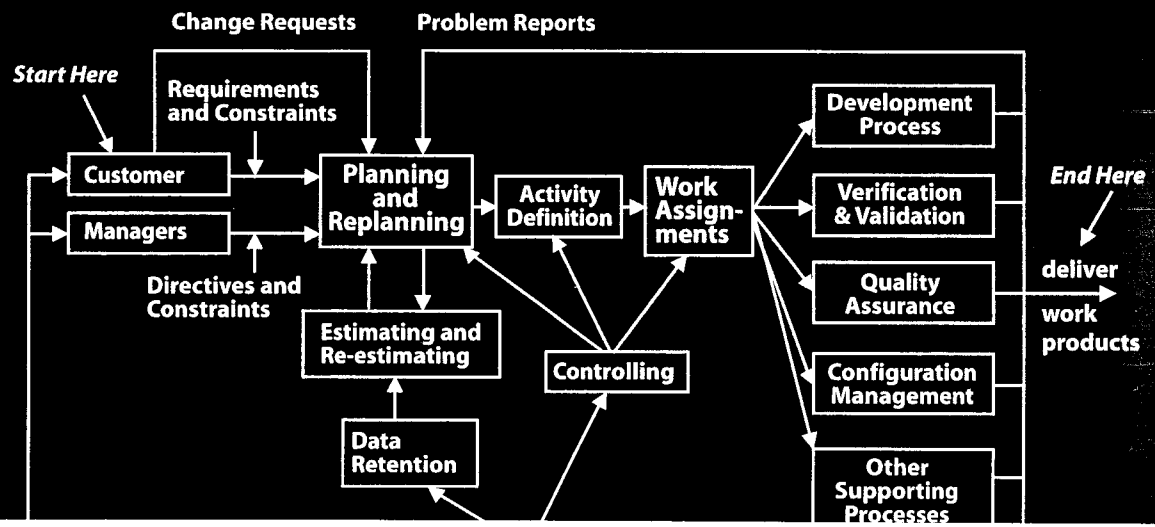
WWW. LINK AVAILABLE

# Managing AND Leading Software Projects

*Richard E. (Dick) Fairley*



# 5

# PROJECT PLANNING TECHNIQUES

Failing to plan is planning to fail.
—Alan Lakein

## 5.1 INTRODUCTION TO PROJECT PLANNING TECHNIQUES

Previous chapters of this text have addressed the requirements, constraints, and directives elements of the workflow model in Figure 1.1 of Chapter 1 (repeated here as Figure 5.1), the roles of customer and management, and the nature of plans and planning. This chapter is concerned with the planning, activity definition, and estimating elements highlighted in Figure 5.1. Additional estimation techniques are presented in Chapter 6.

Planning techniques, activity definition, and estimation of effort and schedule includes the following activities, which are a subset of the activities contained in Table 4.1*b* in Chapter 4.

- Develop an architecture decomposition view (ADV) of the product architecture and allocate requirements to the elements of the ADV
- Develop a work breakdown structure that includes work elements for the ADV modules and the allocated requirements for each element of work
- Develop work packages for the tasks in the work breakdown structure (WBS)
- Define a schedule of objectively measurable milestones
- Prepare a schedule network and identify the critical path(s)

*Managing and Leading Software Projects*, by Richard E. Fairley

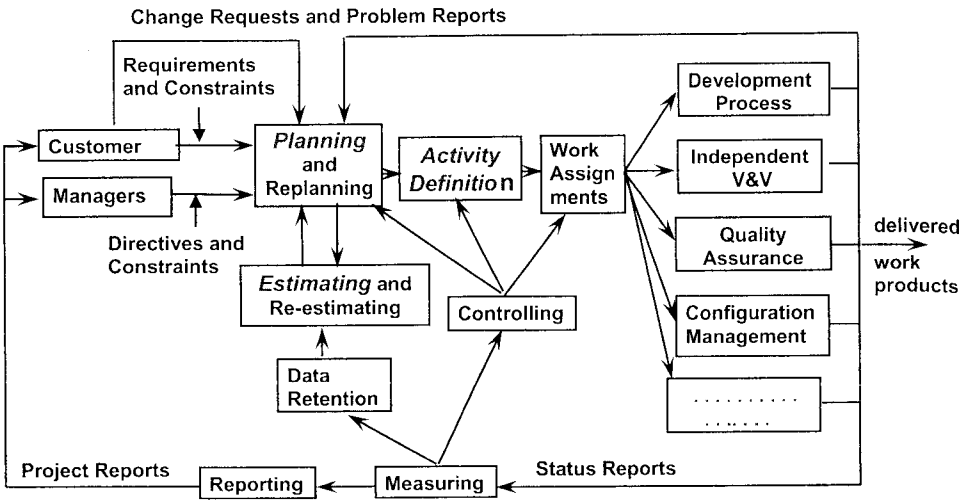



**FIGURE 5.1** Project workflow, emphasizing planning, estimating, and activity definition

- Prepare a PERT estimate of project duration
- Identify numbers and kinds of resources needed, when they will be needed, and for how long
- Prepare an estimate of optimal effort, cost, schedule, and resources
- Negotiate with the customer to obtain a balance among requirements, cost, and project duration that satisfies the project constraints

It is self-evident that you cannot prepare a plan for developing a software product if you don't know what product to make. It is equally evident that the more you understand about the product to be made, the more confident you will be in the details of your plan. The initial version of your project plan will be, by necessity, high level and imprecise; however, you can refine the plan as the architectural structure of the product evolves and as your understanding of the project grows based on clarification of the product foundations covered in Chapter 3 of this text (system requirements, system architecture, software requirements, and design constraints). Planning is thus an iterative process; the more you understand about the product the better plan you can make.

## 5.2 OBJECTIVES OF THIS CHAPTER

After reading this chapter and completing the exercises, you should understand:

- the scope of planning
- rolling-wave planning
- scenarios for developing a project plan
- developing an architecture decomposition view

- developing a work breakdown structure
- developing the project schedule
- developing resource profiles
- resource Gantt charts
- estimating project cost

The planning techniques presented in this chapter are informed by the Project Planning process area of the CMMI-DEV-v1.2 process framework, the planning elements of ISO and IEEE Standards 12207, IEEE Standard 1058, and the PMI Body of Knowledge. These elements are described in Appendix 5A to this chapter.

Terms used in this chapter and throughout this text are defined in the Glossary at the end of the text. Presentation slides for this chapter and other supporting material are available at the URL listed in the Preface.

## 5.3 THE SCOPE OF PLANNING

The scope of your project may involve developing requirements, negotiating schedule and budget, acquiring facilities and resources, building the software product,[19] installing it, training users, and maintaining the system on an ongoing basis. Or, you may be handed a set of changes to be made along with a schedule and a budget and be given the responsibility of managing a project to make the modifications. Or, your project may fall somewhere between these extremes. In any case, this chapter (and this entire text) considers the full scope of activities that may be required to manage large and complex software projects. As emphasized throughout this text, the activities of project management must be adapted and tailored to fit the needs of each project.

## 5.4 ROLLING-WAVE PLANNING

Rolling-wave planning acknowledges that it is impossible to develop plans at the level of detail indicated throughout this chapter during the initial planning phase of your software projects. When you are conducting a project, a recommended approach is to augment the high-level master plan with detailed plans for the coming month, for the subsequent month, and for three months hence. Each month the plans are moved forward one month, that is, moved forward in a rolling-wave manner. The plans for the next month should be detailed and specific. The plans for two and three months hence should be as specific as possible. Rolling the three-month plan forward each month provides an opportunity to:

[19] Software *products* are built by *vendors* for sale to numerous customers; software *systems* are built by *contractors* for specific individual customers on a contractual basis. The terms "system" and "product" are used interchangeably in this text unless the distinction is important; the distinction will be clarified in these cases.

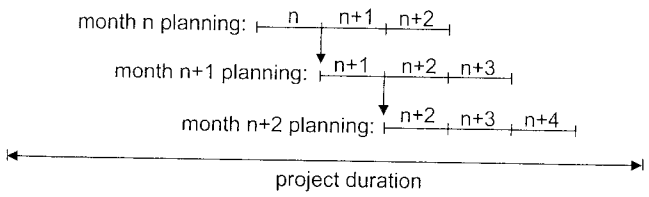



**FIGURE 5.2** Rolling-wave updating of detailed plans each month

- have resources available when they are needed,
- clear roadblocks and coordinate work activities, and
- identify and confront risk factors before they become problems.

Rolling-wave planning is illustrated in Figure 5.2.

## 5.5 SCENARIOS FOR DEVELOPING A PROJECT PLAN

At minimum, you must have some operational requirements for the product or system to prepare the initial version of your plan. Ideally you would have prioritized operational requirements, technical specifications, a functional block diagram, and a decomposition view of the product's architectural structure on which to base your plan. However, this ideal basis is seldom realized when preparing the initial version of a project plan.

Initial planning typically proceeds from one of the following scenarios:

1. You are given a set of operational requirements and constraints on one or more of the schedule, budget, and resources. For example, a system that will have a specified list of operational features and quality attributes must be delivered in 9 months; 6 software developers are available to implement the system. Your first task is to determine whether it is feasible to build the envisioned product (or modify an existing product) within those parameters. This may involve working with the customer to clarify the requirements and using historical data and rules of thumb to determine the feasibility of the project.

If the project is not feasible, with a high probability of success, you and your customer must prioritize the requirements into Essential, Desirable, and Optional categories (which is always a good thing to do). It must be possible to implement all of the Essential requirements, with a very high probability of success, within the development constraints on schedule, budget, and resources. The customer must agree to accept a product that implements all of the Essential requirements and as many of the Desirable requirements as can be implemented within the constraints on schedule, budget, and resources. Or, the development constraints must be relaxed, or some combination of de-scoping the requirements and relaxing the development constraints must be pursued.

2. You may be given a list of features and quality attributes and asked to estimate, and then commit to, the schedule, budget, and resources needed to develop a system

or product having those features and quality attributes. In this case you must first review, clarify, and elaborate whatever product information is available. You should not commit to requirements that are infeasible because of the current state of technology or lack of expertise in your organization; those requirements should be labeled as design goals to be achieved to the extent possible. In this scenario, you should prepare a range of estimates with associated probabilities of success and make a commitment to an estimate having not less than 90% probability of success. The assumptions on which your estimate and commitment are based must be documented and accepted by your customer.

3. You may be given a completion date and a budget and be asked to determine the characteristics of a product that can be built or modified within the constraints of specified time and money. For example, what operational features and quality attributes can you and 6 of your software developers build and deliver in 9 months for a product of a specified kind?

In any case, your initial project plan must achieve a balance among requirements, schedule, budget, resources, and technology. Subsequent revisions of your plan must maintain this balance as requirements and other factors change. In all cases, your first task in developing a project plan is to review, clarify, and further elaborate whatever information is available concerning the product to be built or modified.

For each of the scenarios above, the next step is to refine the requirements to remove areas of uncertainty and to prepare a decomposition view of the product architecture and a work breakdown structure as a basis for preparing a more accurate estimate.

## 5.6 DEVELOPING THE ARCHITECTURE DECOMPOSITION VIEW AND THE WORK BREAKDOWN STRUCTURE

Architectural design of software is concerned with specifying the software modules, their interrelationships, and their connections to the environment of the software. Several different kinds of views are used to document different kinds of relationships. The views are depicted using notations, such as those illustrated in Figure 5.3, to document structural, functional, and behavioral relationships. Other architectural views are also useful [Bass03].

A partial ADV for ATM software is presented in Figure 5.4. Note that the requirements listed in Table 5.1 are allocated to the leaf nodes of the financial transactions component of the ADV.

Note the use of terms "shall" for Essential requirements, "should" for desirable requirements, and "could" for Optional requirements. "shall" is a contractually binding term; "should" indicates desired but not essential requirements; "could" indicates options that could be included if time, budget, and resources permit.

Figure 5.5*a* illustrates a tree-structured representation of a work breakdown structure (WBS) for the ATM project. An alternative representation (an indented list) is presented in Figure 5.5*b*. The leaf nodes of the tree (or the list) specify tasks. A *task* is a smallest unit of project planning, measurement, and control. The higher level nodes in a WBS are *activities*; activities are composed of subordinate activities

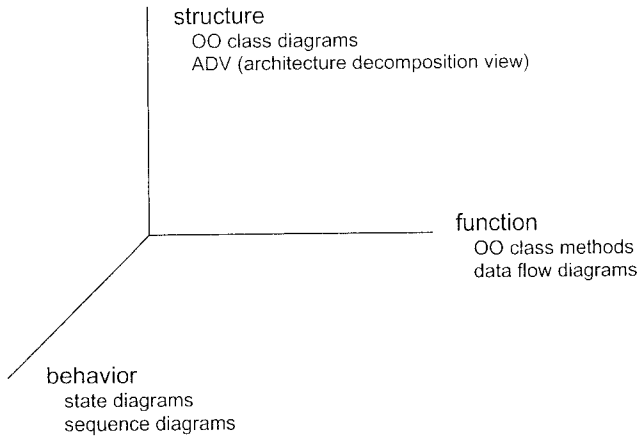



**FIGURE 5.3** Three architectural views of software and examples of notations used. The architecture decomposition view (ADV) specifies the hierarchical "is-part-of" relationship among software modules. The ADV is used by project managers (you) to develop the work breakdown structure (WBS)

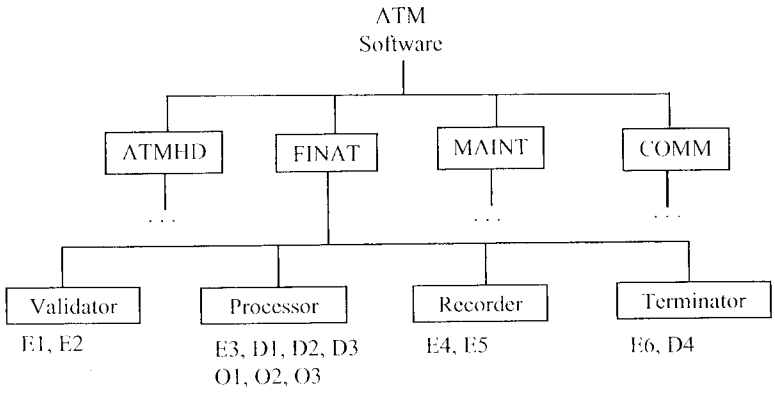


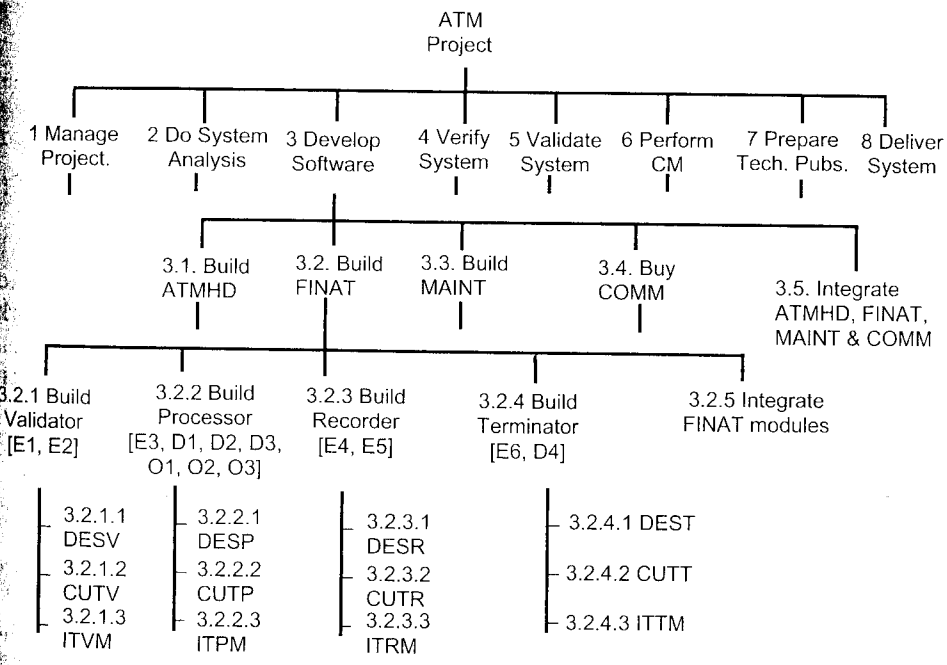

**FIGURE 5.4** A partial architecture decomposition view (ADV) of ATM software

and tasks. The relationships among activities and tasks in a WBS are thus containment or "is-part-of" relationships in the same way that the relationships among software module in an architectural decomposition view are "is-part-of" relationships among the modules.

The WBS is a fundamental tool for planning, estimating, measuring, and controlling a software project. The role of a WBS is to partition the activities and tasks of a software project into manageable units with clearly defined roles, responsibilities, and authorities for each unit. In addition a WBS depicts the interfaces and lines of communication among work activities and tasks. One of the primary design criteria

**TABLE 5.1 Some prioritized requirements for ATM software**

| | |
|---|---|
| | *Essential requirements* |
| E1 | Financial transactions shall be authorized by an ATM card and a password |
| E2 | Financial transactions shall be terminated if a customer fails to enter the correct password «settable» times |
| E3 | Financial transaction shall allow quick cash withdrawals |
| E4 | Financial transaction shall provide a printed receipt for each transaction |
| E5 | The ATM shall retain the information listed in the requirements specification, section 3.2.1, for each customer transaction |
| E6 | Financial transaction shall process Terminate requests from customers |
| | *Desirable requirements* |
| D1 | Financial transaction should accommodate balance inquiry transactions |
| D2 | Financial transaction should accommodate standard withdrawal transactions |
| D3 | Financial transaction should accommodate deposit transactions |
| D4 | Customers should be allowed to conduct multiple transactions per session |
| | *Optional requirements* |
| O1 | Financial transaction could support debit card transactions |
| O2 | Financial transaction could support payment of utility bills |
| O3 | Financial transaction could allow customers to purchase postage stamps which will be disbursed by the ATM hardware |




DESx: detailed design of module x; CUTx: coding & unit testing x; ITxC: integrating and testing of x

**FIGURE 5.5A** Tree-structured form of a WBS

1 Manage Project
2 Do System Analysis
3 Develop Software
  3.1 Build ATM Hardware Drivers (ATMHD)
  3.2 Build Financial Transaction Handler
    3.2.1 Build Validator [E1, E2]
      3.2.1.1 Design Validator
      3.2.1.2 Code & Unit Test Validator
      3.2.1.3 Integrate & Test Validator
    3.2.2 Build Transaction Processor (FINAT) [3, S1, D2, D3, O1, O2, O3]
      3.2.2.1 Design Transaction Processor
      3.2.2.2 Code & Unit Test Transaction Processor
      3.2.2.3 Integrate & Test Processor Components
    3.2.3 Build Recorder [E4, E5]
      3.2.3.1 Design Recorder
      3.2.3.2 Code & Unit Test Recorder
      3.2.3.3 Integrate & Test Recorder Module
    3.2.4 Build Terminator [E6,D4]
      3.2.4.1 Design Recorder
      3.2.4.2 Code & Unit Test Recorder
      3.2.4.3 Integrate & Test Recorder Module
    3.2.5 Integrate FINAT Modules
  3.3 Build Maintenance & Diagnostic Module (MAINT)
  3.4 Buy the Communications Package (COMM)
  3.5 Integrate ATMHD, FINAT, MAINT, and COMM
4 Verify System
5 Validate System
6 Perform CM
7 Prepare Technical Publications
8 Deliver System

**FIGURE 5.5B** Indented form of a WBS

for developing the decomposition view of software architecture (the ADV) is to decompose the product in a manner that permits assignment of concurrent work tasks to different teams and individuals; there is thus a close relationship between designing a software product and designing the work activities to build the product. This criterion can be stated as follows:

> *The decomposition view of software architecture (the ADV) must be structured to provide concurrent work assignments for those available to develop the software.*

Conversely, it can be said that the structure of the team that develops the software will influence the decomposition view of the delivered software [Conway68].

The distinction between an ADV and a WBS is often blurred by embedding the ADV directly into the WBS without rephrasing it. This blurring must be avoided. The elements of an ADV are product modules; they are specified by *noun phrases* that designate things, as in Figure 5.4. Work breakdown structures for software

projects are process-oriented, hierarchical decompositions of *work* activities and tasks. The elements of a WBS are specified by *verb phrases* that indicate actions to be taken, as in Figures 5.4*a* and 5.4*b* (e.g., *manage* project, *develop* software, *perform* CM). The elements of an ADV are related to the elements in a WBS by embedding within the WBS the work needed to develop or otherwise obtain the software modules.

The top level of your WBS should include all of the major work activities within the scope of your project; that is, the top level should encompass all work activities necessary to satisfy the requirements, constraints, and contractual commitments for your project (e.g., project management, system analysis, software development, verification and validation, and product delivery). Each node of the WBS in your initial project plan should be decomposed into sublevels until each of the following WBS decomposition criteria are satisfied:

1. hidden complexities are exposed (i.e., the job to be done is understood);
2. opportunities for reuse of existing software components can be identified;
3. the necessary hardware resources, such as computer memory and processor speed, can be specified (which may result in revision of the hardware requirements); and
4. estimates of effort needed to develop the software can be made.

Satisfying criterion 1 is necessary in order to satisfy criterion 2; this may not be possible without prototyping, feasibility studies, and revision of the requirements. Also the estimated effort to find, assess, and modify modules to be reused must be balanced against the effort required to develop new modules. It is much better to confront these issues early in your project rather than later.

The WBS depicted in Figures 5.5*a* and 5.5*b* is partially decomposed. Elements 1, 2, 4 to 8, and 3.1, 3.3, and 3.4 should be expanded as necessary to satisfy the WBS decomposition criteria. For example, decomposition of element 1, Manage Project, might be as follows:

1 Manage Project
- 1.1 Initiate project
  - 1.1.1 Identify stakeholders
  - 1.1.2 Develop/clarify requirements
  - 1.1.3 Prepare initial estimates
  - 1.1.4 Prepare initial project plan
  - 1.1.5 Obtain commitment to the plan
- 1.2 Conduct project
  - 1.2.1 Measure and control project
  - 1.2.2 Lead and direct personnel
  - 1.2.3 Communicate and coordinate
  - 1.2.4 Manage risk
- 1.3 Closeout project
  - 1.3.1 Obtain product acceptance
  - 1.3.2 Conduct postmortem sessions
  - 1.3.3 Prepare and distribute lessons-learned report
  - 1.3.4 Assist in reassigning project personnel

During planning the various paths in your WBS may be decomposed to different levels in order to satisfy the WBS decomposition criteria. Familiar work of low complexity will require less decomposition to permit confident estimates than a new kind of work of uncertain complexity. An identified opportunity for reuse of an existing module will require less decomposition if you are confident the candidate module will be suitable, and more decomposition to assess its suitability if you are less confident in it. The elements of a WBS are typically decomposed to three or four levels during planning; one or two additional levels are typically added during execution of the project.

## 5.7 GUIDELINES FOR DESIGNING WORK BREAKDOWN STRUCTURES

The work breakdown structure (WBS) is a fundamental tool for planning a software project and for measuring and controlling the progress of a project; it integrates the managerial and technical activities of a software project. A well-designed WBS is thus an essential element of a software project management plan. Fifteen guidelines for designing work breakdown structures are itemized in Table 5.2 and discussed below. Additional guidelines for using the WBS to track the progress of your project are presented in Chapter 11. As indicated by these guidelines, your WBS should be designed and structured with the same care used to design the architectural views of the software system or product.

**TABLE 5.2 Fifteen guidelines for designing work breakdown structures**

| Guideline | Work Breakdown |
|---|---|
| 1: | Use the Architecture Decomposition View (ADV) of the software architecture as the basis for developing the WBS. |
| 2: | Structure the ADV and the WBS to facilitate work assignments. |
| 3: | Develop and use process-oriented work breakdown structures. |
| 4: | Embed the work activities to develop and modify product modules in the WBS. |
| 5: | Partition the scope of the project into not more than 7 or 8 functional areas at the top level of the WBS. |
| 6: | Limit the fan-out of each element (i.e., the scope of each element) in the WBS to seven or less. |
| 7: | Limit the maximum depth of the WBS to six or fewer levels. |
| 8: | Use a decimal numbering system to systematically identify work activities and tasks. |
| 9: | Allocate prioritized requirements to development activities and tasks in the WBS. |
| 10: | Design the WBS top-down, bottom-up, and middle out. |
| 11: | Use work packages to specify project tasks. |
| 12: | Analyze work packages for desired properties. |
| 13: | Derive the schedule network from the work packages. |
| 14: | Determine resource requirements using the work packages and the schedule network. |
| 15: | Revise and elaborate the WBS periodically and as events dictate. |

*WBS Design Guideline 1: Use the decomposition view of the software architecture (ADV) as a basis for developing the WBS* As illustrated in Figure 5.4, the elements of an ADV are denoted by noun phrases; they are things. The work to develop the elements of an ADV is embedded in the WBS by adding appropriate verbs to the noun phrases in the ADV.

*WBS Design Guideline 2: Structure the ADV to facilitate work assignments* The decomposition view of the software architecture embedded in the WBS should provide opportunities for concurrent work activities. For example, the hardware driver, financial transaction, diagnostics, and communication modules in Figure 5.4 can be developed or otherwise obtained by different teams working concurrently; the COMM package can also be procured concurrently. Similarly the validator, processor, recorder, and terminator modules of the financial transaction module can be developed by individuals or teams working concurrently on the modules.

*WBS Design Guideline 3: Develop and use process-oriented work breakdown structures* As illustrated in Figures 5.5*a* and 5.5*b*, a WBS specifies work activities, tasks, and the containment relationships among them. Each activity and task is specified by a verb phrase that indicates actions to be taken.

*WBS Design Guideline 4: Embed the work activities to develop and modify product modules in the WBS* The activities and tasks to develop or otherwise obtain the modules in the decomposition view of the software architecture in Figure 5.4 are embedded in the WBS of Figures 5.5*a* and 5.5*b* by converting the noun phrases in Figure 5.4 to the corresponding verb phrases in Figures 5.5*a* and 5.5*b*.

*WBS Design Guideline 5: Partition the scope of the project into not more than seven or eight functional areas at the top level of the WBS* The top level of a WBS should partition all the work activities to be accomplished into seven or eight elements, as illustrated in Figures 5.5*a* and 5.5*b*. Limiting the number of activities to seven or eight elements at the top level facilitates management of the intellectual complexity of a project by partitioning it into a small number of work activities to be directly managed by you, the project manager, and by those who report directly to you. Subordinate activities and tasks are assigned to team leaders and team members who are responsible for those activities and tasks.

*WBS Design Guideline 6: Limit the fan-out of each element in the WBS to seven or eight* The fan-out of a WBS element is the number of branches connecting an element to its immediate subordinate elements. As described above, one role of a WBS is to designate roles, responsibilities, and authorities in a software project. Limiting fan-out has the advantage of controlling the complexity of each work activity by limiting the number of subordinate activities or tasks that must be managed to accomplish that work activity; intellectual manageability of a project is thus obtained. This advantage is not dissimilar to the advantage gained by limiting the fan-out of product modules in the architecture decomposition view of software architecture.

*WBS Design Guideline 7: Limit the depth of the WBS to six or fewer levels* The depth of a WBS is the length of the longest path(s) in the WBS; in Figures 5.5*a* and 5.5*b* the depth is 4, which is indicated by the number of digits in the lowest level task designators. Limiting the depth of each path in a WBS to six or fewer levels has similar advantages to limiting fan-out in controlling intellectual manageability of a project. Decomposing a path more than six levels to satisfy the WBS decomposition criteria of exposing complexity and risk factors indicates areas of the product and/or process architecture that must be studied in greater detail and reconfigured as appropriate.

Consider a software project that has 10 developers working for 12 months (480 staff-weeks). If each development task represents one staff-week of effort there would be 480 leaf nodes in the software development sub-tree of the WBS. Assuming software development (design, code, test) is 50% of the total effort, and assuming all project tasks are decomposed to a level of one staff-week, the WBS would have 960 leaf nodes. In contrast, a WBS that has 6 levels with a fan-out of 7 at each node (a $6 \times 7$ WBS) would have $7^6$ leaf node tasks (117,659). Clearly, a $6 \times 7$ WBS is sufficient for the largest mega-projects.

Another design consideration: if a node in your WBS has, say, 3 or 4 sub-levels with fan-outs of 5 or 6 at each node, this may indicate the need to "spin off" that segment of the WBS into a separate subproject (a $3 \times 6$ WBS has 216 leaf node tasks).

*WBS Design Guideline 8: Use a decimal numbering system to specify work activities and tasks in the WBS* The numbering system illustrated in Figures 5.5*a* and 5.5*b* provides a systematic way of specifying the containment and sibling relationships among activities and tasks. Task 3.2.4.3, in Figures 5.5*a* and 5.5*b*, for example, is on level 4 of the WBS because there are 4 digits in its identifier; it is the 3th element of the 4th element of the 2nd element of the 3rd element in the WBS. All elements having a 2 in the 2nd position are sibling software work activities (e.g., the Build FINAT activities in Figures 5.5*a* and 5.5*b*). Some organizations specify the numbers to be used in designating the top elements in work breakdown structures. For example, every work element in every WBS that starts with a 3 would designate software work, and elements of work starting with a 6 would denote configuration management. This convention facilitates uniform reporting and accounting practices among projects across an organization.

*WBS Design Guideline 9: Allocate prioritized requirements to development activities and tasks in the WBS* Each activity and task in a WBS indicates work that must be accomplished and work products that must be produced. Allocating requirements to development activities and tasks, as illustrated in Figures 5.5*a* and 5.5*b*, provides prioritized specifications for the work products to be produced by those activities and tasks. In addition to specifying the features to be provided, the allocated requirements should specify design constraints, capacities, performance, interfaces, and quality attributes, as appropriate.

Product features should be uniquely allocated to tasks so that work assignments for building the modules are clearly defined. Other requirements (design constraints, capacities, performance, interfaces, and quality attributes) may apply to multiple modules, perhaps including the entire system or product, as discussed in

Chapter 3. Those requirements should be allocated to the highest level activity to which they apply and be "flowed down" to the descendents of that activity.

*WBS Design Guideline 10: Design the WBS top down, bottom up, and middle out* Designing a WBS is best done iteratively by interleaving top-down, bottom-up, and middle-out strategies. In this regard the cognitive processes involved when developing a WBS (i.e., designing a software project) are not unlike those observed in designers of software [Walz93].

Top-down development of a WBS proceeds by partitioning the scope of the project into a set of top-level activities and successively decomposing activities until a set of tasks is specified that satisfy the WBS decomposition criteria listed above. Bottom-up development of a WBS proceeds by identifying a set of tasks that must be performed and grouping related tasks into activities. Middle-out development of a WBS proceeds by identifying a mid-level activity that must be performed, decomposing it into tasks and/or subordinate activities, grouping it with similar activities, and connecting the related set of activities to a higher level activity.

*WBS Design Guideline 11: Use work packages to specify development tasks* Work packages are specifications for the activities and tasks in a WBS. Tasks are the lowest level elements in the WBS. Work packages for activities are aggregations of work packages for subordinate activities and tasks.

A work package should contain:

- the corresponding WBS number and name,
- a brief description of the task,
- estimated duration,
- resources needed,
- predecessor and successor tasks,
- work products to be produced,
- work products that will be placed under version control (baselined),
- risk factors (i.e., potential problems that might interfere with successful completion of the work package), and
- objective acceptance criteria for the work products generated by the task.

A template for work packages is illustrated in Table 5.3*a*; an example is provided in Table 5.3*b*.

*WBS Design Guideline 12: Analyze work packages for desired properties* The attributes of work packages, and collections of work packages, can be analyzed to determine various project factors. For example, the estimated cost of personnel to execute a work package can be determined from the numbers and kinds of people specified and the estimated duration of the task. In Table 5.3*b*, the cost of personnel is the loaded salaries (i.e., pay plus overhead) for 10 staff-weeks of senior designer effort. The cost of other resources can be similarly determined: the workstation and software tools in Table 5.3*b* may be available at no cost, or a cost to be borne by this task, or the cost may be amortized across this task and other tasks that will use

**TABLE 5.3A Template for work packages**

| | |
|---|---|
| Task identifier: | «WBS number and name» |
| Task description: | «brief description» |
| Estimated duration: | «days or weeks» |
| Resources needed: | |
| Personnel: | «numbers of people needed to complete this task» |
| Skills: | «personnel skills needed to complete this task» |
| Tools: | «software and hardware needed» |
| Travel: | «to where? for how long?» |
| Other: | «other resources needed to complete this task» |
| Predecessor tasks: | «to be completed before this task can begin» |
| Successor tasks: | «to start after this task is completed» |
| Work products: | «outputs of this task» |
| Baselines: | «work products to be placed under version control» |
| Risk factors: | «potential problems for this task» |
| Acceptance criteria: | «for the work products of this task» |

**TABLE 5.3B A work package example**

| | |
|---|---|
| Task identifier: | 3.2.2.1 Design transaction processor |
| Task description: | Specify internal architecture of the transaction processor module |
| Estimated duration: | 2 weeks |
| Resources needed: | |
| Personnel: | 2 senior telecom designers |
| Skills: | Designers must know UML |
| Tools: | One workstation running Rapsody |
| Travel: | Three day design review in San Diego for 2 people |
| Predecessor tasks: | 3.2.1 Develop system architecture |
| Successor tasks: | 3.3.2.2 Implement transaction processor |
| Work products: | Architectural specification for transaction processor and test plan |
| Baselines created: | Architectural specification for transaction processor and text plan |
| Risk factors: | Designers not identified |
| Acceptance criteria: | Successful design inspection by peers and approval of transaction processor design by the software architect |

those resources; the cost of travel can be determined and included in the cost of executing the work package (in Table 5.3*b*, two round trips to San Diego and 3 days travel support for 2 people).

The estimated costs for a collection of work packages can be aggregated (i.e., rolled up) to determine the elements of cost for various kinds of activities and to determine the overall cost estimate for the parent activity. Estimated costs of development tasks and activities can be rolled up to provide an estimated cost for software development, which can be used as a basis of estimation for the entire project. For example, a project would be estimated to cost $100,000 USD if software development was estimated to cost $50,000 USD and was estimated to be 50% of

overall project cost (50% perhaps determined from historical data within the organization).

If the roll-up of costs results in an estimate that exceeds the constraint on the project budget, you can start at the top level and reallocate portions of the budget to activities and tasks in a top-down manner so that the allocations to the subordinate elements of each activity do not exceed the amount allocated to that activity. This may involve eliminating or simplifying some product requirements and/or incurring greater levels of risk.

*WBS Design Guideline 13: Derive the schedule network from the work packages* A schedule network for a set of tasks can be constructed from the durations, predecessors, and successors of the work packages for those tasks, as explained in the following section of this chapter. Constructing the schedule network may reveal discontinuities, circularities, and other inconsistencies among predecessor and successor tasks that can be resolved by iterative refinement of the work package specifications.

*WBS Design Guideline 14: Determine resource requirements using the work packages and the schedule network* Knowing the time in the schedule when various tasks are planned to occur permits determination of the dates when various kinds of resources will be needed and the durations for which they will be required; for example, the need date for the unidentified senior designers in Figure 5.5*b* can be determined from the development schedule. If the need date is three months hence, there is adequate time to acquire the designers; if the need date is next week, you are probably in big trouble because failure to complete the work package on schedule will delay subsequent tasks and might delay completion of the project. Resource profiles for the various kinds of resources needed can be produced by summing up the resource requirements across the schedule, as illustrated later in this chapter.

*WBS Design Guideline 15: Revise and elaborate the WBS periodically and as events dictate* The elements of the initial WBS are decomposed to levels that satisfy the WBS decomposition criteria. As the project evolves, understanding grows and circumstances change; increased detail can be added to facilitate work assignments to individuals and teams. Additional work elements may be identified and others revised. The WBS should be updated each month in a rolling-wave manner. Also events such as major changes to requirements, schedule, and resources must be reflected in a revised WBS. The WBS must be placed under version control to clearly identify the current version and to provide a historical record of the evolution of the WBS.

An alternative approach is to interchange the order of guidelines 11, 12, 13, and 14 by first developing the schedule network and resource estimates for each task in the schedule network (guidelines 13 and 14) and then using the schedule network and resource estimates to specify and analyze the work packages (guidelines 11 and 12). In any case, as will be shown in Chapter 8, work package specifications for the WBS elements are essential for allocating the work to development teams and tracking the progress of their work.

## 5.12 KEY POINTS OF CHAPTER 5

- Project plans must be consistent with product requirements; you cannot prepare a plan for developing a software product if you don't know what product to make.
- The more you understand about the product to be made, the more confident you will be in the details of your plan.
- A project plan must be updated periodically and as events dictate using a rolling-wave approach.
- Your initial plan and subsequent plans must maintain a balance among requirements, schedule, budget, and resource availability.
- Essential elements of a project plan include a WBS, an activity network, resource profiles for the various kinds or resources, and strategies for dealing with identified risk factors.
- The work breakdown structure (WBS) is a fundamental tool for planning, tracking, and controlling a software project.
- The architecture decomposition view (ADV) of the software architecture provides the basis for developing a WBS.
- The ADV is product-oriented; noun phrases are used to specify things.
- The WBS is process-oriented; verb phrases are used to specify activities and tasks.
- Using the guidelines for designing a WBS will ensure that the WBS is designed with the same care that is used to design the product.
- Your initial WBS should be decomposed to satisfy the WBS decomposition criteria.
- Work packages are the specifications for tasks and activities in the WBS.
- Work packages for activities are aggregations of work packages for subordinate tasks and activities.
- The schedule network, resource requirements, cost estimates, and risk factors can be derived from work packages.
- The critical path method (CPM) can be used to determine the minimum estimated duration of a project and the slack times associated with noncritical tasks.
- The Program Evaluation and Review Technique (PERT) can be used to determine the times, at various levels of probability, required to reach project milestones, including the final milestone.
- A task-Gantt chart can be used to depict the critical path, illustrate slack times for noncritical tasks, and determine resource profiles for the various kinds of resources.
- A resource-Gantt chart can be used to depict the resource loading for various resources.
- Acceptable options for reconciling schedule/resource conflicts include reconfiguring the schedule network, extending the schedule so that fewer resources are needed in peak weeks, adding more resources to maintain the schedule, using more productive resources so that fewer numbers are needed, descoping

the requirements so that fewer resources and less time are needed, and combinations of the above.

- Unacceptable options for reconciling schedule/resource conflicts include producing an unrealistic plan that has no chance of being successfully implemented; planning for overtime; and reducing or eliminating quality control tasks such as inspections, reviews, and testing.
- Resource profiles can be used to calculate effort and the costs of the various resources; project schedule can be determined from the critical path or from PERT calculations.
- SEI, ISO, IEEE, and PMI provide frameworks, standards, and guidelines for project planning techniques (see Appendix 5A to this chapter).

## REFERENCES


[Bass03] Bass, L., P. Clements, and R. Kazman. *Software Architecture in Practice*, 2nd ed. Addison Wesley, 2003.

[CMMI06] SEI, *CMMI® Models and Modules*. http://www.sei.cmu.edu/cmmi/models/, 2006.

[Conway68] Conway, M. E. "How Do Committees Invent?" *Datamation* (April 1968). Vol. 14, No. 4, pp. 28–31.

[IEEE1058] IEEE Std 1058™–1998. *IEEE Standard for Software Project Management Plans*. Engineering Standards Collection. IEEE Product: SE113. Institute of Electrical and Electronic Engineers, August 2003.

[IEEE12207] IEEE/EIA 12207.0/.1/.2. *Industry Implementation of International Standard ISO/IEC 12207:1995 Standard for Information Technology–Software Life Cycle Processes*. Engineering Standards Collection; IEEE Product: SE113. Institute of Electrical and Electronic Engineers, August 2003.

[PMI04] PMI. *A Guide to the Project Management Body of Knowledge*, 3rd ed. (PMBOK® Guide). Project Management Institute, 2004.

[Walz93] Walz, D. B., J. Elam, and B. Curtis. Inside a software design team: Knowledge acquisition, sharing, and integration. Communications of the ACM, 36 (October 1993). pp. 63–67.


## EXERCISES

**5.1.** List and briefly explain three factors that might prevent you, as the project manager, from preparing a project estimate that has a 90% or greater probability of success.

**5.2.** The assumptions on which your estimate and your commitment are based must be documented and accepted by your manager and your customer. List and briefly explain five (relevant and reasonable) assumptions you might make in preparing an estimate that would be accepted.

**5.3.** The architecture decomposition view depicts the hierarchical "is-part-of" containment relationship among software modules. List and briefly explain the desirable attributes of software modules in an ADV.