Timeous Technetronic | Making Business Smarter

Software Project Planning Practices

1. Vision and Scope Document
2. Software Project Plan
3. The Process-Estimation
4. Project Schedule
5. Risk Plan
6. Establish a Change Control Board

1. Vision and Scope Document

Library

When the project begins, the project manager has a unique role to play. The start of the project is the time when the scope of the project is defined; only the project manager is equipped to make sure that it's defined properly. Everyone else has a role to play later on: users and stakeholders will provide expertise, requirements analysts will write specifications, programmers will build the code, etc. Everyone involved in the project has some input into the scope, but only the project manager is solely dedicated to it. Defining the scope is the most productive thing a project manager can do to get the project underway. The Vision and Scope document is the project manager's tool for doing that.

The vision and scope document is one of the most important tools that a project manager has; it is also one of the easiest to implement. A good vision and scope document will help a project avoid some of the costliest problems that a project can face. By writing the document and circulating it among everyone involved in the project, the project manager can ensure that each of the stakeholders and engineers share a common understanding of the needs being addressed—and that the software must address them.

Vision and Scope Document Outline

1.	Problem Statement
a)	Project background
b)	Stakeholders
c)	Users
d)	Risks
e)	Assumptions
2.	Vision of the Solution
a)	Vision statement
b)	List of features
c)	Scope of phased release (optional)
d)	Features that will not be developed

Project background

This section contains a summary of the problem that the project will solve. It should provide a brief history of the problem and an explanation of how the organization justified the decision to build software to address it. This section should cover the reasons that the problem exists, the organization's history with this problem, any previous projects which were undertaken to try to address it, and the way that the decision to begin this project was reached.

Stakeholders

This is a bulleted list of the stakeholders. Each stakeholder may be referred to by name or by title or role ("support group manager", "CTO", "senior manager"). The needs of each stakeholder are described in a few sentences.

Users

This is a bulleted list of the users. As with the stakeholders, each user can either be referred to by name or role ("support rep", "call quality auditor", "home website user")—however, if there are many users, it is usually inefficient to try to name each one. The needs of each user are described.

Risks

This section lists any potential risks to the project (see Figure 2-1 for examples). It should be generated by a project team's brainstorming session. It could include external factors that could impact the project, issues or problems that could potentially cause project delays or raise issues. (The process for assessing and mitigating risk below can be used to generate the risks for this section.)

Assumptions

This is the list of assumptions that the stakeholders, users or project team have made. Often, these assumptions are generated during an estimation session (see Chapter 3). If the same process is being used, the rest of the vision and scope document should be ready before the meeting and used as the basis for estimation. The assumptions generated during the estimation kickoff meeting should then be reviewed, and any non-technical assumptions should be copied into this section. (Technical assumptions—meaning assumptions that affect the design and development but not the scope of the project—should not be included in this document. The estimate results will still contain a record of these assumptions, but they are not useful for this particular audience.)

If process is not being used to generate the assumptions, the project manager should hold a brainstorming session with the team to come up with a list of assumptions instead. (See Chapter 3 for more information on assumptions.)

Vision statement

The goal of the vision statement is to describe what the project is expected to accomplish. It should explain what the purpose of the project is. This should be a compelling reason, a solid justification for spending time, money and resources on the project. The best time to write the vision statement is after talking to the stakeholders and users and writing down their needs; by this time, a concrete understanding of the project should be starting to jell.

List of features

This section contains a list of features. A feature is as a cohesive area of the software that fulfils a specific need by providing a set of services or capabilities. Any software package—in fact, any engineered product—can be broken down into features. The project manager can choose the number of features in the vision and scope document by changing the level of detail or granularity of each feature, and by combining multiple features into a single one. Sometimes those features are small ("screw-top cap", "holds one litre of liquid"); sometimes they are big ("four wheel drive", "seats seven passengers"). It is useful to describe a product in about ten features in the vision and scope document, because this usually yields a level of complexity that most people reading it are comfortable with. Adding too many features will overwhelm most readers, and each extra feature adds time and effort to the project and makes it difficult or unrealistic to accomplish. Each feature should be listed in a separate paragraph or bullet point. It should be given a name, followed by a description of the functionality that it provides. This description does not need to be detailed; it can simply

be a few sentences that give a general explanation of the feature. However, if there is more information that a stakeholder or project team member feels should be included, it is important to include that information. For example, it is sometimes useful to include a use case in a specific feature (see Chapter 6), as long as it is written in such a way that all of the stakeholders can read and understand it.

Scope of phased release (Optional)

Sometimes software projects are released in phases: a version of the software with some subset of the features is released first, and a newer, more complete version is released later. This section describes the plan for a phased release, if that approach is to be taken.
This is useful when there is an important deadline for the software, but developing the entire software project by that deadline would be unrealistic. The most common way to compromise on this release date is to divide the features into two or more releases. In that case, this section should identify specifically when those versions will be released, and which features will be included in each version. It's reasonable to divide one feature up between two releases, as long as it is made clear exactly how that will happen.
If a project manager needs to release a project in phases, it is critical that the project team be consulted. Some features are much more difficult to divide than others, and the engineers might see dependencies between features that are not clear to the stakeholders and project manager. After the phased release plan is written down and agreed upon, the project team should always be asked to re-estimate the effort and a new project plan should be generated (see below). This will ensure that the phased release is feasible and compatible with the company's priorities.

Features that will not be developed

Features are often left out of a project on purpose. When a feature is explicitly left out of the software, it should be added to this section to tell the reader that a decision was made to exclude it. For example, one way to handle an unrealistic deadline is by removing one or more features from the software, in which case the removed features should be moved into this section. The reason these features should be moved rather than deleted from the document is that otherwise readers might assume that they were overlooked and bring the up in a review. This is especially important during the review of the document because it allows everyone to agree on the exclusion of the feature (or object to it).

2. Software Project Plan

The project plan is used by many people in the organization. The project manager uses it to communicate the project's status to the stakeholders and senior managers, and to plan the team's activities. The team members use it to understand the context for the work they are doing. The senior managers use it to verify that the project's cost and schedule are reasonable and under control, and that the project is being done in an efficient and cost-effective manner. The stakeholders use it to make sure that the project is on track, and that their needs are being addressed.

The project plan defines the work that will be done on the project and who will do it. A typical project plan consists of:

A *statement of work* (SOW) that describes all work products (specifications, test plans, code, defect reports and any other product of work performed over the course of the project) that will be produced and a list of people who will perform that work
A *resource list* that contains a list of all resources that will be needed for the product and their availability

A *work breakdown structure* (WBS) and a set of effort estimates

A project schedule
A risk plan that identifies any risks that might be encountered and indicates how those risks would be handled should they occur

Statement of Work

The SOW is included as part of the project plan, but it should It should be a separate document that can stand on its own. It should contain each of the following:

The list of features being developed. If the software is being released in phases, the features should be divided into those phases as well.

A description of the intermediate deliverable or work product that will be built.

The estimated effort involved for each work product to be delivered (possibly based on the results of the estimation session), if known.

Resource List

The project plan should contain a list of all resources that will be used on the project. This list should go beyond what's covered by the project schedule by including a description of each resource, as well as any limits on that resource's availability.

Most project management software packages provide a feature to maintain a resource list. If this is not available, the resource list can either be a spreadsheet or a word processor document containing a simple list, with one line per resource. The list should give each resource a name, a brief one-line description, and list the availability and cost (if applicable) of the resource. All resources should be handled in the same way, regardless of type.
Estimates and Project Schedule

Once the statement of work and the resource list have been created, the project manager should build a project schedule. This is usually done in several steps:

A work breakdown structure (WBS) is defined. This is a list of tasks which, if performed, will generate all of the work products needed to build the software.

estimate of the effort required for each task in the WBS is generated.

A project schedule is created by assigning resources and determining the calendar time required for each task.

The project plan should include the complete revision history of the WBS—it should contain a list of any tasks that are added, changed or removed, and when those changes occurred. It should also include estimates and project schedule, including any revisions that were made during the review meetings. (Chapter 3 contains a repeatable process for generating a WBS and estimates. Chapter 4 describes how to create a project schedule.)

Risk Plan

A risk plan is a list of all risks that threaten the project, along with a plan to mitigate some or all of those risks. Some people say that uncertainty is the enemy of planning. If there were no uncertainty, then every project plan would be accurate and every project would go off without a hitch. Unfortunately, real life intervenes, usually at the most inconvenient times. The risk plan is an insurance policy against uncertainty.

3. Process-Estimation

To someone who has never estimated a project in a structured way, estimation seems little more than attempting to predict the future. This view is reinforced when off-the-cuff estimates are inaccurate and projects come in late. But a good formal estimation process, one that allows the project team to reach a consensus on the estimates, can improve the accuracy of those estimates, making it much more likely that projects will come in on time. This is an estimation process that is straightforward to implement. Using it, a project manager can help the team to create successful estimates for any software project by using sound techniques and understanding what makes estimates more accurate.

The Original estimation method was developed in the 1940s at the Rand Corporation as a forecasting tool. It has since been adapted across many industries to estimate many kinds of tasks, ranging from statistical data collection results to sales and marketing forecasts. It has proven to be a very effective estimation tool, and it lends itself well to software projects. The repeatable process presented here was developed in the 1990s by Mary Sakry and Neil Potter.

To use this process, the project manager selects a moderator and an estimation team with three to seven members. This process consists of two meetings run by the moderator. The first meeting is the kickoff meeting, during which the estimation team creates a WBS (Work Breakdown Structure) and discusses assumptions. After the meeting, each team member creates an effort estimate for each task. The second meeting is the estimation session, in which the team revises the estimates as a group and achieves consensus. After the estimation session, the project manager summarizes the results and reviews them with the team, at which point they are ready to be used as the basis for planning the software project.

Name	Internal Process Script
Purpose	A project team generates estimates and a work breakdown structure.
Summary	A repeatable process for estimation. Using it, a project team can generate a consensus on estimates for the completion of the project.
Work Products	***Input*** Vision and scope document, or other documentation that defines the scope of the work product being estimated ***Output*** Work breakdown structure (WBS) Effort estimates for each of the tasks in the WBS
Entry Criteria	The following criteria should be met in order for our internal process to be effective: The vision and scope document (or other documentation that defines the scope of the work product being estimated) has been agreed to by the stakeholders, users, managers and engineering team. If no vision and scope document is available, there must be enough supporting documentation for the team to understand the work product. The kickoff meeting and estimation session have been scheduled, with at least two hours allowed for each. The project manager and the moderator agree on the goal of the estimation session by identifying the scope of the work to be estimated.
Basic Course of Events	1. **Choose the team.** The project manager selects the estimation team and a moderator. The team should consist of 3 to 7 project team members. The team should include representatives from every engineering group that will be involved in the development of the work product being estimated.

	2. **Kickoff meeting.** The moderator prepares the team and leads a discussion to brainstorm assumptions, generate a WBS and decide on the units of estimation. 3. **Individual preparation.** After the kickoff meeting, each team member individually generates the initial estimates for each task in the WBS, documenting any changes to the WBS and missing assumptions. 4. **Estimation session.** The moderator leads the team through a series of iterative steps to gain consensus on the estimates. At the start of the iteration, the moderator charts the estimates on the whiteboard so the estimators can see the range of estimates. The team resolves issues and revises estimates without revealing specific numbers. The cycle repeats until either no estimator wants to change his or her estimate, the estimators agree that the range is acceptable or two hours have elapsed. 5. **Assemble tasks.** The project manager works with the team to collect the estimates from the team members at the end of the meeting and compiles the final task list, estimates and assumptions. 6. **Review results.** The project manager reviews the final task list with the estimation team.
Alternative Paths	1. During step 1, if the team determines that there is not enough information known about the project to perform an estimate, the script ends. Before the script can be started again, the project manager must document the missing information by creating or modifying the vision and scope document (see Chapter 2). 2. During either step 1 or 3, if the team determines that there are outstanding issues that must be resolved before the estimate can be made, they agree upon a plan to resolve the issues and the script ends.
Exit Criteria	The script ends after the team has either generated a set of estimates or has agreed upon a plan to resolve the outstanding issues.

Kickoff Meeting

The moderator leads the meeting, which consists of the following activities:

The moderator explains the method to any new estimators.

If any team member has not yet read the vision and scope document and supporting documentation, the moderator reviews it with the team. (If this happens, the meeting should be expected to take an extra half-hour to hour.)

The moderator reviews the goal of the estimation session with the team, and checks that each team member is sufficiently knowledgeable to contribute.

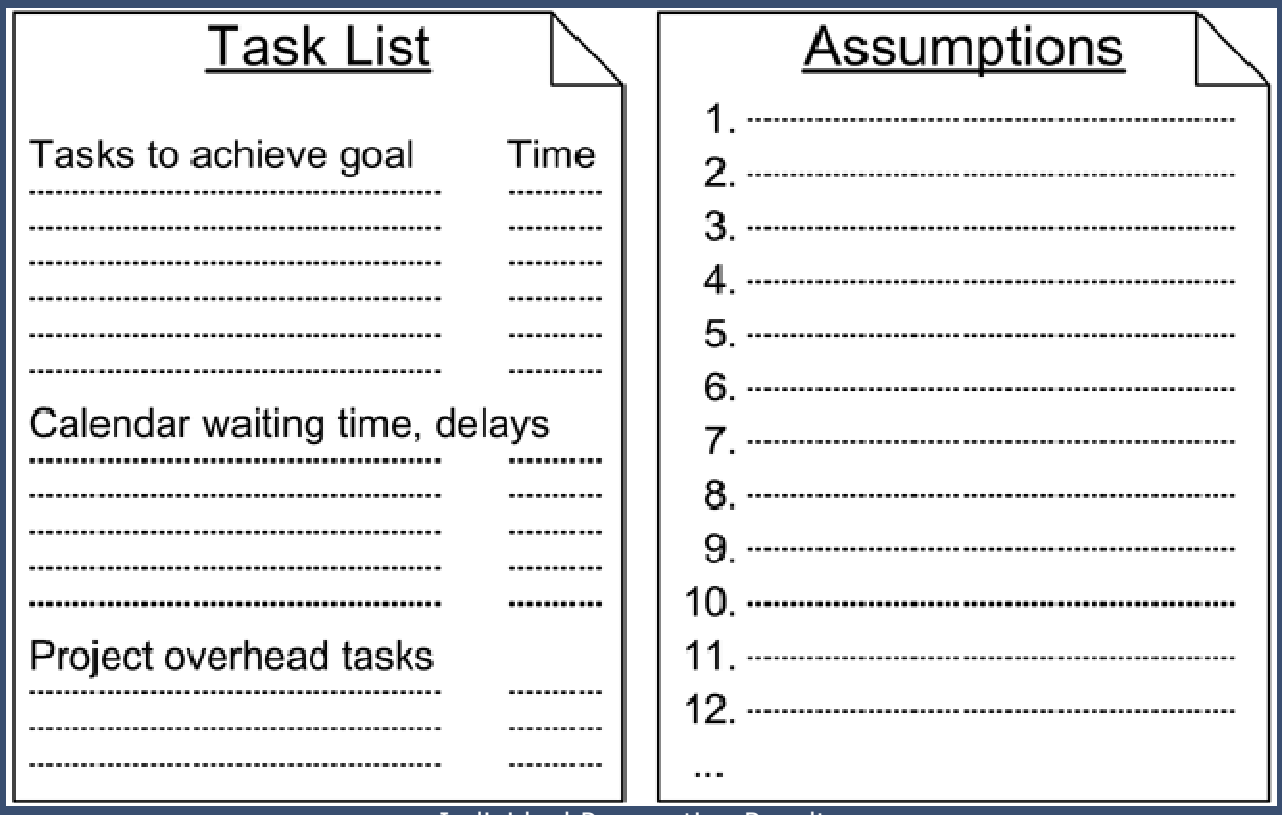
The team discusses the product being developed and brainstorms any assumptions.

The team generates a task list consisting of 10-20 major tasks. These tasks represent the top level of the work breakdown structure—additional detail can be generated later and/or discussed in the assumptions. This high-level task lists is the basis for the estimates that are going to be created.

The team agrees on the units of estimation (days, weeks, pages, classes, etc.).

Individual Preparation

After the kickoff meeting, the moderator writes down all of the assumptions and tasks that were generated by the team during the kickoff meeting and distributes them to the estimation team. Each team member independently generates a set of preparation results, a document which contains an estimate for each of the tasks, any assumptions that the team member made in order to create the estimates and any additional tasks which should be included in the WBS but which the team missed during the kickoff meeting. (The figure below shows the format of the individual preparation results.) Each team member builds preparation results by first filling in the tasks, and then estimating the effort for each task. An estimate for each task should be added to the "Tasks to achieve goal" section of the preparation results; the "Time" column should contain the estimate for each task.



Individual Preparation Results

Estimation session

The estimation session starts with each estimator filling out an estimation form. Blank estimation forms should be handed out to meeting participants, who fill in the tasks and their initial estimates from their individual preparations. During the estimation session, the team members will use these estimation forms to modify their estimates. After the estimation session, they will serve as a record of each team member's estimates for the individual tasks, which the project manager uses when compiling the results.

Name _____ Date _____ Estimation Form 1 / 1

Goal Statement To estimate the time to develop prototype for customers A & B Units days

Category ☑ goal tasks ☑ quality tasks ☐ waiting time ☐ project overhead

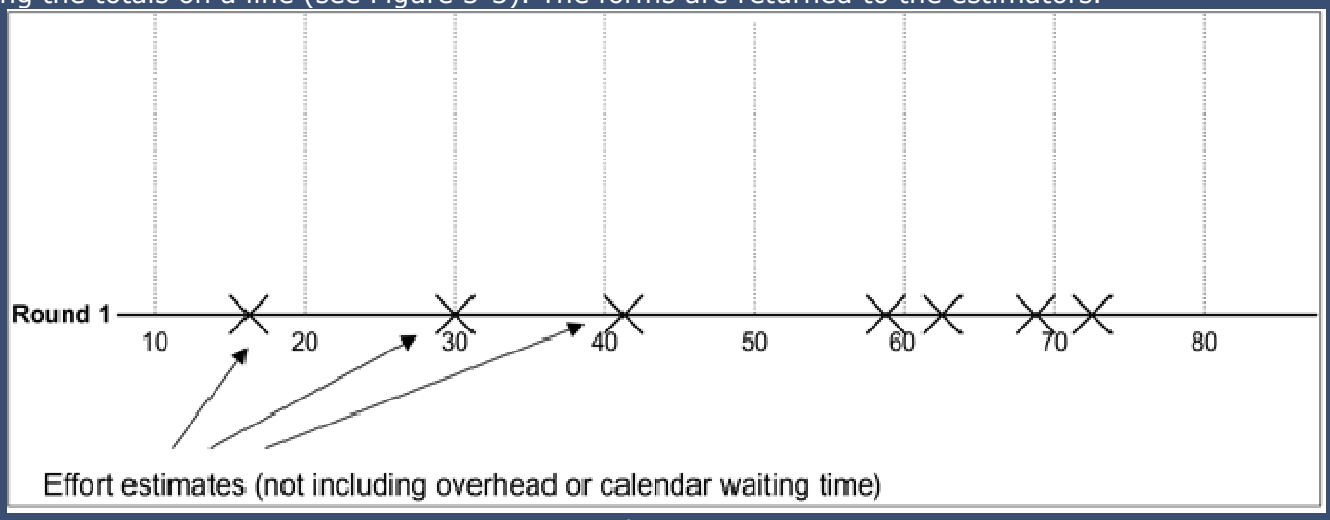
WBS# or Priority	Task Name	Est.	Delta 1	Delta 2	Delta 3	Delta 4	Total	Assumptions
1	Interview customers (A+B)	3	+2	+1				Needs offsite trip
2	Develop requirements docs	6	+5	-2	+1			Start from scratch
3	Inspect requirements docs	1	+2	+2	-2			Team of 4 BSAs
4	Do rework	1	+4					
5	Prototype design	20	-3	+4	-2			Includes DB
6	Test design	5	+3					20% exists now
	Delta		+13	+5	-3			
	Total	36	49	54	51			

Filled-in estimation form

(© The Process Group, copied with permission)

The moderator then leads the team through the estimation session:

1. The moderator collects all of the estimate forms. The estimates are tabulated on a whiteboard by plotting the totals on a line (see Figure 3-3). The forms are returned to the estimators.



Initial estimates

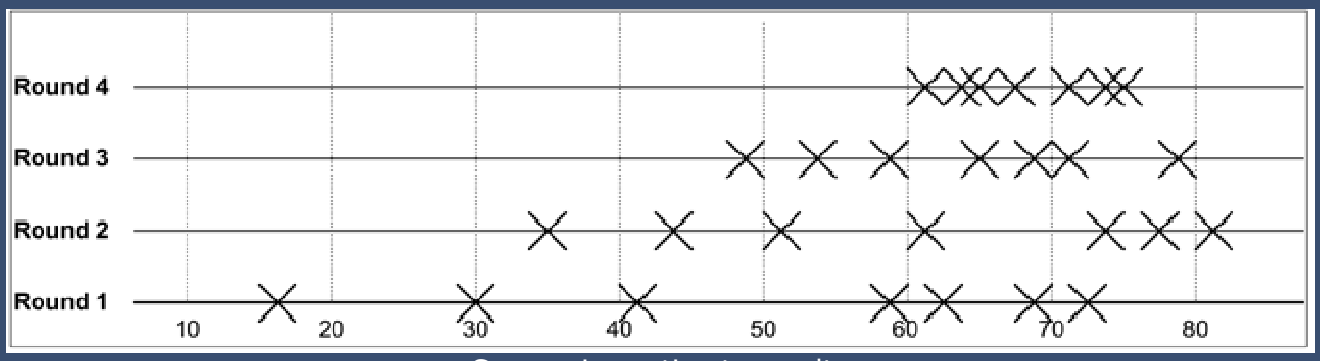
2. Each estimator reads out clarifications and changes to the task list written on the estimation form. Any new or changed tasks, discovered assumptions or questions are raised. Specific estimate times are not discussed.

3. The team resolves any issues or disagreements that are brought up. Since individual estimate times are not discussed, these disagreements are usually about the tasks themselves, and are often resolved by adding assumptions. When an issue is resolved, team members should write clarifications and changes to the task list on their estimation forms. This usually takes about 40 minutes for the first round, and 20

minutes for the following rounds.

4. The estimators all revise their individual estimates by filling in the next "Delta" column on their forms. (Using a delta allows the estimators to write "+4" or "-3" to add 4 or remove 3 from the estimate. They write the new total at the bottom of the sheet.)

This cycle repeats until either all estimators agree that the range is acceptable, the estimators feel they do not need to change their estimates, or two hours have elapsed.



Converging estimate results

4. Project Schedule

The project schedule is the core of the project plan. It is used by the project manager to commit people to the project and show the organization how the work will be performed. Schedules are used to communicate final deadlines and, in some cases, to determine resource needs. They are also used as a kind of checklist to make sure that every task necessary is performed. If a task is on the schedule, the team is committed to doing it. In other words, the project schedule is the means by which the project manager brings the team and the project under control.

The *project schedule* is a calendar that links the tasks to be done with the resources that will do them. Before a project schedule can be created, the project manager must have a work breakdown structure (WBS), an effort estimate for each task, and a resource list with availability for each resource. If these are not yet available, it may be possible to create something that looks like a schedule, but it will essentially be a work of fiction. A project manager's time is better spent on working with the team to create a WBS and estimates (using a consensus-driven estimation method) than on trying to build a project schedule without them. The reason for this is that a schedule itself is an estimate: each date in the schedule is estimated, and if those dates do not have the buy-in of the people who are going to do the work, the schedule will almost certainly be inaccurate.

There are many project scheduling software products which can do much of the tedious work of calculating the schedule automatically, and plenty of books and tutorials dedicated to teaching people how to use them. However, before a project manager can use these tools, he should understand the concepts behind the WBS, dependencies, resource allocation, critical paths, Gantt charts and earned value. These are the real keys to planning a successful project.

The most popular tool for creating a project schedule is Microsoft Project. There are also free and open source project scheduling tools available for most platforms which feature task lists, resource allocation, predecessors and Gantt charts.

Other project scheduling software packages include:

Open Workbench
dotProject

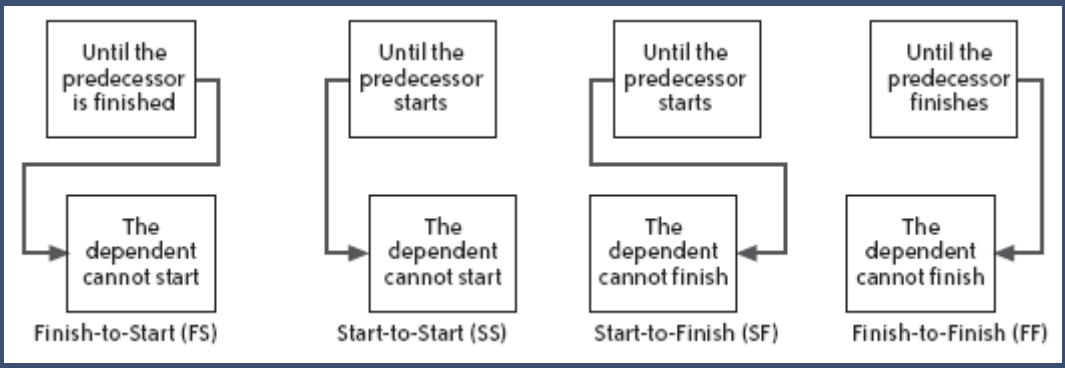
netOffice
TUTOS

Allocate Resources to the Tasks
The first step in building the project schedule is to identify the resources required to perform each of the tasks required to complete the project. A resource is any person, item, tool, or service that is needed by the project that is either scarce or has limited availability.

Many project managers use the terms "resource" and "person" interchangeably, but people are only one kind of resource. The project could include computer resources (like shared computer room, mainframe, or server time), locations (training rooms, temporary office space), services (like time from contractors, trainers, or a support team), and special equipment that will be temporarily acquired for the project. Most project schedules only plan for human resources—the other kinds of resources are listed in the resource list, which is part of the project plan.

One or more resources must be allocated to each task. To do this, the project manager must first assign the task to people who will perform it. For each task, the project manager must identify one or more people on the resource list capable of doing that task and assign it to them. Once a task is assigned, the team member who is performing it is not available for other tasks until the assigned task is completed. While some tasks can be assigned to any team member, most can be performed only by certain people. If those people are not available, the task must wait.

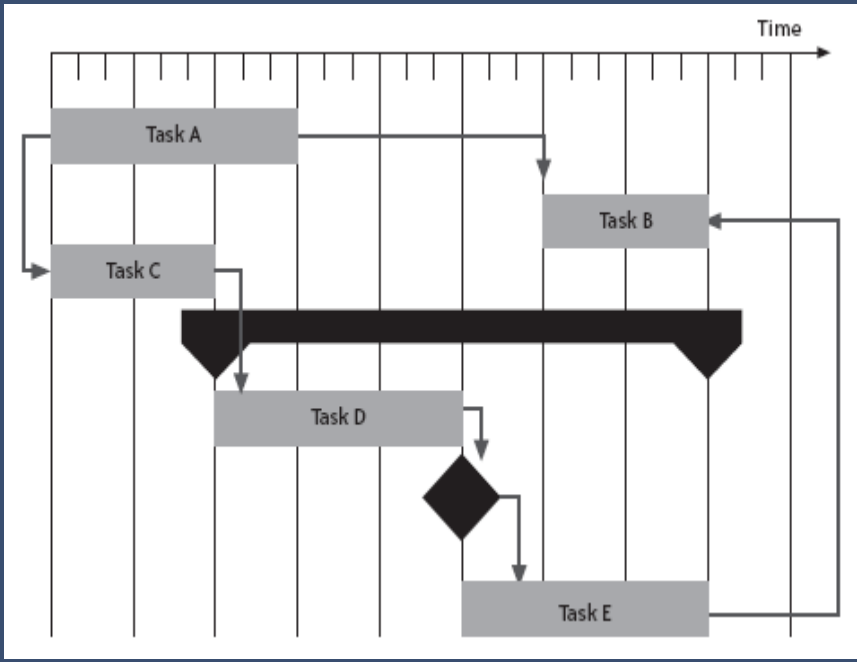
Identify Dependencies
Once resources are allocated, the next step in creating a project schedule is to identify dependencies between tasks. A task has a dependency if it involves an activity, resource, or work product that is subsequently required by another task. Dependencies come in many forms: a test plan can't be executed until a build of the software is delivered; code might depend on classes or modules built in earlier stages; a user interface can't be built until the design is reviewed. If a real process is used to generate estimates, many of these dependencies will already be represented in the assumptions. It is the project manager's responsibility to work with everyone on the engineering team to identify these dependencies. The project manager should start by taking the WBS and adding dependency information to it: each task in the WBS is given a number, and the number of any task that it is dependent on should be listed next to it as a predecessor. The following figure shows the four ways in which one task can be dependent on another.



Create the Schedule

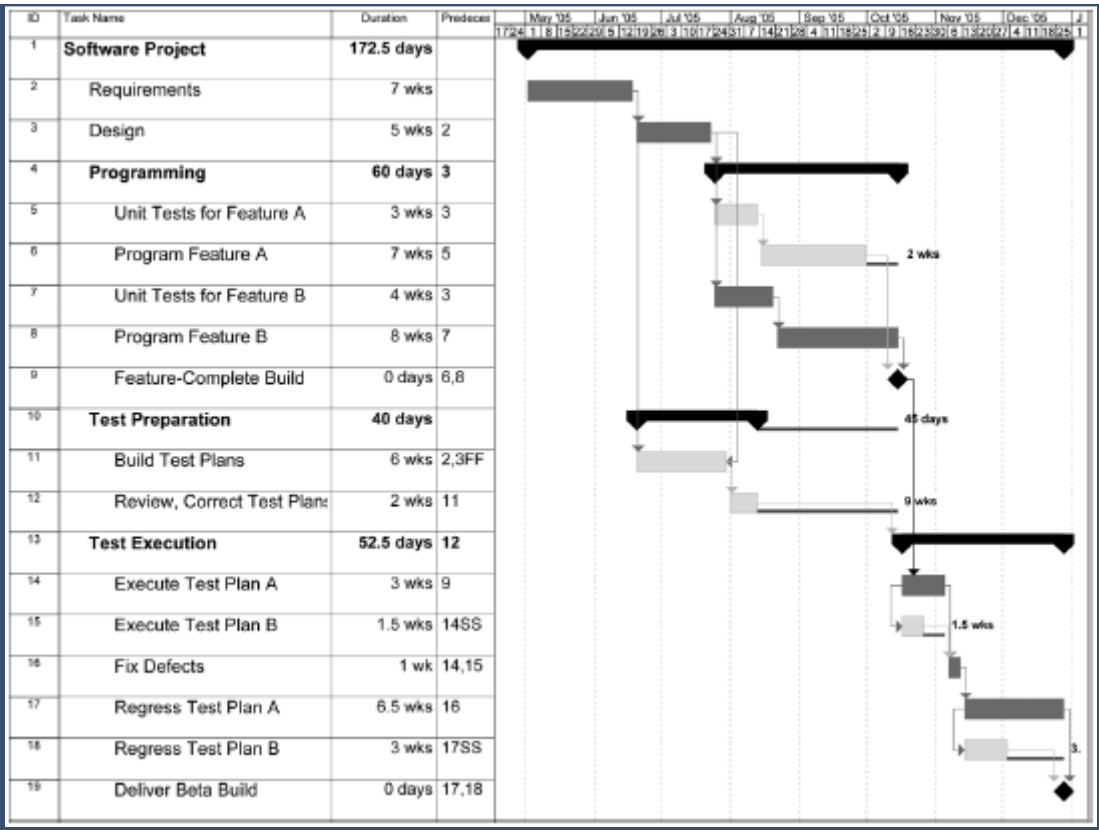
Once the resources and dependencies are assigned, the software will arrange the tasks to reflect the dependencies. The software also allows the project manager to enter effort and duration information for

each task; with this information, it can calculate a final date and build the schedule.
The most common form for the schedule to take is a Gantt chart. The following figure shows an example:



Each task is represented by a bar, and the dependencies between tasks are represented by arrows. Each arrow either points to the start or the end of the task, depending on the type of predecessor. The black diamond between tasks D and E is a milestone, or a task with no duration. Milestones are used to show important events in the schedule. The black bar above tasks D and E is a summary task, which shows that these tasks are two subtasks of the same parent task. Summary tasks can contain other summary tasks as subtasks. For example, if the team used an extra session to decompose a task in the original WBS into subtasks, the original task should be shown as a summary task with the results of the second estimation session as its subtasks.

The following figure shows an example of a Gantt chart created in Microsoft Project 2003:



ID	Task Name	Duration	Predecessors
1	**Software Project**	**172.5 days**	
2	Requirements	7 wks	
3	Design	5 wks	2
4	**Programming**	**60 days**	3
5	Unit Tests for Feature A	3 wks	3
6	Program Feature A	7 wks	5
7	Unit Tests for Feature B	4 wks	3
8	Program Feature B	8 wks	7
9	Feature-Complete Build	0 days	6,8
10	**Test Preparation**	**40 days**	
11	Build Test Plans	6 wks	2,3FF
12	Review, Correct Test Plans	2 wks	11
13	**Test Execution**	**52.5 days**	12
14	Execute Test Plan A	3 wks	9
15	Execute Test Plan B	1.5 wks	14SS
16	Fix Defects	1 wk	14,15
17	Regress Test Plan A	6.5 wks	16
18	Regress Test Plan B	3 wks	17SS
19	Deliver Beta Build	0 days	17,18

5. Risk Plan

Risk assessment is an important part of planning a software project because it allows the project manager to predict potential problems that will threaten the project and take steps to mitigate those problems. Adding a risk plan to a software project plan is an effective way to keep the project from being derailed by surprises or emergencies.

A *risk plan* is a list of all risks that threaten the project, along with a plan to mitigate some or all of those risks. Some people say that uncertainty is the enemy of planning. If there were no uncertainty, then every project plan would be accurate and every project would go off without a hitch. Unfortunately, real life intervenes, usually at the most inconvenient times. The risk plan is an insurance policy against uncertainty.

Name	Risk Planning Script
Purpose	To assess risks and create a risk plan.
Summary	The risk planning meeting happens in three parts: a brainstorming session to identify risks; a discussion where the probability and impact of each risk is estimated; and a discussion to identify actions that can mitigate risks. The end result is a risk management plan, which should be included verbatim in the final project plan.
Work Products	***Input*** Any project documentation that has been developed so far ***Output*** Risk plan Assumptions generated by the process

	Assumptions in the vision and scope document
Entry Criteria	The project manager has gathered the project team for a two-hour meeting to assess the project's risks
Basic Course of Events	1. **Brainstorm potential risks.** The project manager leads a brainstorming session to identify risks. Team members suggest every risk they can think of; the project manager writes the risks on a whiteboard as they come up. Brainstorming should be reminiscent of microwave popcorn: a few ideas should "pop" at first, followed by a large number being fired rapidly, slowing down to a final few "pops". The team will generally be able to judge when the risk identification is over. 2. **Estimate the impact of each risk.** The team assigns a number from 1 (highly unlikely) to 5 (very likely to occur) to represent the estimated probability of each risk. Similarly, impact should be estimated by assigning a number from 1 (for a risk with low impact) to 5 (for a risk which, if it occurs, will require an enormous effort to clean up). 3. **Build the risk plan.** The team identifies actions to be taken to mitigate high-priority risks and creates a risk plan that documents these actions.
Exit Criteria	The risk plan is finished.

Brainstorm potential risks

Risks should be as specific as possible. It's true that "The project might be delayed" or "We will go out of business" are risks; however, they are far too vague to do anything about. When a vague risk comes up, the project manager should prod the team into making it more specific. What are the possible sources of the project delay? How have past projects been delayed? For example, if the last project was late because a major stakeholder quit and was replaced by someone who disagreed with his predecessor's vision, the team should write that down as a risk. The assumptions documented in the vision and scope document and identified in a session are another good source of potential risks. The team should go through them and evaluate each assumption for potential risks as part of the risk brainstorming session.

Estimate the impact of each risk

Once the team has generated a final set of risks, they have enough information to estimate two things: a rough estimate of the probability that the risk will occur, and the potential impact of that risk on the project if it does eventually materialize. The risks must then be prioritized in two ways: in order of probability, and in order of impact. Both the probability and impact are measured using a relative scale by assigning each a number between 1 and 5.

These numbers are arbitrary; they are simply used to compare the probability or impact of one risk with another, and do not carry any specific meaning. The numbers for probability and impact are assigned to each risk; a priority can then be calculated by multiplying these numbers together. It is equally effective to assign a percentage as a probability (i.e. a risk is 80% likely to occur) and a real duration for impact (i.e. it will cost 32 man-hours if the risk occurs). However, many teams have trouble estimating these numbers, and find it easier to just assign an arbitrary value for comparison.
Many people have difficulty prioritizing, but there is a simple technique that makes it much easier. While it's difficult to rank all of the risks in the list at once, it is usually not hard to pick out the one that's most likely to occur. Assign that one a probability of 5. Then select the one that's least likely to occur and assign that one a probability of 1. With those chosen, it's much easier to rank the others relative to them. It might help to find another 5 and another 1, or if those don't exist, find a 4 and a 2. The rest of the probabilities should start to fall in place. Once that's done, the same can be done for the impact.

After the probability and impact of each risk have been estimated, the team can calculate the priority of each risk by multiplying its probability by its impact. This ensures that the highest priority is assigned to those risks that have both a high probability and impact, followed by either high-probability risks with a low impact or low-probability risks with a high impact. This is generally the order in which a good project manager will want to try to deal with them: it allows the most serious risks to rise to the top of the list.

Make a mitigation plan

All of this risk brainstorming and estimation is only useful if it leads to the team taking actions to avoid the most pressing risks. The remainder of the risk planning meeting should be dedicated to identifying these actions. The project manager should start with the highest-priority risk, working with the team to decide on any actions that should be taken. After that, the team should move down the list of risks, until they decide that the priority of each of the remaining risks is low enough that no action would be required.
The team can take any or all of these actions to mitigate a risk:

Alter the project plan. The project schedule can be adjusted to help reduce the risk. Riskier tasks can be moved earlier in the project, or given more time. This will give the team an early warning or a time cushion in case the risks materialize. The project manager can also hold an additional estimation session to break down the riskiest tasks into sub-tasks. More detailed planning will help reduce the risk.

Add additional tasks. There are certain actions that can be added to the schedule to help avoid risks. For example, if there is a high probability that a critical team member will leave the organization, cross-training tasks can be assigned to other people. This will increase total effort in the project, but it will be worth it if the team member leaves.

Plan for risks. For risks with a high impact that do not need specific tasks or project plan changes, the project manager should have the team spend a few minutes identifying the steps that should be taken in case the risk does occur. These do not need to be added to the project schedule, but they should be written down and added to the risk plan. This way, if the risk does occur, nobody will panic. Problems that have a large impact on the project can be demoralizing to the team and may throw the project into chaos. Simply having pre-planned the steps needed to fix the problem is highly reassuring; it keeps the team feeling like they are on track.

Once the mitigation steps are identified, all of these risks and actions should be documented in a risk plan. The easiest way to do that is to create a simple spreadsheet with five columns: Risk (one to three sentences which describe each risk), Probability (the estimated probability from 1 to 5), Impact (the estimated impact from 1 to 5), Priority (Probability × Impact) and Action (the specific actions that will be taken to mitigate the risk, or "None" if the risk is deemed a low enough priority to ignore).

Sample Risk Plan

Risk plan for project **ERP (Retail) Cum POS**

Assessment team members **Team Timeous**

Risk	Prob.	Impact	Priority	Actions

Sample risk plan (MS Excel version).

6. Establish a Change Control Board

Change control is method for implementing only changes that are worth pursuing, and for preventing unnecessary or overly costly changes from derailing the project. Change control is essentially an agreement between the project team and the managers that are responsible for decision-making on the project to evaluate the impact of a change before implementing it. Many changes that initially sound like good ideas will get thrown out once the true cost of the change is known. The potential benefit of the change is written down, and the project manager works with the team to estimate the potential impact that the change will have on the project. This gives the organization all of the information necessary to do a real cost-benefit analysis. If the benefit of the change is worth the cost, the project manager updates the plan to reflect the new estimates. Otherwise, the change is thrown out and the team continues with the original plan.

The most important part of change control is a change control board (CCB). There are certain people in the organization who have the power to change the scope of the project. Usually there is a senior manager or decision-maker who has the authority to make sweeping changes at will; sometimes there are several people in this position. For change control to be effective, these people must be part of the CCB.

In addition, the CCB should contain people from the project team:

The project manager

An important stakeholder or user (or someone who understands and advocates their perspective)

Someone who understands the effort involved in making the change (usually, this is a representative from the programming team)

Someone who understands the engineering decisions that the team makes over the course of the project (a design team member, requirements analyst or, if neither is available, a programmer who participated in the design of the software)

Someone who is familiar with the expected functionality of the software and with the behaviour being discussed for each individual change (typically a tester)

This last person fulfils a very important role in the change control process. Typically, she is involved in the tracking of changes and defects in the product. When a bug is reported, part of her job is to figure out

whether it is a defect (meaning that the software does not behave the way its specification requires it to behave) or a change (meaning that the software behaves as designed, but that this behaviour is not what the users or stakeholders need).

Change Control Process

The following script describes an effective change control process:

Name	**Change Control Process Script**
Purpose	To control changes by evaluating their impact before agreeing to implement them.
Summary	The change control process ensures that the impact of each change is evaluated before the decision is made to implement that change. A change is proposed by anyone evaluating the software. A change control board (CCB), made up of the decision-makers, project manager, stakeholder or user representatives, and selected team members, evaluates the change. The CCB either accepts or rejects the change.
Work Products	***Input*** Issue report in the defect tracking system that describes the change ***Output*** Modified issue report that reflects the impact of the change and the decision on whether or not to move forward with it
Entry Criteria	A change has been discovered, and an issue report that describes it has been entered into the defect tracking system.
Basic Course of Events	1. A CCB member (typically a tester) who is familiar with the expected functionality of the software reads and understands the issue report which describes the requested change. 2. The CCB member familiar with the change meets with the project manager to explain its scope and significance. Together, they identify all project team members who will be impacted by the change, and work with them to evaluate its impact. The project manager updates the issue report to reflect the result of that evaluation. 3. At the next CCB meeting, the project manager presents the scope and significance of the change, along with its expected impact. The CCB discusses the change, and performs a cost-benefit analysis to determine if its benefits are worth the cost. The CCB approves the change. 4. The project manager updates the issue report to indicate that the change has been approved, and then updates the project plan to reflect the change. The team members begin implementing the change.
Alternative Paths	1. In step 1, if the CCB member does not understand the change, it can be returned to the submitter for further explanation. The submitter may choose to either update the issue report to clarify the change (in which case the script returns to step 1) or drop it entirely (in which case the change control process ends). 2. In step 3, if the CCB determines that the benefits of the change are not worth the cost, they can reject it. The change control process ends, and no changes are made to the project. The project manager updates the issue report to reflect the fact that it was rejected.
Exit Criteria	The project plan has been updated to reflect the impact of the change, and work to implement the change has begun.