</table>

Figure 12-1. Typical automation and tool components that support the process workflows.
PROCESS AUTOMATION

The Project Environment:
The project environment artifacts evolve through three discrete states.
The Prototype Environment includes an architecture test bed for prototyping project architecture to evaluate trade-offs during inception & elaboration phase of the life cycle.
The Development environment should include a full suite of development tools needed to support various Process workflows & round-trip engineering to the maximum extent possible.
The Maintenance Environment should typically coincide with the mature version of the development.

There are four important environment disciplines that are critical to management context & the success of a modern iterative development process.

Round-Trip engineering
Change Management
Software Change Orders (SCO)
Configuration baseline Configuration Control Board
Infrastructure
Organization Policy
Organization Environment
Stakeholder Environment.

Round Trip Environment
Tools must be integrated to maintain consistency & traceability.
Round-Trip engineering is the term used to describe this key requirement for environment that support iterative development.
As the software industry moves into maintaining different information sets for the engineering artifacts, more automation support is needed to ensure efficient & error free transition of data from one artifacts to another.
Round-trip engineering is the environment support necessary to maintain Consistency among the engineering artifacts.
Change Management
Change management must be automated & enforced to manage multiple iterations & to enable change freedom. Change is the fundamental primitive of iterative Development.

I. Software Change Orders
The atomic unit of software work that is authorized to create, modify or obsolesce components within a configuration baseline is called a software change orders (SCO)
The basic fields of the SCO are Title, description, metrics, resolution, assessment & disposition
Change management

II. Configuration Baseline

A configuration baseline is a named collection of software components & supporting documentation that is subjected to change management & is upgraded, maintained, tested, statuses & obsolesced a unit.

There are generally two classes of baselines:

External Product Release

Internal testing Release

Three levels of baseline releases are required for most Systems:

1. Major release (N)
2. Minor Release (M)
3. Interim (temporary) Release (X)

Major release represents a new generation of the product or project.
A **minor** release represents the same basic product but with enhanced features, performance or quality.

**Major & Minor** releases are intended to be external product releases that are persistent & supported for a period of time.

**An interim** release corresponds to a developmental configuration that is intended to be transient.

Once software is placed in a controlled baseline all changes are tracked such that a distinction must be made for the cause of the change. Change categories are

- **Type 0:** Critical Failures (must be fixed before release)
- **Type 1:** A bug or defect either does not impair (Harm) the usefulness of the system or can be worked around
- **Type 2:** A change that is an enhancement rather than a response to a defect
- **Type 3:** A change that is necessitated by the update to the environment
- **Type 4:** Changes that are not accommodated by the other categories.

**Change Management**

**III Configuration Control Board (CCB)**

A CCB is a team of people that functions as the decision authority on the content of configuration baselines.

A CCB includes:

1. Software managers
2. Software Architecture managers
3. Software Development managers
4. Software Assessment managers
5. Other Stakeholders who are integral to the maintenance of the controlled software delivery system?

**Infrastructure**

The organization infrastructure provides the organization’s capital assets including two key artifacts - Policy & Environment

**I Organization Policy:**

A Policy captures the standards for project software development processes.

The organization policy is usually packaged as a handbook that defines the life cycles & the process primitives such as

- Major milestones
- Intermediate Artifacts
- Engineering repositories
- Metrics
- Roles & Responsibilities
Infrastructure

II Organization Environment
The Environment that captures an inventory of tools which are building blocks from which project environments can be configured efficiently & economically

Stakeholder Environment
Many large scale projects include people in external organizations that represent other stakeholders participating in the development process they might include

- Procurement agency contract monitors
- End-user engineering support personnel
- Third party maintenance contractors
- Independent verification & validation contractors
- Representatives of regulatory agencies & others.

These stakeholder representatives also need to access to development resources so that they can contribute value to overall effort. These stakeholders will be access through on-line

An on-line environment accessible by the external stakeholders allow them to participate in the process as follows

Accept & use executable increments for the hands-on evaluation.
Use the same on-line tools, data & reports that the development organization uses to manage & monitor the project
Avoid excessive travel, paper interchange delays, format translations, paper * shipping costs & other overhead cost
Figure 12-6. Extending environments into stakeholder domains