Lecture Notes for Lecture 10 of CS 5500 (Foundations of Software Engineering) for the Spring 2021 session at the Northeastern University San Francisco Bay Area Campuses.

Managing and Tracking Project Resources

Philip Gust,
Clinical Instructor
Department of Computer Science

Information and examples in this lecture are based on recommended readings.

http://www.ccis.northeastern.edu/home/pgust/classes/cs5500/2021/Spring/index.html

Managing and Tracking Project Resources

Review of Lecture 9

- Communication is key in a software project. For a successful project execution, effective communication to all stakeholders is essential.
- According to a Project Management Institute survey report, one in five project fail because of ineffective communications.
- In this lecture, we studied communication in software projects, look at ways to adopt effective communication methods and examined some tools for project communication.

Managing and Tracking Project Resources

Introduction

- HP co-founder Bill Hewlett observed that you cannot manage what you cannot measure. He was a strong proponent of the importance of tracking as well as managing engineering processes.
- He led the effort to create a corporate engineering group within HP to help product groups track projects and use the information to improve their management of the process.
- The information they gained by tracking the progress of projects, and the tools they developed, gave HP a significant time and cost advantage in product development over competitors who did not.

Managing and Tracking Project Resources

Introduction

- Managing and tracking software development enables teams to assess progress against their own goals and estimates, and to make adjustments as soon as possible.
- In the next two lectures, we will explore three areas of software development where managing and tracking plays an especially important role in the success of software projects.
- In this lecture we will study how to manage and track project resources. In the next lecture we will study how to manage and track project communications and the development process.

Managing and Tracking Project Resources

Introduction

- Project resources refer to those things that the team requires to carry out a software project and, complete it successfully.
- In a software project, code is not a resource: it is the product of the development process and does not exist until after the project is completed.
- The two primary resources of a software development project are:
  - the *effort* of people who create the software
  - the *time* that the software takes to complete

Managing and Tracking Project Resources

Introduction

- The goals of managing and tracking project resources are to
  - ***plan*** ways to utilize the resources that minimize the resources required to complete the project.
  - ***track*** the development process and determine whether the use of the resources continues to match the plan
  - ***revise*** the plan by adjusting utilization of the resources, the amount of resources available, or the requirements, in a way that satisfies the needs of the customer.

Managing and Tracking Project Resources

Planning resource utilization

- The reality of a software project is that hundreds of tasks must be completed to accomplish the overall project.
- Some of the tasks are relatively independent and can be worked on whenever there is a sufficient block of time and developers are available to complete them
- Other tasks are on a critical path. If these tasks are not completed by a required time, the completion date of the entire project will fall behind because later tasks depend on them.

Managing and Tracking Project Resources

Planning resource utilization

- Planning resource utilization involves
  - determining the tasks that must be completed for the project
  - producing a range of estimates for each tasks under different resource utilization assumptions (developers and time)
  - identifying the network of inter-dependencies among the tasks, and the tasks that are critical within the network
  - Allocating resources to the tasks in such a way as to minimize the overall resources required to complete the project, and the impact of delays that can occur during the project

Managing and Tracking Project Resources

Planning resource utilization

- *Project scheduling* distributes estimated effort across a planned duration by allocating the developers and effort to tasks. This schedule evolves over time.
- During early stages of project planning, a *macroscopic schedule* identifies major process framework activities and the product functions to which they are applied.
- As the project gets under way, each entry on the macroscopic schedule is refined into a *detailed schedule*. Specific actions and tasks to accomplish an activity are identified and scheduled.

Managing and Tracking Project Resources

Planning resource utilization

- Scheduling for software engineering projects can be viewed from two rather different perspectives.
- In the first, an end date for release has already been established. The software organization is constrained to distribute effort within the prescribed time frame.
- The second perspective sets a rough chronological bounds, but effort is distributed to make best use of resources, and an end date is defined after careful analysis of the software.
- Unfortunately, the first situation occurs far more frequently than the second.

Managing and Tracking Project Resources

Tracking the development process

- Once a development plan and schedule is in place, it is important to track the actual development process and compare progress on tasks and slips that occur.
- The *residual* of a task is the difference between the scheduled and actual completion with the number of developers and the planned task duration.
- Residuals can result from, under-estimating the amount of time required for the developers assigned. They can also result from under-estimating unknows that reduce the certainty of the task.

Managing and Tracking Project Resources

Tracking the development process

- If a task with a residual is not a critical task, the impact is that its developers, who were supposed to start on their next tasks, will be delayed from working on those tasks.
- If a task with a residual is a critical task, not only will later tasks for those developers be delayed, but other developers whose tasks depend on the critical task will be blocked from starting their tasks.
- The impact of residuals can be modeled, and effect on the overall schedule can be estimated. A schedule should build in sufficient *slack* based on task uncertainties to ensure on-time completion.

Managing and Tracking Project Resources

Revising the plan

- If the results from tracking the development process indicates a slip in schedule that is unlikely to be balanced by gains in other parts of the process, there are several alternatives to consider,
  - Re-evaluate the plan and determine whether there is another way to allocate existing resources resulting in a schedule that makes up for the slip, by re-ordering tasks or reallocating developers, or both.
  - Determine whether additional time can be allotted to the schedule to offset the slip caused by the residuals, or whether adding developer effort would be effective if additional time is not available.
  - Find out whether the requirements can be modified in such a way that similar functionality that requires less development time is possible, or functionality can be spread out over several releases.

Managing and Tracking Project Resources

Revising the plan

- Regardless of how the plan is revised, it should also be re-examined to see if similar risks are lurking, and how they can be reduced before moving on.
- This information should be captured for later used by the current team and disseminated for use by other teams within the same organization (“lessons learned”).

Managing and Tracking Project Resources

Principles for project scheduling

- A number of basic principles help guide the creation of a software project schedule. Each is applied as the schedule evolves.
  - ***Compartmentalization***. Compartmentalize the project into a number of manageable activities and tasks. To accomplish this, decompose both the product and the process.
  - ***Interdependency***. Some tasks must occur in sequence, while others can be in parallel. Some cannot commence until the work product produced by another is available. Others can occur independently.
  - ***Time allocation***. Each task must be allocated some number of work units, a start date and a completion date based on dependencies, and whether work will be conducted on a full-time or part-time basis.

Managing and Tracking Project Resources

Principles for project scheduling

- Several basic principles help guide the creation of a software project schedule. Each is applied as the schedule evolves.
  - ***Effort validation***. Every project has a defined number of developers on the team. As time allocation occurs, ensure that no more than the number of developers has been scheduled at any given time.
  - For example, a project has three developers (e.g., three person-days are available per day of assigned effort, assuming 100% load). On a given day, seven concurrent tasks must be accomplished.
  - If each task requires 0.50 person-days of effort, more effort has been allocated than there are developers to do the work. This is known as *over-allocation*.

Managing and Tracking Project Resources

Principles for project scheduling

- Several basic principles help guide the creation of a software project schedule. Each is applied as the schedule evolves.
  - ***Defined responsibilities***. Every task that is scheduled should be assigned to a specific team member.
  - ***Defined outcomes***. Every task that is scheduled should have a defined outcome. The outcome is a work product (e.g., a component design) or part of a work product. These may be combined in deliverables.
  - ***Defined milestones***. Every task or group of tasks should be associated with a project milestone, when one or more work products has been reviewed for quality and has been approved.

Managing and Tracking Project Resources

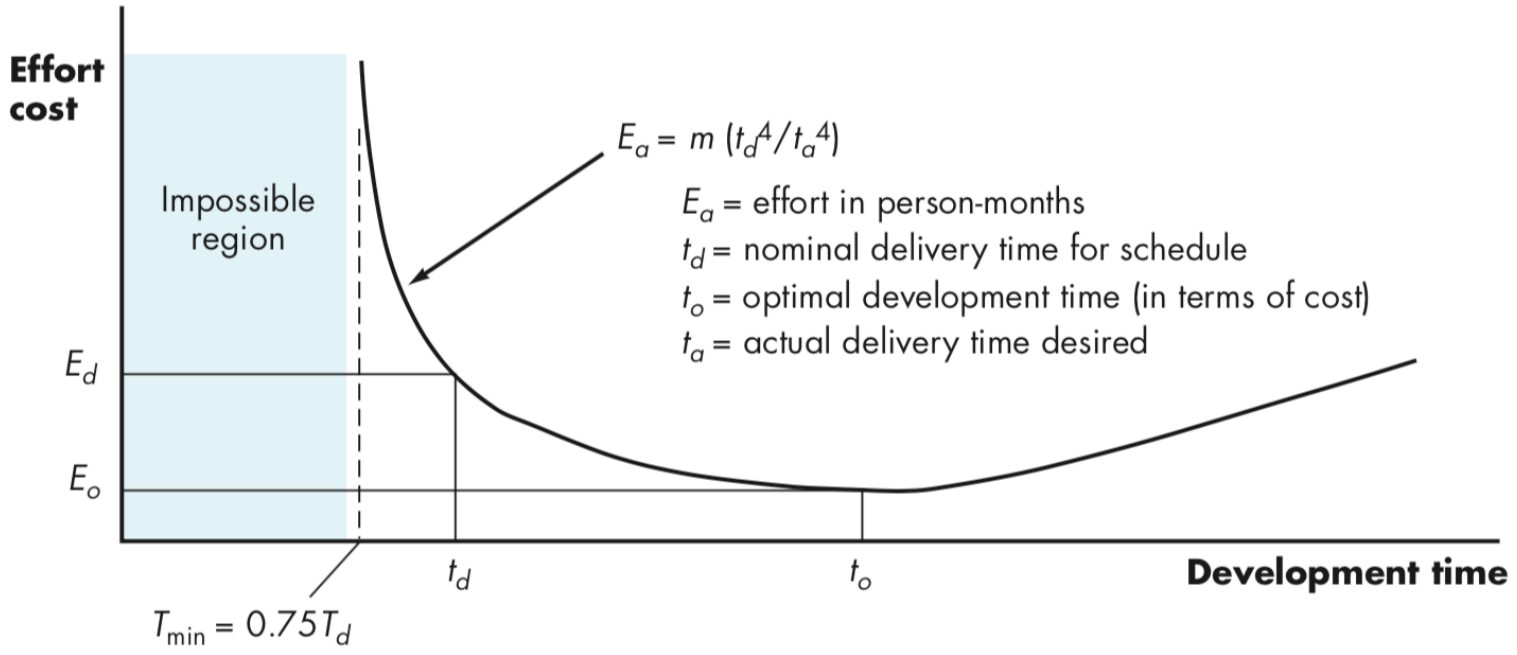
People vs. effort

- A common myth, believed by many managers, is that “If we fall behind schedule, we can always add more developers and catch up later in the project.”
- However, adding developers late in a project often has a disruptive effect on the project, causing schedules to slip even further.
- The developers added must learn the system, and the those who teach them are the same people who were doing the work. While teaching, no work is done, and the project falls further behind.
- More people increases the communication paths and complexity of communication throughout a project. Communication is essential, but more paths requires more effort and therefore more time.

Managing and Tracking Project Resources

People vs. effort

- The *Putnam-Norden-Rayleigh (PNR) Curve* shows the relationship between effort applied and delivery time for a software project.



Managing and Tracking Project Resources

People vs. effort

- The curve indicates a minimum value t_o indicating the least cost for delivery (i.e., delivery time with the least effort). As we move left of t_o (i.e., we try to speed up delivery), the curve rises nonlinearly.
- Assume that a project team estimated a level of effort E_d is needed for a nominal delivery time t_d that is optimal in terms of schedule and available resources.

Managing and Tracking Project Resources

People vs. effort

- Accelerating delivery, the curve rises sharply to the left of t_d. In fact, the PNR curve indicates the project delivery time cannot be compressed much beyond $0.75t_d$.
- With further compression, the project moves into “the impossible region” and risk of failure becomes very high. The PNR curve also indicates that the lowest cost delivery option, $t_o = 2t_d$.

Managing and Tracking Project Resources

People vs. effort

- The implication is that delaying project delivery can reduce costs significantly. Consider a complex real-time software project with an estimated 33 000 LOC and 12 person-years of effort.
- We can use Putnam and Meyers' software equation, an empirical estimation model from lecture 8 to calculate that if we assign 8 developers, the project can be completed in 1.3 years.
- However, by extending the date to 1.75 years, the highly non-linear nature of the model yields 3.8 person-years.
- This implies that extending the end-date by six month can reduce the number of developers required from 8 to 4.
- Of course, this must be weighed against business cost associated with the delay.

Managing and Tracking Project Resources

Effort distribution

- Each of the software project estimation techniques discussed in lecture 8 leads to estimates of work units (e.g., person-months) required to complete software development.
- A recommended distribution of effort across the software process is often referred to as the 40-20-40 rule. Forty percent of all effort is allocated to front-end analysis and design.
- A similar percentage is applied to back-end testing. You can correctly infer that coding (20 percent of effort) is deemphasized.

Managing and Tracking Project Resources

Effort distribution

- This effort distribution should be used as a guideline only. The characteristics of each project dictate the distribution of effort.
- Work expended on project planning rarely accounts for more than 2 to 3 percent of effort, unless the plan commits an organization to large expenditures with high risk.
- Customer communication and requirements analysis may comprise 10 to 25 percent of project effort. Effort on analysis or prototyping should increase proportionally with project size and complexity.
- A range of 20 to 25 percent of effort is normally applied to software design. Time for design review and subsequent iteration must also be considered.

Managing and Tracking Project Resources

Effort distribution

- Because of the effort applied to software design, code should follow with relatively little difficulty. A range of 15 to 20 percent of overall effort can be achieved.
- Testing and subsequent debugging can account for 30 to 40 percent of software development effort. The criticality of the software often dictates the amount of testing that is required.
- If software is human rated (i.e., software failure can result in loss of life), even higher percentages are typical.

Managing and Tracking Project Resources

Effort distribution

- Note that the 40-20-40 rule is under attack. Some believe more than 40 percent of effort should go to analysis and design.
- On the other hand, some proponents of agile development argue that less time should be expended “up front” and that a team should move quickly to construction.

Managing and Tracking Project Resources

Defining a task set

- Regardless of the process model, the work that a software team performs is achieved through a set of tasks that enable them to define, develop, and ultimately support a software product.
- No one task set is appropriate for all projects. The set of tasks that are appropriate for a large, complex system would likely be perceived as overkill for a small, relatively simple software product.
- An effective software process should define a collection of task sets, each designed to meet the needs of different types of projects.

Managing and Tracking Project Resources

Defining a task set

- To develop a project schedule, a task set must be distributed on the project timeline. Most software organizations encounter the following types of projects:
  - ***Concept development*** projects initiated to explore some new business concept or application of some new technology.
  - ***New application development*** projects undertaken as a consequence of a specific customer request.
  - ***Application enhancement*** projects occur with major changes to function, performance, or interfaces that are observable by the end user.
  - ***Application maintenance*** projects that correct, adapt, or extend existing software in ways that may not be immediately obvious to the end user.
  - ***Reengineering*** projects that are undertaken with the intent of rebuilding an existing (legacy) system in whole or in part.

Managing and Tracking Project Resources

Task set example

- **Concept development** projects are initiated when the potential for some new technology must be explored. Concept development projects are approached by applying the following major tasks:
  1. ***Concept scoping*** determines the overall scope of the project.
  2. ***Preliminary concept planning*** establishes the organization's ability to undertake the work implied by the project scope.
  3. ***Technology risk assessment*** evaluates the risk associated with the technology to be implemented as part of the project scope.
  4. ***Proof of concept*** demonstrates the viability of a new technology in the software context.
  5. ***Concept implementation*** implements the concept representation in a manner that can be reviewed by a customer and is used for "marketing" purposes when a concept must be sold to other customers or management.
  6. ***Customer reaction*** to the concept solicits feedback on a new technology concept and targets specific customer applications

Managing and Tracking Project Resources

Task refinement

- The major tasks may be used to define a macroscopic schedule for a project. However, the macroscopic schedule must be refined to create a detailed project schedule.
- Refinement begins by taking each major task and decomposing it into a set of subtasks with related work products and milestones.
- For example consider *Task 1, Concept Scoping*. Task refinement can be accomplished using an outline format. In this example, process design language illustrates the flow of the concept scoping activity.
- The tasks and subtasks noted in the process design language refinement form the basis for a detailed schedule for the concept scoping activity.

Managing and Tracking Project Resources

Task refinement

1. Concept Scoping
    1. Identify need, benefits and potential customers
    2. Define desired output/control and input events that drive the application
        1. TR: Review written description of need
        2. Derive a list of customer visible outputs/inputs
        3. TR: Review outputs/inputs with customer and revise as required
    3. Define the functionality/behavior for each major function
        1. TR: Review output and input data objects derived in task 1.2
        2. Derive a model of functions/behaviors
        3. TR: Review functions/behaviors with customer and revise as required
    4. Isolate those elements of the technology to be implemented in software
    5. Research availability of existing software
    6. Define technical feasibility
    7. Make quick estimate of size
    8. Create a Scope Definition

Managing and Tracking Project Resources

Task network

- Individual tasks and subtasks have inter-dependencies based on their sequence.
- When more than one person is involved in a software engineering project, it is likely that development activities and tasks will be performed in parallel.
- When this occurs, concurrent tasks must be coordinated so that they will be complete when later tasks require their work products.

Managing and Tracking Project Resources

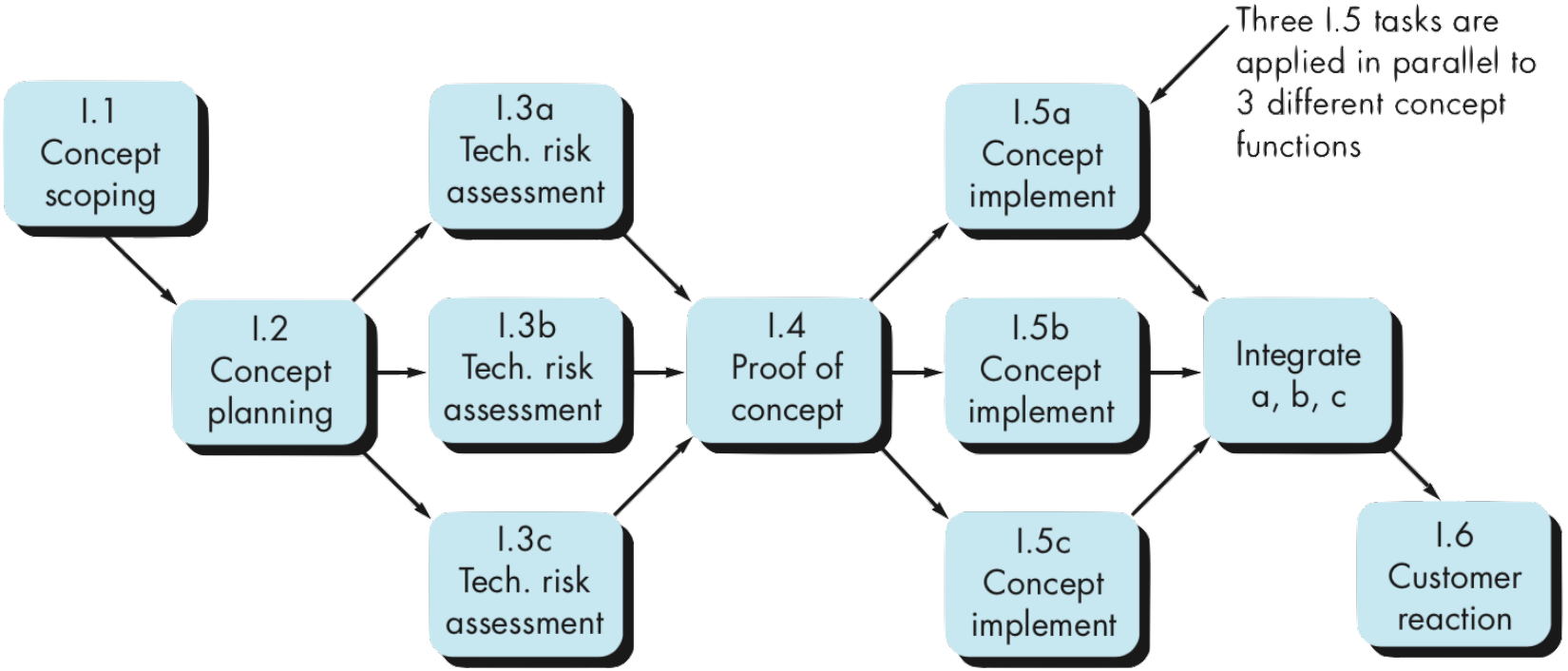
Task network

- A *task network*, also called an *activity network*, is a graphic representation of the task flow for a project.
- It is sometimes used as the mechanism through which task sequence and dependencies are input to an automated project scheduling tool.
- In its simplest form, used when creating a macroscopic schedule, the task network depicts major software engineering tasks.

Managing and Tracking Project Resources

Task network

- A schematic task network for a concept development project.



Managing and Tracking Project Resources

Task network

- The concurrent nature of software engineering activities leads to a number of important scheduling requirements.
- Parallel tasks occur asynchronously, so must determine inter-task dependencies to ensure continuous progress toward completion.
- Also, be aware of tasks that lie on the critical path. That is, tasks that must be completed on schedule for the project as a whole to be completed on schedule.
- Note that the task network shown is *macroscopic*. In a detailed task network (a precursor to a detailed schedule), each activity shown in the network would be expanded.
- For example, Task 1.1 would be expanded to show all tasks detailed in the refinement of Tasks 1.1.

Managing and Tracking Project Resources

Scheduling

- Scheduling a software project is much the same as any multi-task engineering effort, so generalized scheduling tools and techniques can be applied with little modification for software projects.
- **Program evaluation and review technique (PERT)** and the **critical path method (CPM)** are two project scheduling methods that can be applied to software development.
- Both techniques are driven by information already developed in earlier planning activities:
  - estimates of effort
  - decomposition of the product function
  - selection of the appropriate process model and task set
  - decomposition of the tasks that are selected

Managing and Tracking Project Resources

Scheduling

- Dependencies among tasks may be defined using a task network. Tasks, sometimes called the **work breakdown structure (WBS)**, are defined for the product as a whole or for individual functions.
- Both PERT and CPM provide quantitative tools that allow you to
  - determine the critical path—the chain of tasks that determines the duration of the project
  - establish “most likely” time estimates for individual tasks by applying statistical models
  - calculate “boundary times” that define a time “window” for a particular task.

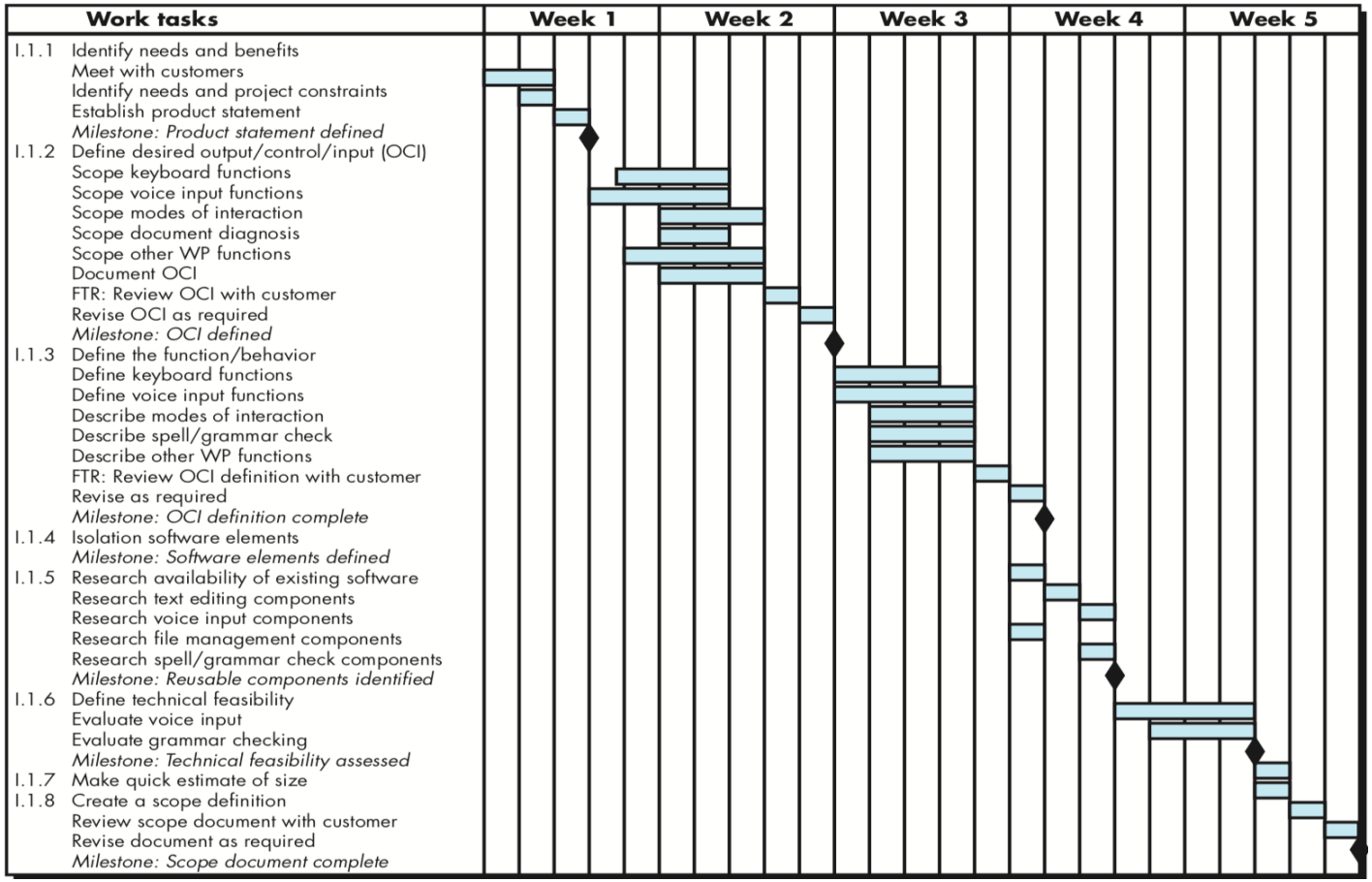
Managing and Tracking Project Resources

Timeline charts

- When creating a software project schedule, begin with a set of tasks (the work breakdown structure). If automated tools are used, the work breakdown is input as a task network or task outline.
- Effort, duration, and start date are then input for each task. and tasks may be assigned to specific individuals. Using this input, a time-line chart, also called a **Gantt chart**, is generated.
- A time-line chart can be developed for the entire project. Alternatively, separate charts can be developed for each project function or for each individual working on the project.
- The timeline chart shown on the next page depicts a part of a software project schedule that emphasizes the concept scoping task for a word-processing (WP) software product.

Managing and Tracking Project Resources

Timeline charts



Managing and Tracking Project Resources

Timeline charts

- All project tasks (for concept scoping) are listed in the left-hand column. The horizontal bars indicate the duration of each task.
- When multiple bars occur at the same time on the calendar, task concurrency is implied. The diamonds indicate milestones.
- Once the information necessary for the generation of a timeline chart has been input, the majority of software project scheduling tools produce **project tables.**
- These are tabular listing of all project tasks, planned and actual start and end dates, and a variety of related information. Used in conjunction with the time-line chart, project tables enable you to track progress

Managing and Tracking Project Resources

Timeline charts

- Here is part of a project table for the timeline chart shown earlier.

Work tasks	Planned start	Actual start	Planned complete	Actual complete	Assigned person	Effort allocated	Notes
I.1.1 Identify needs and benefits							Scoping will require more effort/time
Meet with customers	wk1, d1	wk1, d1	wk1, d2	wk1, d2	BLS	2 p-d	
Identify needs and project constraints	wk1, d2	wk1, d2	wk1, d2	wk1, d2	JPP	1 p-d	
Establish product statement	wk1, d3	wk1, d3	wk1, d3	wk1, d3	BLS/JPP	1 p-d	
Milestone: Product statement defined	wk1, d3	wk1, d3	wk1, d3	wk1, d3			
I.1.2 Define desired output/control/input (OCI)							
Scope keyboard functions	wk1, d4	wk1, d4	wk2, d2		BLS	1.5 p-d	
Scope voice input functions	wk1, d3	wk1, d3	wk2, d2		JPP	2 p-d	
Scope modes of interaction	wk2, d1		wk2, d3		MLL	1 p-d	
Scope document diagnostics	wk2, d1		wk2, d2		BLS	1.5 p-d	
Scope other WP functions	wk1, d4	wk1, d4	wk2, d3		JPP	2 p-d	
Document OCI	wk2, d1		wk2, d3		MLL	3 p-d	
FTR: Review OCI with customer	wk2, d3		wk2, d3		all	3 p-d	
Revise OCI as required	wk2, d4		wk2, d4		all	3 p-d	
Milestone: OCI defined	wk2, d5		wk2, d5				
I.1.3 Define the function/behavior							

Managing and Tracking Project Resources

Tracking a schedule

- A properly developed project schedule becomes a road map that defines the tasks and milestones to be tracked and controlled as the project proceeds. Tracking can be accomplished in a number of different ways:
  - Conducting periodic project status meetings in which each team member reports progress and problems.
  - Evaluating the results of all reviews conducted throughout the software engineering process.
  - Determining whether formal project milestones (the diamonds shown in the timeline chart) have been accomplished by the scheduled date.
  - Comparing the actual start date to the planned start date for each project task listed in the resource table.
  - Meeting informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.
- All of these tracking techniques are used by experienced project managers.

Managing and Tracking Project Resources

Tracking a schedule

- When faced with severe deadline pressure, project managers can use a technique called time-boxing. It recognizes that the complete product may not be deliverable by the predefined deadline.
- An incremental software paradigm is chosen, and a schedule is derived for each incremental delivery. The tasks associated with each increment are then time-boxed.
- This means that the schedule for each task is adjusted by working backward from the delivery date for the increment.

Managing and Tracking Project Resources

Tracking a schedule

- A “box” is put around each task. When a task hits the boundary of its time box (± 10 percent), work stops and the next task begins.
- By the time the time-box boundary is encountered, it is likely that 90 percent of the task has been completed. The remaining 10 percent, although important, can
  - be delayed until the next increment
  - be completed later if required
- Rather than becoming “stuck” on a task, the project proceeds toward the delivery date.

Managing and Tracking Project Resources

Earned value analysis

- We have already discussed qualitative approaches to project tracking. Each provides an indication of progress, but an assessment of the information provided is somewhat subjective.
- There is also a quantitative technique for assessing progress as the team progresses through the work on tasks allocated to the project schedule. It is known as *earned value analysis (EVA)*.

Managing and Tracking Project Resources

Earned value analysis

- Earned value analysis provides a common value scale for every project task, regardless of the type of work being performed.
- The total hours to do the whole project are estimated, and every task is given an earned value based on its estimated percentage of the total.
- Earned value is a measure of progress. It enables you to assess the "percent of completeness" of a project using quantitative analysis rather than rely on a gut feeling.
- It can provide accurate and reliable readings of performance from as early as 15 percent into the project.

Managing and Tracking Project Resources

Earned value analysis

- To determine the earned value, the following steps are performed:
  - The **budgeted cost of work scheduled (BCWS)** is determined for each work task represented in the schedule.
    - During estimation, the work (in person-hours or person-days) of each software engineering task is planned. Hence, $BCWS_i$ is the effort planned for work task *i*.
    - To determine progress at a point along the project schedule, the value of BCWS is the sum of the $BCWS_i$ values for all work tasks that should have been completed by that point in time on the schedule

Managing and Tracking Project Resources

Earned value analysis

- To determine the earned value, the following steps are performed:
  - The BCWS values for all work tasks are summed to derive the **budget at completion (BAC)**. Hence,
  - BAC = Σ ($BCWS_k$) for all tasks *k*

Managing and Tracking Project Resources

Earned value analysis

- To determine the earned value, the following steps are performed:
  - Next, the value for **budgeted cost of work performed (BCWP)** is computed. The value for BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.

Managing and Tracking Project Resources

Earned value analysis

- The distinction between the BCWS and the BCWP is that the former represents the budget of the activities that were planned to be completed
- The latter represents the budget of the activities that actually were completed.” Given values for BCWS, BAC, and BCWP, important progress indicators can be computed:
  - Schedule performance index, SPI = BCWP/BCWS
  - Schedule variance, SV = BCWP – BCWS
- SPI is an indication of the efficiency with which the project is utilizing scheduled resources.
- An SPI value close to 1.0 indicates efficient execution of the project schedule. SV is simply an absolute indication of variance from the planned schedule.

Managing and Tracking Project Resources

Earned value analysis

- The percent scheduled for completion is given as:
  - Percent scheduled for completion = BCWS / BAC
- This provides an indication of the percentage of work that should have been completed by time *t*.
- The percent complete is given as:
  - Percent complete = BCWP/BAC
- This provides a quantitative indication of percent of completeness of the project at a given point in time *t*.

Managing and Tracking Project Resources

Earned value analysis

- It is also possible to compute the actual cost of work performed (ACWP).
- The value for ACWP is the sum of the effort actually expended on work tasks that have been completed by a point in time on the project schedule. It is then possible to compute
  - Cost performance index, CPI = BCWP / ACWP
  - Cost variance, CV = BCWP - ACWP
- A CPI value close to 1.0 provides a strong indication that the project is within its defined budget. CV is an absolute indication of cost savings (against planned costs) or shortfall at a stage of a project.

Managing and Tracking Project Resources

Earned value analysis

- The major benefit of earned value analysis is that it illuminates scheduling difficulties before they might otherwise be apparent.
- This enables you to take corrective action before a project crisis develops.