This may involve renegotiating the project constraints and deliverables with the customer. A new schedule of when work should be completed also has to be established and agreed with the customer.

If this renegotiation is unsuccessful or the risk mitigation actions are ineffective, then you should arrange for a formal project technical review. The objectives of this review are to find an alternative approach that will allow the project to continue, and to check whether the project and the goals of the customer and software developer are still aligned.

The outcome of a review may be a decision to cancel a project. This may be a result of technical or managerial failings but, more often, is a consequence of external changes that affect the project. The development time for a large software project is often several years. During that time, the business objectives and priorities inevitably change. These changes may mean that the software is no longer required or that the original project requirements are inappropriate. Management may then decide to stop software development or to make major changes to the project to reflect the changes in the organizational objectives.

23.3 Project scheduling

Project scheduling is the process of deciding how the work in a project will be organized as separate tasks, and when and how these tasks will be executed. You estimate the calendar time needed to complete each task, the effort required, and who will work on the tasks that have been identified. You also have to estimate the resources needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be. In terms of the planning stages that I discussed in the introduction of this chapter, an initial project schedule is usually created during the project startup phase. This schedule is then refined and modified during development planning.

Both plan-based and agile processes need an initial project schedule, although the level of detail may be less in an agile project plan. This initial schedule is used to plan how people will be allocated to projects and to check the progress of the project against its contractual commitments. In traditional development processes, the complete schedule is initially developed and then modified as the project progresses. In agile processes, there has to be an overall schedule that identifies when the major phases of the project will be completed. An iterative approach to scheduling is then used to plan each phase.

Scheduling in plan-driven projects (Figure 23.4) involves breaking down the total work involved in a project into separate tasks and estimating the time required to complete each task. Tasks should normally last at least a week, and no longer than 2 months. Finer subdivision means that a disproportionate amount of time must be spent on replanning and updating the project plan. The maximum amount of time for

Activity charts

An activity chart is a project schedule representation that shows which tasks can be carried out in parallel and those that must be executed in sequence, due to their dependencies on earlier activities. If a task is dependent on several other tasks then all of these must finish before it can start. The 'critical path' through the activity chart is the longest sequence of dependent tasks. This defines the project duration.

http://www.SoftwareEngineering-9.com/Web/Planning/activities.html

any task should be around 8 to 10 weeks. If it takes longer than this, the task should be subdivided for project planning and scheduling.

Some of these tasks are carried out in parallel, with different people working on different components of the system. You have to coordinate these parallel tasks and organize the work so that the workforce is used optimally and you don't introduce unnecessary dependencies between the tasks. It is important to avoid a situation where the whole project is delayed because a critical task is unfinished.

If a project is technically advanced, initial estimates will almost certainly be optimistic even when you try to consider all eventualities. In this respect, software scheduling is no different from scheduling any other type of large advanced project. New aircraft, bridges, and even new models of cars are frequently late because of unanticipated problems. Schedules, therefore, must be continually updated as better progress information becomes available. If the project being scheduled is similar to a previous project, previous estimates may be reused. However, projects may use different design methods and implementation languages, so experience from previous projects may not be applicable in the planning of a new project.

As I have already suggested, when you are estimating schedules, you must take into account the possibility that things will go wrong. People working on a project may fall ill or leave, hardware may fail, and essential support software or hardware may be delivered late. If the project is new and technically advanced, parts of it may turn out to be more difficult and take longer than originally anticipated.

A good rule of thumb is to estimate as if nothing will go wrong, then increase your estimate to cover anticipated problems. A further contingency factor to cover unanticipated problems may also be added to the estimate. This extra contingency factor depends on the type of project, the process parameters (deadline, standards, etc.), and the quality and experience of the software engineers working on the project. Contingency estimates may add 30% to 50% to the effort and time required for the project.

23.3.1 Schedule representation

Project schedules may simply be represented in a table or spreadsheet showing the tasks, effort, expected duration, and task dependencies (Figure 23.5). However, this style of representation makes it difficult to see the relationships and dependencies

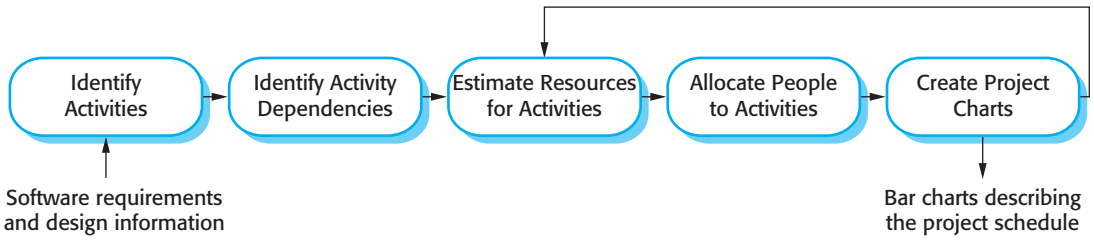

Figure 23.4 The project scheduling process

between the different activities. For this reason, alternative graphical representations of project schedules have been developed that are often easier to read and understand. There are two types of representation that are commonly used:

1. Bar charts, which are calendar-based, show who is responsible for each activity, the expected elapsed time, and when the activity is scheduled to begin and end. Bar charts are sometimes called 'Gantt charts', after their inventor, Henry Gantt.
2. Activity networks, which are network diagrams, show the dependencies between the different activities making up a project.

Normally, a project planning tool is used to manage project schedule information. These tools usually expect you to input project information into a table and will then create a database of project information. Bar charts and activity charts can then be generated automatically from this database.

Project activities are the basic planning element. Each activity has:

1. A duration in calendar days or months.
2. An effort estimate, which reflects the number of person-days or person-months to complete the work.
3. A deadline by which the activity should be completed.
4. A defined endpoint. This represents the tangible result of completing the activity. This could be a document, the holding of a review meeting, the successful execution of all tests, etc.

When planning a project, you should also define milestones; that is, each stage in the project where a progress assessment can be made. Each milestone should be documented by a short report that summarizes the progress made and the work done. Milestones may be associated with a single task or with groups of related activities. For example, in Figure 23.5, milestone M1 is associated with task T1 and milestone M3 is associated with a pair of tasks, T2 and T4.

A special kind of milestone is the production of a project deliverable. A deliverable is a work product that is delivered to the customer. It is the outcome of a significant project phase such as specification or design. Usually, the deliverables that are

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Figure 23.5 Tasks, durations, and dependencies

required are specified in the project contract and the customer's view of the project's progress depends on these deliverables.

To illustrate how bar charts are used, I have created a hypothetical set of tasks as shown in Figure 23.5. This table shows tasks, estimated effort, duration, and task interdependencies. From Figure 23.5, you can see that task T3 is dependent on task T1. Task T1 must, therefore, be completed before T3 starts. For example, T1 might be the preparation of a component design and T3, the implementation of that design. Before implementation starts, the design should be complete. Notice that the estimated duration for some tasks is more than the effort required and vice versa. If the effort is less than the duration, this means that the people allocated to that task are not working full-time on it. If the effort exceeds the duration, this means that several team members are working on the task at the same time.

Figure 23.6 takes the information in Figure 23.5 and presents the project schedule in a graphical format. It is a bar chart showing a project calendar and the start and finish dates of tasks. Reading from left to right, the bar chart clearly shows when tasks start and end. The milestones (M1, M2, etc.) are also shown on the bar chart. Notice that tasks that are independent are carried out in parallel (e.g., tasks T1, T2, and T4 all start at the beginning of the project).

As well as planning the delivery schedule for the software, project managers have to allocate resources to tasks. The key resource is, of course, the software engineers who will do the work, and they have to be assigned to project activities. The resource allocation can also be input to project management tools and a bar chart generated, which shows when staff are working on the project (Figure 23.7). People may be

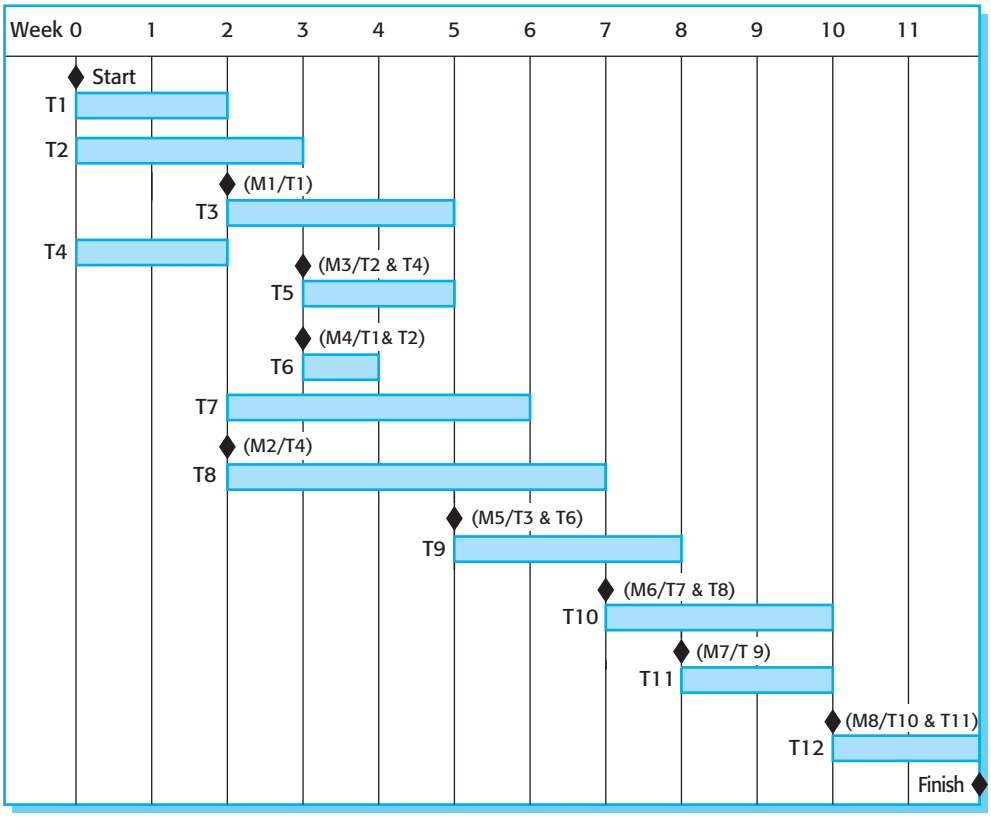

Figure 23.6
Activity bar chart

working on more than one task at the same time and, sometimes, they are not working on the project. They may be on holiday, working on other projects, attending training courses, or engaging in some other activity. I show part-time assignments using a diagonal line crossing the bar.

Large organizations usually employ a number of specialists who work on a project when needed. In Figure 23.7, you can see that Mary is a specialist, who works on only a single task in the project. This can cause scheduling problems. If one project is delayed while a specialist is working on it, this may have a knock-on effect on other projects where the specialist is also required. These may then be delayed because the specialist is not available.

If a task is delayed, this can obviously affect later tasks that are dependent on it. They cannot start until the delayed task is completed. Delays can cause serious problems with staff allocation, especially when people are working on several projects at the same time. If a task (T) is delayed, the people allocated may be assigned to other work (W). To complete this may take longer than the delay but, once assigned, they cannot simply be reassigned back to the original task, T. This may then lead to further delays in T as they complete W.

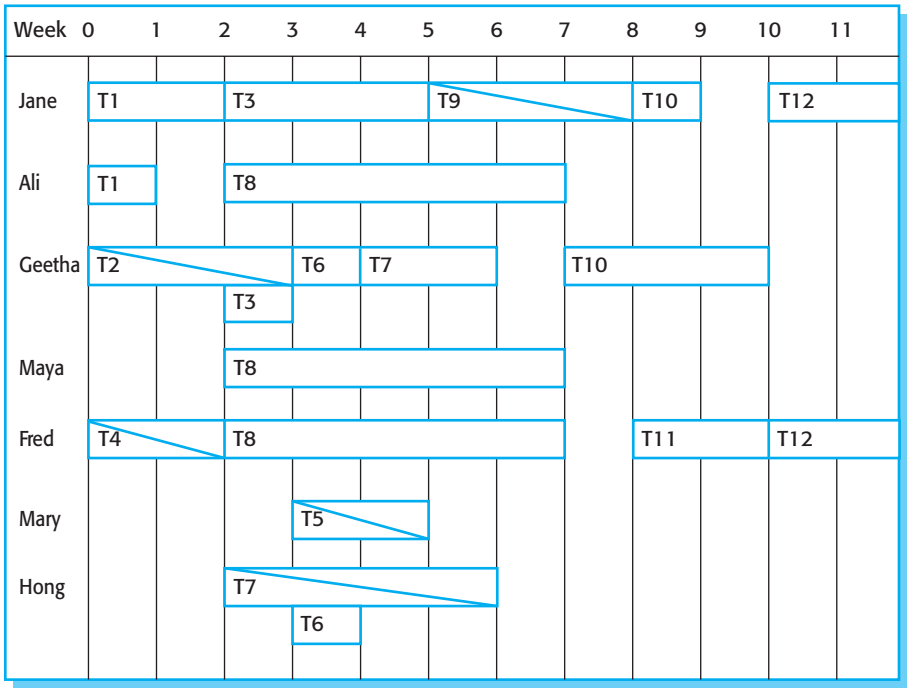

Figure 23.7 Staff allocation chart

23.4 Agile planning

Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments. Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development. The decision on what to include in an increment depends on progress and on the customer's priorities. The argument for this approach is that the customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes. Cohn's book {Cohn, 2005 ##1735} is a comprehensive discussion of planning issues in agile projects.

The most commonly used agile approaches such as Scrum (Schwaber, 2004) and extreme programming (Beck, 2000) have a two-stage approach to planning, corresponding to the startup phase in plan-driven development and development planning:

1. Release planning, which looks ahead for several months and decides on the features that should be included in a release of a system.
2. Iteration planning, which has a shorter-term outlook, and focuses on planning the next increment of a system. This is typically 2 to 4 weeks of work for the team.