THE NEW COLLEGE (AUTONOMOUS)

Sponsored by: The Muslim Educational Association of Southern India

(Affiliated to the Madras University & Re-Accredited by NAAC with 'A' Grade)

Chennai-600014

SOFTWARE PROJECT MANAGEMENT

Subject Code: 17BRM514

Semester-V

Date: 12-07-2018

Batch: 2018-2019

Prepared By

Prof. N. ANVER HUSSAIN

Department of Information Systems Management, Shift-II

The New College (Autonomous)

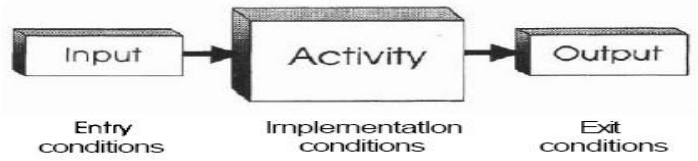
Chennai-600014

What Is Meant by a "Process"?

A process can be defined simply as a systematic approach that is performed to achieve a specific purpose. In the context of a software development process, a process is defined as an ordered set of activities that, after completed, results in a software "product" that is delivered to a customer.

Managing Software Development Projects

Each activity, of the many activities that comprise a software development process, has definable input and definable output as shown by the simple activity model in Figure.



Activity Model

The input to an activity, also called entry conditions, defines the conditions that must be satisfied before the activity can begin. Similarly, the output ofan activity, called exit conditions, defines the conditions that must be satisfied beforethe activity can be considered completed. Each activity can be further described byimplementation conditions; these are a procedure or set of steps that explains"what" must be done, "how" it will be done, and even "ways" to accomplish it.

An example of an activity is "performing a test" (for example, system test). The entry conditions might be:

- A completed and approved test plan is available.
- All test scripts are written.
- The prerequisite hardware and software are installed.
- Product code which has successfully exited the prior activity (i.e., code,inspection, unit test, etc.)

The implementation condition might be:

- Follow the approved test plan, such as running the test scripts, providingweekly status reports of both the problems being identified and the progress being made, and so on.

The exit conditions might be:

- All test scripts have run successfully.
- All problems have been resolved.

- ✓ A software development process should be made up of a comprehensive set of activities from which the members of a new project can select the right set.

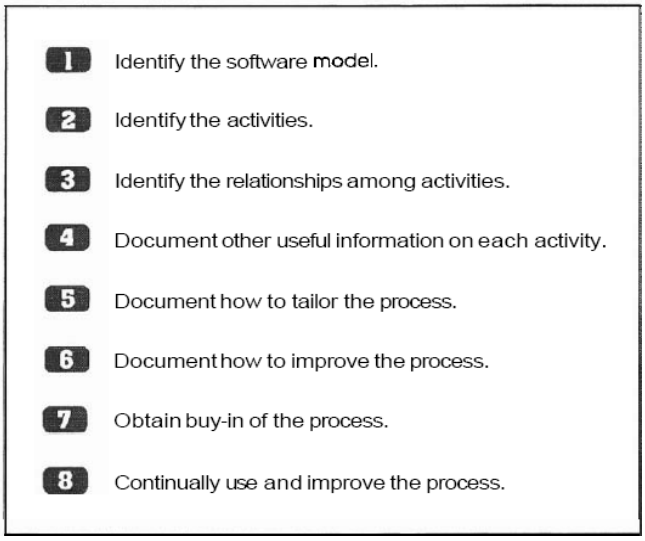
- ✓ A software development process must be examined routinely for improvement.

- ✓ A software development process must never require a user to perform a task that is not useful to the product or project.

- ✓ A software development process must have a simple, expedient method to allow its users to suggest improvements and to deviate.

The Steps to Defining a Software Development Process

Defining the right software development process for your organization will have a profound impact on controlling the schedules, costs, and quality of a project.



Steps to design a software development process

Step 1. Identify the Software Model

The first step in defining a software development process is deciding the **software process model** that best fits the needs of your organization. There are numerous models and variations of models from which to choose. The bibliography at the end of this book lists publications that offer a good start in researching which software process model(s) is best for you; however, several models will be introduced briefly here to help you understand the role that a software process model plays in defining your desired software development process. The models introduced are:

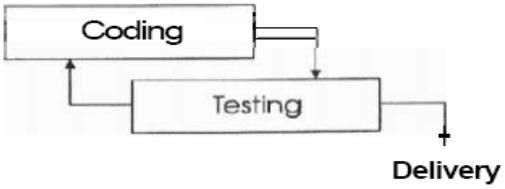
- Code-and-fix
- Waterfall
- Incremental
- Iterative

Many definitions exist for these models; however, the definitions offered here are intended to be simple yet informative. Most models are derived, at least in part, from one or more of these basic models.

Code-and-Fix Model

The code-and-fix model, shown in Figure 1.3, is the oldest and simplest of all the models. Little advance planning is performed, and the user of this model quickly begins the coding stage of the product to be developed. Typically some amount of coding is completed, then the evolving product is tested and the problems discovered are corrected. Then more coding occurs, more testing, more problems are corrected,

and the coding and testing cycle continues until the product has been fully developed and is delivered to the customer. This model is most suited for very small and simple projects. Because advance planning is usually lacking and an informal development style usually accompanies the implementation of this model, the quality of the overall product that is delivered to the customer often is lower than the quality resulting from the implementation of other models. Furthermore, the code often is difficult to maintain because the product's design was not carefully planned and documented.



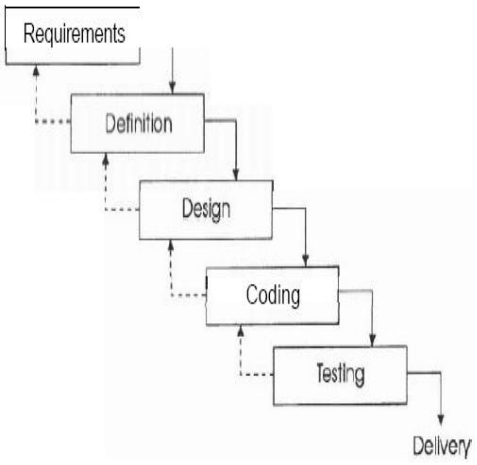
code – and – fix model

Waterfall Model

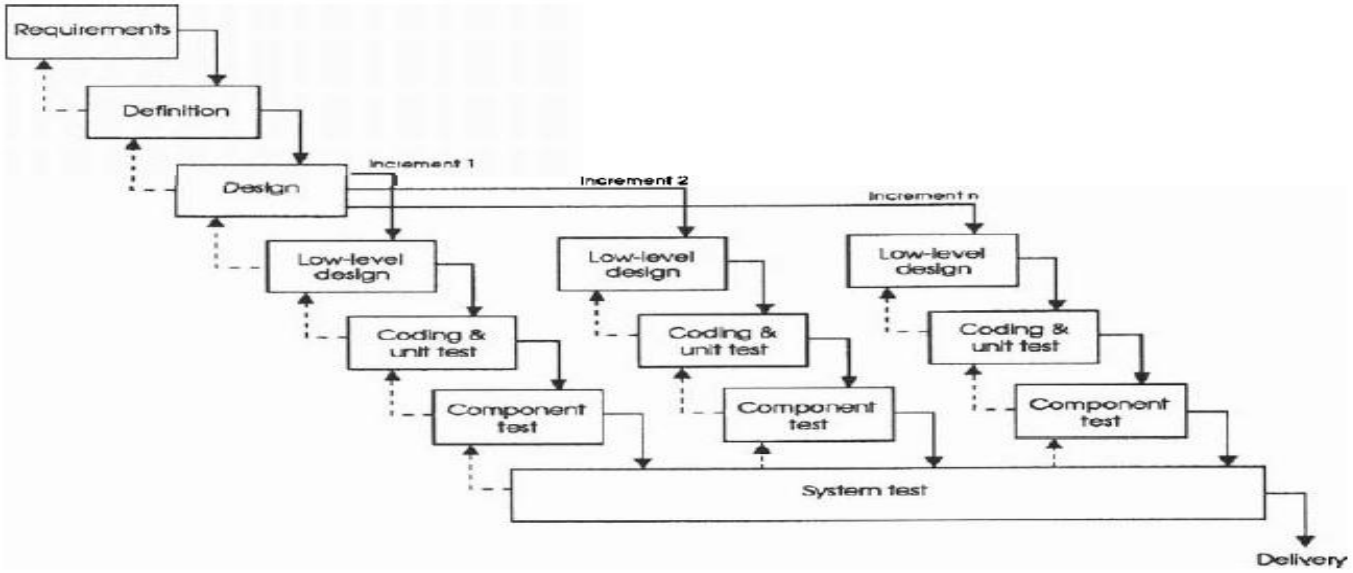
Figure shows a simple representation of the waterfall model. The name is derived from the appearance of the model; that is, as one stage is completed, the project's activities focus on the next stage, so there is a downward flow, as shown by the solid arrows from one activity to the next. The dotted arrows show feedback loops that are activated when there is a need to revisit an earlier stage to redefine, redesign, recode, or whatever. Even though the figure shows the feedback arrows pointing only to the immediately preceding stage, they can return to any stage. The waterfall model generally is suited for medium to large projects with well-defined requirements. This model relies heavily on each stage being completed the first time through and operates best when the change activity, although it might be pervasive, does not result in major changes to the definition and design of the product.

Incremental Model

The incremental model is shown in Figure. While this model is quite similar to the waterfall model, here the product is developed incrementally-in pieces. The figure shows that the requirements, definition, and high-level design are completed and documented. Then the product is developed further in pieces according to a master building plan. After each piece is low-level designed, coded, and tested, it enters into the system test where it is further integrated and tested. The figure represents a common view of the incremental model; however, the development of the pieces could, in fact, begin as early as the definition stage. A popular advantage of the incremental model is that the pieces of the product can be developed largely in parallel to one another. Thus development activities overlap, which has the benefit of potentially reducing the technical risk of product



Water Fall Model

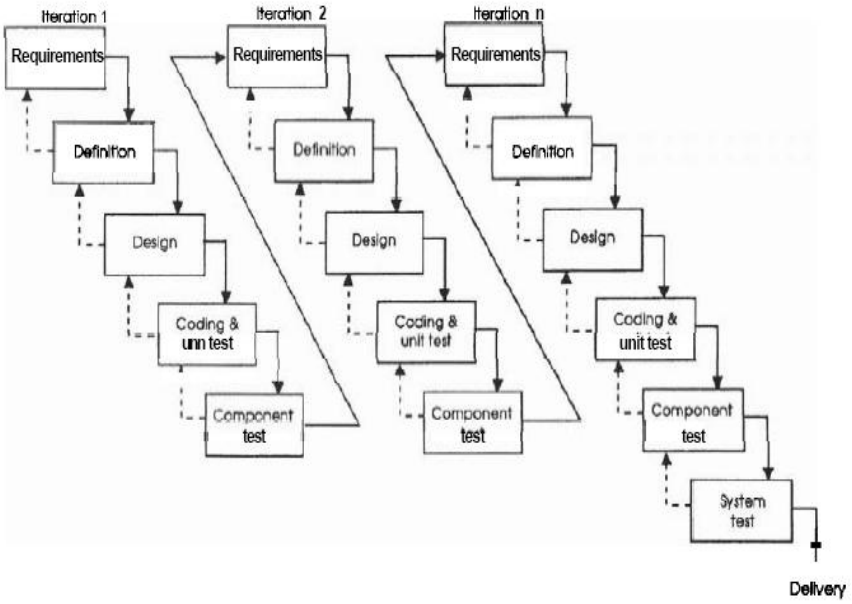


Incremental Model

Iterative Model

The iterative model, represented in Figure, sometimes is confused with the incremental model. Remember, the incremental model can be used when there is a desire to develop, in increments, a product that has known requirements and definition. In contrast, the iterative model typically is used to develop a product that has not been fully defined. The model is used to develop a product to the extent that the requirements and definition are known. As the requirements become better understood and the product is further defined, then the next iteration of the product is developed. Each iteration forms the base for the next iteration. Although the figure shows each iteration beginning with the requirements being revisited, the iterations can begin with the definition or design stages.

The iterative model is particularly suited for use when the requirements and product definition are not well understood and there is a need to begin development more quickly and create a very early version that will demonstrate the look and feel



Iterative Model

of the product. These early drops can help customers identify and fine-tune the requirements and corresponding product definition for which they are searching. The iterative model shares many of the advantages of the incremental model but has the distinct benefit of adapting to changing product requirements. The iterative model also introduces additional process complexity and potentially longer product cycles.

Choosing a Model

Don't feel trapped into choosing one of the four common models that were just introduced. There are literally dozens of other models or variations of standard models from which you can choose (including clean room, spiral). Instead, view these models as examples of the flexibility you have to choose an existing model or create unique one that works best for your organization. Key factors to consider as you search for the right model include:

- Product complexity
- Project size
- Degree that product requirements are documented and understood
- Need for early availability of product function

- Need for customer involvement during development
- Single or multiple customers
- Available software development and project management tools
- Required quality level of product
- Experience and skills of project members, including management
- Physical proximity of project members to one another
- Number of locations and companies involved
- Number of integrated products to be developed
- Maturity of product technology
- Magnitude of anticipated changes during development
- Staffing profile

Step 2. Identify the Activities

After the software process model has been selected, the next step is to identify the primary activities that need to be implemented to satisfy it. The list of activities should be comprehensive because the members of new projects will pick and choose activities from this list as they tailor the software development process to meet the unique needs of a new project. **A** sample abbreviated list follows. A description of each activity in this short list is presented later in this chapter.

- Requirements
- Objectives
- Specifications
- High-level design
- Publication content plans
- Test plans
- Low-level design
- Code
- Unit and function test

Managing Software Development Projects

- Component test
- First-draft publications
- System test
- Second-draft publications
- Regression test
- Package
- Delivery

After the list is complete, write a description for each activity in the list. For example, descriptions for the objectives and specifications might be:

Product Objectives

Description: The product objectives describe the solution to the problem or set of problems described in the requirements. This document defines, at a high level, a product that will satisfy a marketing opportunity and focuses on the perceived needs of the targeted customer. The objectives document also will provide the underlying direction to be followed by the project as functional and design trade-offs are made throughout the development of the product. Direction for both the programming and publication pieces of the product is addressed.

Product Specifications

Description: The product specifications describe, in detail, the externals of the product; that is, they describe what the product will look like to the user. Every function, command, screen, prompt, and other user interface item must be documented so that all participants in the development of the product know the product they are to build, document, test, and support. The product objectives provide the direction and basis needed to develop the product specifications.

Step 3. Identify the Relationships among Activities

With the activities listed and described, now define the relationship between related activities. This can be achieved by listing the entry and exit conditions for each activity. If an activity (called B) cannot begin until another activity (called A) has been completed, then the completion of activity A would be stated as an entry condition to activity B. Let's look again at the earlier example for the objectives and the specifications.

Enty conditions: The requirements are distributed for review or approval.

Exit conditions: The requirements are approved; the objectives are approved and all problems are resolved.

Product Specifications

Entry conditions: The objectives are distributed for review or approval.

Exit conditions: The objectives are approved; the specifications are approved and all problems are resolved.

Step 4. Document Other Useful Information on Each Activity

As the software development process is being defined, the architects of the process will need a method to record tips and other useful information about the intended **use** of each activity. As members of new projects attempt to follow the software development process and tailor it to meet their unique project needs, they will need to review this information. One way to record this information is simply to add a new section to be documented for each activity. The section can be called *notes.* Let's look at an example of how the notes section can be defined by revisiting the running example that addresses the objectives and specifications activities.

Project Objectives

Notes: The objectives can be started after a draft of the requirements is available; however, the objectives must not be approved before the requirements have been approved. This will ensure that the requirements are understood fully before the approvers of the objectives can declare that the objectives are indeed satisfying the perceived needs of the targeted audience. The objectives, after being approved, can be changed only through the designated change control process.

Project Specifications

Notes: The specifications can be started after a draft of the objectives is available; however, the specifications must not be approved before the objectives have been approved. This will ensure that all major problems with the functional direction of26 Managing Software Development Projects the product are resolved before the detailed definition of the product is completed and approved via the specifications. The high-level design should be completed before the specifications are considered finished. This helps to ensure that any high-level design considerations that could impact the externals of the product are reflected properly in the specifications.

The specifications, after being approved, can be changed only through the designated change control process. Later in the chapter you will find an abbreviated list of activities that define a software development process. This is the same list shown in Step 2. Each activity presented contains the four sections described in Steps **2, 3,** and 4: Description, Entry conditions, Exit conditions, and Notes.

Step 5. Document How to Tailor the Process

Following Steps 1 through 4 will result in a documented software development process. This process can be defined further by creating a procedure for each activity that describes, in more detail, what must be done, how it will be done, and ways it can be accomplished. A procedure can be a stand-alone document typically anywherefrom one to 10 pages in length. A procedure, for example, can describe the steps to be followed in writing and approving the specifications. A formatted template also can be defined as a starting point for people about to write specifications.

Rules in tailoring a software development process must be documented and easy to understand.

No two software development projects are exactly alike, even in the same organization. For this reason, the software development process used across an organization must be able to be tailored to meet the unique needs of new projects. The users of the process need to have clear direction as to what liberties they can take in tailoring the process. Don't assume that people will feel empowered to tailor the process. Furthermore, without tailoring rules, some users of the process might tailor out activities that no project should sacrifice (such as specifications and some testing activities).Although a software development process might be comprehensive enough to meet the needs of projects of over 100,000 lines of code, be prepared to answer the question "What liberties can the members of a much smaller project take in tailoring the process?"

Items that need to be addressed when tailoring rules are documented follow. Which activities can be eliminated and which cannot? Clearly designate the activities that are required and those that are optional. If any activities are denoted as optional, yet in most cases they are expected to be performed, and then describe the conditions under which they can be optional.

Don't hesitate to require critical activities. For example, specifications always must be written for a project; otherwise, how will anyone truly know what is to be developed, tested, and documented? Moreover, how can an ill-defined product be supported successfully after it has been delivered to a customer? Which activities can be combined and which cannot?

For example, say there are four code tests: unit test, function test, component test, and system test. Is it acceptable to combine one or more of the tests in an effort to be more productive? What if two other tests were defined in the software development process: performance test and usability test? Must these tests be performed independently, or can they be merged with, say, component test and system test?

More examples: If the objectives and the specifications are both required documents, then can they be combined into a single document? Must low-level design always be done? Or can a programmer proceed directly to coding from completing high-level design? Can user publications be distributed for review only once, or mustan updated second version also be made available for review?

Is it okay to add new activities?
Baited question. Of course it is. Moreover, it is important to ensure that all the members of an organization understand that they must take ownership of the software development process that results after all tailoring has been done. If the defined process has deficiencies (and they all do in the intensely dynamic software industry) , then project members must feel responsible for "making it right" for their project. Who must approve the proposed tailoring? Must all members of a project reach a consensus on the tailoring proposed? Must anyone outside of the project approve the tailoring proposal, such as an assurance group or management?

Typical projects require at least several project documents to be created (such as specifications, test plans), and, for very large projects, sometimes dozens of different project documents may be required. Use past-project documents as the basis for creating new project documents. Some documents may require only moderate changes from project to project. Identify these documents within your organization and capitalize on them. Using past-project documents as starting points for the creation of new project documents can be a great productivity technique.

At least two complete examples showing how to tailor the software development process should be available.
It is recommended that at least two complete examples of tailoring the software development process are made available for users of the process. Examples are powerful teaching tools and also can improve productivity. The first example should be representative of a typical project within the organization. This example will be heavily copied as a starting point for most projects. The second example is for a very small project.

The small-project example is important because it shows users of the process just how flexible the process can be to meet their needs. It also helps to ensure that very small projects do not incur unnecessary overhead and "bureaucracy" in using the process.

It is better to re-create a software development process from a better-fitting software process model than to try to force a project to use an ill-fitting software development process.

When the software development process is being tailored to a new project, and it is decided that the software process model upon which the software development process was built is not the best model for the new project, then what should you do? The most productive approach is to start over at Step 1. However, the steps should go much faster than the first time through because many of the activities already defined for the original software development process will apply directly or with only minor adjustments to the new software development process. Re-creating a software development process from a better-fitting software model is far better than trying to force a project to use an ill-fitting software development process.

Step 6. Document How to Improve the Process

Now that the software development process is defined and rules on tailoring the process are documented, there needs to be a well thought out method to ensure that the software development process can be improved continually. Three cases for addressing changes need to be addressed.

1. Change requests

Change requests are suggestions for making changes to the process. As project members use the software development process, they often will identify areas for improvement. These change requests need to be collected and addressed relatively quickly. Why quickly? For two reasons: so that the documented software development process can be updated (improved) on a real-time basis so that current and new projects can benefit from these suggestions, and also to demonstrate to the organization’s members that the leadership is serious about these improvements and that suggestions are not only welcomed, but they are encouraged and appreciated.

2. Deviation requests

Deviation requests are requests to deviate one ***time*** only from some aspect of the software development process. Deviation requests are submitted while a project is in progress. These requests require a very speedy response because the decision to grantor denies them can have a significant impact on the schedule and cost of a project. As discussed later, deviation requests are evaluated carefully to determine if the software development process also should be modified.

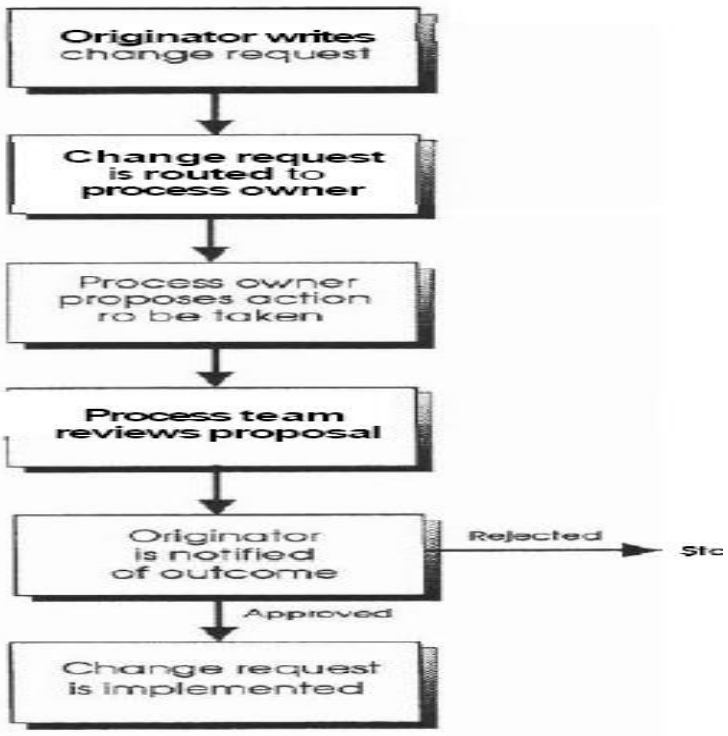
3. "Post project review" change requests

When a project has just completed and the product has begun its packaging and delivery-to-the-customer stages, the project's membership (or a representation thereof) should meet to perform a post project review. Any recommendations for changing the software development process should be submitted and considered relatively quickly, for the reasons stated earlier."But," you ask, "where do these change requests and deviation requests get submitted and how do they get acted upon?" An organization can adopt many approaches to address changes to the process. The approach chosen has a lot to do with the size of the organization, whether the members are in close proximity to one another or geographically dispersed, the process maturity level of the organization, and so on. However, let's examine an approach that can be effective in almost any work environment. A *software development process team,* **process team,** is formed. The process team is made up of members who, collectively, represent all areas from across the organization. $is team has total responsibihy 'for defining, documenting, simpllfymg, improving, and managing the implementation of the software development process.

The overall responsibility for the software development process needs to be distributed across a representative set of peoples evenly and as fairly as possible.

Each process team member "owns-that is, is responsible for--some piece of the process. For example, each activity or major subject defined in the software development process (such as specifications, code, component test, tailoring rules) is assigned to a team member. A team member, also referred to as a *process owner,* can own more than one activity or major subject area, but the overall responsibility for the software development process needs to be distributed as evenly and as fairly across the team as possible. It is important that everyone on the process team personally feels ownership for some aspect of the software development process. The process owner has the prime responsibility for being the advocate for his or her activity or subject area. So how do change requests and deviation requests get acted upon? Let's first take the case of change requests as shown in Figure.

A change request can be written by any user of the software development process. The change request is submitted to the process team and is routed to the appropriate process owner. (The originator can send the request directly to the process owner if he or she is known.)The process owner proposes the action to be taken on the request and then sends the request and the proposed action to the process team members.



Change Request

The process team members can approve, disapprove, or abstain from taking a position. If a consensus is not reached, then the process owner works with the process team members either to reach a consensus or to not approve the change request. The process owner then notifies the originator as to the outcome of the change request. If the request is rejected, the originator can escalate the decision to a predefined management path where the change request can be reevaluated. Usually, however, it is expected that the originator will accept the process team's position. If the change request is approved, then the process owner has the responsibility to ensure that the change is implemented in a timely manner. After the change has-been written up satisfactorily in the software development process documentation, it must be approved again by the process team members to ensure that the implementation is understood, correct, and complete. After the documentation has been approved, the members of the organization must be educated on the change so that it is adopted across the organization immediately.

Now examine an approach that can be-used for deviation requests. Figure shows a project member writing a deviation request. The request is then reviewed by the leader of the project. Why the project leader? Because it is important that a deviation is first agreed to by the person with the overall responsibility for the project. If the project leader supports the deviation request, it is routed to the process owner just as a change request is routed. The process owner is fully empowered to approve or reject the deviation request. If the deviation is rejected, the deviation request process usually stops. However, the project leader can escalate the deviation request to a predefined management path, just as we defined for change requests. If the deviation is granted, either by the process owner or because of an escalation to management, then no further effort is required and the one-time-only deviation is a done deal. Although random change requests, deviation requests, and change requests resulting from post project reviews offer good feedback as to the applicability of the software development process in meeting the needs of the organization, other action scan be performed as well. For example, a survey can be conducted periodically to gain further insight. Also, management can talk to members of the organization periodically, individually or in small groups.

Mostly non management personnel should own and operate the software development process.

Just what is management's role in defining and implementing the software development process? It is strongly suggested that *no managers* predominantly own and operate the process. This is because it is mostly nonmanagement that must use the process; therefore, nonmanagement must feel the ownership for the definition -and application of the process. Although management, particularly at the top, must be supportive of enforcing the software development process, the front-line troops ***L A*** - must feel the true ownership of the process and act accordingly.

Step 7. Obtain Buy-in of the Process

At this point, the software development process is fully defined, helpful instruction how to tailor the process are documented, and the methods of improving the process have been identified. What remains to be done? Two things:

1. Obtain the organization's commitment to use the process.
2. Train every member of the organization on the proper use of the process.

The process team must fully approve the newly defined process. As representatives of the overall organization, each process team member is approving for his or her area of the organization. If a process team member is hesitant to approve, then the member can solicit opinions from his or her represented area. Keep in mind, however, that only the representatives can approve or disapprove the process. Although opinions from their constituents are helpful, management has fully empowered the process team members to commit their portion of the organization.

Everyone in the organization must be educated on the description and use of the software development process, including all levels of management.

After the process is approved by the members of the process team, it is time to educate everyone, including all levels of management, on the new process. The education process is the best way to show people that the process is committed and to let them know that their full support is expected. Depending on the complexity of the process, the size of the organization, and other factors, the education can take on different forms. For example, it can be mandatory that everyone in the organization participate in a one-day training class. Another approach is to spend one-half day introducing most members of the organization to the new process and reserving the

Deviation Request

Discipline is the soul of an army. It makes small numbers formidable, procures success to the weak, and esteem to all.

-George Washington

All people want and need to know the acceptable pattern of behavior that is expected of them.

Everyone wants discipline. Everyone wants to work in an environment where people know what to expect. Again, discipline is the glue that holds a project together. It is the tool for managing change--and change is essential for progress. The processes and methodologies employed within a project cannot be sustained without the necessary, underlying discipline. A project needs discipline to achieve the desired level of accomplishment for each of its major parameters. These major project parameters are listed in Figure.

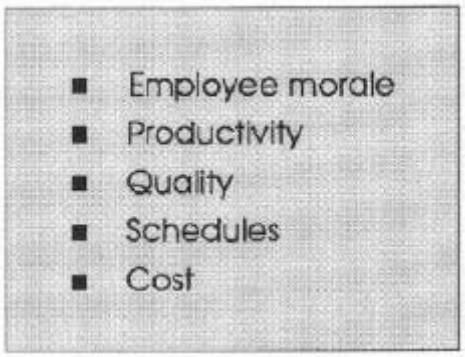
The following sections offer insight on the impact that discipline can have on these major project parameters.

Employee Morale & **Projects run their beet when employee morale is high.**

While great human achievements typically are not accomplished on morale alone, history seems to show that strong morale has added to the effectiveness of many great achievers. Obviously high employee morale offers great value to a project. Good morale can have a positive affect on every major project parameter. However, discipline from the project's leadership is essential for achieving high morale within an organization.

For example, project members want and need to:

- Know what their mission is
- Understand their assignments
- Understand how they are measured against their performance
- Know that they will be recognized and rewarded for their achievements
- Know what to expect from their leaders
- Believe that project leaders make a genuine effort to understand their people
- and maintain good, two-way communications
- Believe that project leaders will make the best decisions for the success of the
- project



Major Project Parameters

When project leaders exhibit discipline in insisting on an environment that satisfies these wants and needs from the project's personnel, then almost anything can be accomplished. Significant accomplishment, however, is impossible when the project's management fails to exercise the necessary level of discipline that is needed to create and sustain such an environment-an environment that encourages high employee morale.

Productivity

Employee productivity is a t its highest when employees know what t o do, how to do it- and do it!

Employee productivity is at its best when project processes are defined, measurable, and enforced-and project members are educated about their roles. Discipline within the project is required to make these things happen. Consider an example. In every software development process, the product passes through phases as it is being developed. Some typical phases are:

Product definition

Product design

Code

Each of the project's phases can, in turn, be defined in more detail. For example,"product specifications" falls within the phase "product definition." The product specifications activity could be divided into five smaller activities:

- Product specifications preparation
- Product specifications review
- Product specifications update
- Product specifications approval
- Product specifications refresh

Each of these activities can be defined further in terms of entry, implementation, and exit conditions. (See Chapter 4 for more on phases, activities, and process conditions.)After the project's processes to be followed are defined to a level at which the participants can measure their adherence, the project members then must be properly trained and educated to understand those processes fully. Finally, those processes must be fully supported and enforced by the project leaders. To make all this happen, the project's leadership must demonstrate discipline.

Quality

Quality will suffer without deliberate discipline.

Quality is another major project parameter that will suffer without discipline. It seems that many people have their own definition of quality. Regardless of the definition used, however, there is always a great need to define and follow processes that will yield the desired product quality. While quality often is associated with the "worker bee" in the trenches doing the designing, coding, or testing, the project's leadership must fist exhibit the discipline that leads to a quality-producing work environment. There is a real temptation to sacrifice quality first-whenever a project falls behind schedule. But quality actually should be the last parameter to be sacrificed, if ever. Sheer discipline from the project's leadership is required to avoid the let's-lower-the-quality trap. The following saying holds true for too many projects-perhaps even yours:

*We never have enough time to do it right, but we always find time to **do** it over,*

Schedules

The need for continuous discipline is perhaps most evident when managing a project's schedules.

This saying leads into the next reason for discipline schedules. How many projects do you know about that actually finished under the same schedule they began? For those projects that changed their schedules, how much of a contributing factor was the lack of project discipline by the project leaders? Earlier it was mentioned that change is essential to progress. When a project's schedules are defined and approved early in the software development process, many assumptions and dependencies are identified.

As time passes and some activities complete and many more begin, the project personnel who participated in the creation and approval of the schedules become more knowledgeable. For example, a certain document that was estimated to take four weeks to write might now require six weeks because the expected dependencies were late or because the effort simply was underestimated. What is happening is that *change* is being introduced into the project equation. In order to maintain the overall schedules, the discipline required to manage this ongoing change must be alive and active. Software development projects are not static. They are extremely lively and in constant need of attention. Discipline from project leaders is vital to maintain the overall, committed project schedules.

Cost

Cost is another major project parameter at the mercy of discipline. Budgets are affected by such factors as the number of programmers involved; the number of computer workstations available, the tools employed, office space, furniture, and so on. The list can be extensive. Even the timing chosen by a project's leaders to begin moving people from one project into another can be quite costly. The opportunity to spend beyond the budget can be too tempting. "Borrowing from Peter to pay Paul "only defers pain into the future. Rationalizing a multitude of ways to recover costs can become easy. Of course, when recovery plans are implemented later, many turnout to have looked better on paper. Here again, discipline by project leaders is essential--essential in routinely controlling budgets so costs can be contained.

A great deal of discipline is required to manage the constant change that is common to all software development projects.

All of the major project parameters--employee morale, productivity, quality, schedules, and cost-influence each other to some degree. For example, if morale is low, then quality and productivity will suffer. This will cause schedules to be extended, which, in turn, will increase costs. But no matter which parameters are used to show this domino effect, any parameter that "goes south" can pull the overall project with it. Again, the management of change is critical to the success of the project. And critical to the management of change is the discipline required to hold all parameters of a project together.

Have you ever noticed that some organizations seem to be more successful than others? That the energy level of the people involved seems to be higher? That these people generally seem to have better attitudes about themselves and the work they are doing? That more things just seem

to go right? Also, have you noticed that these organizations seem to be able to attract the most interest from employees in sister organizations who desire to join?

The better managed projects manage discipline better.

What is so unique about these seemingly "magnetic" organizations that attract good fortune at most turns? The general answer is that they are managed better. The specific answer is discipline exercised by project leaders in both ***what*** they do and ***how*** they do it. Discipline comes in many flavors, but only the discipline that supports the project's mission is desirable. This is the discipline that supports the pattern of productive behavior needed in and wanted by project personnel. This is the discipline that should be encouraged. This is called positive discipline. Positive .discipline is what this chapter is all about.

Before venturing further into this topic, it can help to take a brief look at negative discipline. Remember, discipline is the act of encouraging a desired pattern of behavior. If the leader of a group trains the group's members to follow a certain pattern of behavior, and that behavior is not productive to achieving the group's mission, then the discipline exercised is negative discipline. As an example, consider an organization that needs its employees to take more risk in accepting responsibility. Now consider leader within that organization who continually punishes each risk taker who meets with failure. This leader would be displaying negative discipline because the discipline works against the project's mission. The scenarios at the beginning of this chapter provide additional examples of negative discipline.

Implementing Discipline

Now consider positive discipline once again, focusing on the discipline that project leaders demonstrate in both *what* they do and ***how*** they do it. Figure shows the four essential traits that are what of the well-disciplined organization. This is a good point to examine these traits closer and discuss ***how*** they need to be addressed.

Trait One: Set Realistic Goals

Every organization needs goals. How else can success be measured? Goals must be:

- Simply stated
- Understood by all
- Measurable

When project leaders exhibit discipline in insisting on an environment that satisfies these wants and needs from the project's personnel, then almost anything can be accomplished. Significant accomplishment, however, is impossible when the project's management fails to exercise the necessary level of discipline that is needed to create and sustain such an environment-an environment that encourages high employee morale.

Productivity

Employee productivity is a t its highest when employees know what t o do, how t o do it-and do it!

Employee productivity is at its best when project processes are defined, measurable, and enforced-and project members are educated about their roles. Discipline within the project is required to make these things happen. Consider an example. In every software development process, the product passes through phases as it is being developed. Some typical phases are:

- Product definition
- Product design
- Code

Each of the project's phases can, in turn, be defined in more detail. For example, "product specifications" falls within the phase "product definition." The product specifications activity could be divided into five smaller activities:

- Product specifications preparation
- Product specifications review
- Product specifications update
- Product specifications approval
- Product specifications refresh

Each of these activities can be defined further in terms of entry, implementation, and exit conditions. (See Chapter 4 for more on phases, activities, and process conditions.)After the project's processes to be followed are defined to a level at which the participants can measure their adherence, the project members then must be properly trained and educated to understand those processes fully. Finally, those processes must be fully supported and enforced by the project leaders. To make all this happen, the project's leadership must demonstrate discipline.

Quality

Quality will suffer without deliberate discipline.

Quality is another major project parameter that will suffer without discipline. It seems that many people have their own definition of quality. (See Chapter 6 for more on planning for quality.) Regardless of the definition used, however, there is always a great need to define and follow processes that will yield the desired product quality. While quality often is associated with the "worker bee" in the trenches doing the designing, coding, or testing, the project's leadership must fist exhibit the discipline that leads to a quality-producing work environment. There is a real temptation to sacrifice quality first-whenever a project falls behind schedule. But quality actually should be the last parameter to be sacrificed, if ever. Sheer discipline from the project's leadership is required to avoid the let's-lower-the-quality trap. The following saying holds true for too many projects-perhaps even yours: We never have enough time to do it right, but we always find time to **do** it over,

Schedules

The need for continuous discipline is perhaps most evident when managing a project's schedules.

This saying leads into the next reason for discipline schedules. How many projects do you know about that actually finished under the same schedule they began? For those projects that changed their schedules, how much of a contributing factor was the lack of project discipline by the project leaders? Earlier it was mentioned that change is essential to progress. When a project's schedules are defined and approved early in the software development process, many assumptions and dependencies are identified. As time passes and some activities complete and many more begin, the project personnel who participated in the creation and approval of the schedules become more knowledgeable. For example, a certain document that was estimated to take four weeks to write might now require six weeks because the expected dependencies were late or because the effort simply was underestimated. What is happening is that *change* is being introduced into the project equation. In order to maintain the overall schedules, the discipline required to manage this ongoing change must be alive and active. Software development projects are not static. They are extremely lively and in constant need of attention. Discipline from project leaders is vital to maintain the overall, committed project schedules.

Managing Software Development Projects Cost

Cost is another major project parameter at the mercy of discipline. Budgets are affected by such factors as the number of programmers involved; the number of computer workstations available, the tools employed, office space, furniture, and so on. The list can be extensive. Even the timing chosen by a project's leaders to begin moving people from one project into another can be quite costly. The opportunity to spend beyond the budget can be too tempting. "Borrowing from Peter to pay Paul"only defers pain into the future. Rationalizing a multitude of ways to recover costscan become easy. Of course, when recovery plans are implemented later, many turn out to have looked better on paper. Here again, discipline by project leaders is essential-- essential in routinely controlling budgets so costs can be contained.

A great deal of discipline is required to manage the constant change that is common to all software development projects.

All of the major project parameters--employee morale, productivity, quality, schedules, and cost-influence each other to some degree. For example, if morale is low, then quality and productivity will suffer. This will cause schedules to be extended, which, in turn, will increase costs. But no matter which parameters are used to show this domino effect, any parameter that "goes south" can pull the overall project with it. Again, the management of change is critical to the success of the project. And critical to the management of change is the discipline required to hold all parameters of a project together.

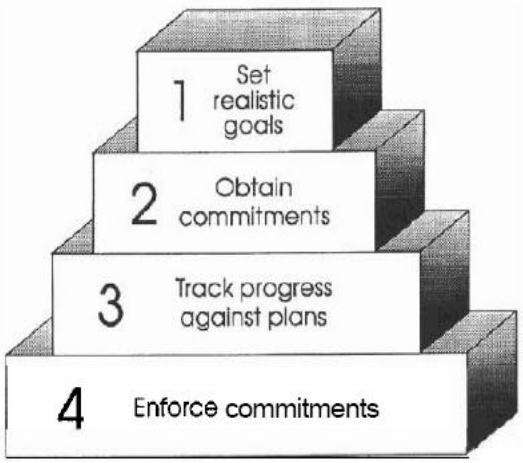
Implementing Discipline

Now consider positive discipline once again, focusing on the discipline that projectleaders demonstrate in both *what* they do and ***how*** they do it. Figure shows the four essential traits that are the *what* of the well-disciplined organization. This is a good point to examine these traits closer and discuss ***how*** they need to be addressed.

Trait One: Set Realistic Goals

Every organization needs goals. How else can success be measured? Goals must be:

- Simply stated
- Understood by all
- Measurable



Traits of the well-disciplined organization

Discipline begins by setting realistic goals. To "do good is not a goal. To build a defect-free product is a goal. However, to expect a defect-free product might not be realistic. If your product will have 1 million lines of code, and your measure of success is to prove that it is 100 percent defect-free, then you will likely go out of business. Why? ~because the tremendous cost to develop1 million lines of code that is defect-free would likely extend schedules and raise the product price to a point that would reduce its competitiveness sigruficantly. However, if your goal is to deliver this product with no more than one defect for every10,000 lines of code, and technology is within reach to make this happen, then your goal is realistic. (In this example, assume that the customer accepts this defect rate. Also assume that the frequency and effect of the defects discovered by the customer are manageable-for instance, the defect is encountered only once during the start of leap year and will not lead to disaster.) . , How do goals (that are theoretically achievable) happen? They happen when the project's leadership establishes and maintains a productive environment. The leaders must make it easy for people to do their jobs and must create a work environment that sets people up for success, not failure. In creating a productive environment, project leaders should strive to:

- Provide the necessary training, processes, and tools
- Offer a sense of accomplishment
- Foster teamwork
- Encourage risk-taking
- Now take a closer look at these elements.
- Provide training, processes, and tools.

A goal is not realistic if the people expected to make it happen have not been trained properly, processes have not been defied and implemented, and the necessary tools have not been made available. The project's leadership is responsible for making these things happen. In the l-million-lines-of code program example, project personnel will not know if they have achieved the acceptable defect rate unless a rigorous software development process has been defined and implemented to track and measure product defects carefully. Offer a sense **of** accomplishment. People achieve their best when they are "stretched-when their skills are used and their potential is tapped. When these things happen, people sense they are valued as members of the team. Project leaders should not hold back in providing people with assignments that are challenging but achievable. A project's goals are closer to being realistic when the project's members are happy about their work. Foster teamwork. Fostering teamwork involves encouraging the participation of all project members. Great human achievements are possible when people work as a team. Whether the project is to harness the great energy of the atom, to walk on the moon, or to build a large complex software program, teamwork draws on individual accomplishments. These individual accomplishments are collected in a fashion that allows greatness to be achieved at a level far beyond the abilities of any one person. Everyone has something to offer to a team. The more participation is encouraged, the greater the likelihood that the project's goals will be met.

Encourage risk-taking. Taking risks is the difference between *doing* the unthinkable and only *dreaming* about it. Establishing a risk-supportive environment can allow the imagined to become reality. It may be *the* ingredient that allows the estimated1-million-lines-of-code program to be done with ***25*** percent less code. Or it may simply make the difference between delivering a product on schedule or much later. An environment that encourages risk and rewards success but does not penalize failure is an organization to be reckoned with. The movers of tomorrow are taking risks today.

Trait Two: Obtain Commitments

The last section, "Trait One: Set Realistic Goals," stresses that a well-disciplined organization defines realistic project goals in a manner that is simply stated, is understood by all, and is measurable. Furthermore, all project players understand their individual assignments and roles in making the bigger picture happen. A second trait of a well-disciplined organization is the obtainment of commitments from *each* person in the organization.

Everyone must feel personally committed for discipline to have its greatest impact.

A committed plan does not exist until *everyone* has made a personal commitment. This means from the very top gun to the troops in the trenches, managers and nonrnanagers alike. People will take more pride in their work when they have a personally committed stake, when they sense they have responsibility and accountability. No greater tool exists for motivating people to do their job **than** getting their personal commitment to making it happen. Giving people the opportunity to participate in developing product content, processes, and schedules are not only beneficial, it is a must.

Trait Three: Track Progress against Plans

At this point, the organization has a realistic project plan (trait one) to which all members of the project have committed (trait two). So far, so good. The third trait is the tracking of each activity against the plan.

It is not enough to plan your work; discipline requires that you also ensure that you are working your plan.

Remember that discipline is the act of encouraging a desired pattern of behavior. Now that a plan for the project is in place, how can the project's leadership be sure that the plan is being followed continually? More important, how will the leadership know when problems arise and where resources should be redeployed to help solve problems and protect the planned schedules? Discipline is required to track the plan on a regular and frequent basis. Tracking the plan also involves recording new problems and ensuring that current problems are being solved satisfactorily.

Scenario: Consider the plight of a person walking through a desert. Without sophisticated navigation tools, it is highly improbable that this person could walk a straight line through the desert. (For this example, assume it is physically possible to track to a straight route, free of obstacles.) In this analogy, the start and endpoints of the person's journey represent the start and end points of a project. The straight line, which is the shortest route through the desert, is symbolic of the shortest project schedules possible. Now picture this person veering a little more off course each week. For any given week, the deviation doesn't represent a major alteration of the final destination. However, as the weeks pass, these minor off course excursions collectively could spell disaster. That is, the final destination would not be reached anywhere near the planned date. If, however, this person’s direction could be reset each week, problems could be

addressed close to the time they occur, so that the final destination's targeted arrival date has a much higher chance of being achieved.

The desert example is simplistic but nevertheless provides insight into the need to track against a project plan at frequent, regular intervals. Often just the act of tracking the plan is a form of preventive maintenance. People are more apt to meet a checkpoint if they are being tracked regularly and frequently than if they are tracked infrequently.

Trait Four: Enforce Commitments

You may want to read this section twice. The reason is simple: If everything mentioned up to now has been done setting realistic goals, obtaining cornmihnents,tracking progress against plans-but this final, fourth trait of the well-disciplined organization is not made to happen, then all bets are off. Enforcing commitments is an absolute must. This is not a strong arm tactic. Rather, the enforcement of commitments represents a statement of support from the project's leadership to the project’s participants.

Most software development projects will encounter several severe problems along the way. (A severe problem is defined here as one that potentially can cause a delay in the final delivery of the product.) Moreover, many severe problems are not totally solvable by the specific group that is experiencing the problem. An example is the team that falls behind schedule in writing test scripts. The people are all working overtime but still may not be able to complete the activity on schedule. The person leading this team has no other resource to add to this effort. The project's leadership, however, can choose to redeploy people from other areas of the project to shore up the development activity of the test scripts. Therefore, one useful approach project leaders can employ to ensure that commitments are met is *management of priorities.* Priorities requiring attention often will vary from week to week. As a result, management of priorities requires discipline to ensurethat the proper activities are getting the needed resources and focus. Often it is morefun and easier to deal with some problems ahead of others. However, this temptationshould be resisted. Instead, it is better to understand problems and take action on resolving them according to priorities that best serve the organization. Another important action to take in enforcing commitments can be called "making it happen now." This is a tightly held philosophy of leaders who have a reputation for getting things done. Whereas management of priorities ensures thatresources within the organization are being diverted dynamically for the good of the total plan, "making things happen now" is the act of dealing swiftly with problems

before they fester and grow out (or further out) of control. This is considered to be a strong positive act of support for the people in the organization.

People need to be rewarded regularly for demonstrating the desired behavior.

Reward those who meet or beat their commitments. Whether the reward is expressed privately or publicly, stated on paper, made with money, or made through some other means, it is important to provide feedback to individuals and to the organization. Let people know when their behavior contributes to the project's goals.

Everyone looks to the project's leadership to provide a work environment that encourages success.

By the same token, proceed cautiously before reprimanding failure. Be firm but fair. Maintain a sense of justice and fair treatment. Most people don't fail intentionally. Could it be that the project's leadership did not provide the proper work environment to facilitate the employee's success? If it is clear that a person is performing unsatisfactorily, don't ignore this. Help the person to develop an acceptable level of performance. If, after reasonable energy expenditure, the person still is not showing the needed improvement, then find a job that fits this person's skills or remove him or her from the company. *Do not do nothing.* All eyes are on the project's leadership to take proper action before the situation deteriorates further.

Unit - II

Project Schedule Planning

The schedules created for a software development project can make or break the project, the product, and the people. The attention and forethought applied to this critical scheduling activity of developing the project schedule plan can mean the difference between:

- Management in control and management in panic
- High product quality and poor product quality
- A full-function product and a limited-function product
- High employee morale and low employee morale
- High employee productivity and low employee productivity
- On-time (or early) delivery and late (or no) delivery
- Competitive product costs and uncompetitive product costs
- Successful market entry and unsuccessful market entry
- Customer satisfaction and customer dissatisfaction
- Marketing strength and marketing weakness
- Timely next release and late (if at all) next release
- Product success and product failure

Simply put, the project scheduling activity can make the difference between profit and loss. The heartbeat of the entire software development process-and the single most important plan of a project-is the project schedule plan. It defines the roadmap of activities that affect virtually every member of a project and is the keystone for communications across a project.

If the schedules defined by the project schedule plan are unreasonable, then the expected progress on the project soon will become blocked. This blockage will cause project challenges to emerge, challenges that would otherwise be unnecessary, challenges that now must be met in order to deal with the obstructions. The failure of a major activity to be completed on schedule eventually will impact the schedules of subsequent project activities. This domino effect could continue until the project topples. If you want to minimize potential rework on a project and meet the project's schedules, costs, and quality, a generous amount of care and attention must be given to the development of the project schedule plan. Experience suggests that a well thought out plan, in concert with a sound tracking and problem management process can save both costs and cycle time for a project.

The project schedule plan is all about getting in control.

As Plato said, "The beginning is the most important part of the work." These rings true for two key areas of a project: defining what you are going to build and developing the overall plan to build it. Defining what you are going to build is embodied across the product requirements, product objectives, and product specifications areas of focus for later chapters. Developing the project schedule plan to build the product is the primary theme of this chapter. Describes the basic concepts that need to be understood before an effective project schedule plan can be developed, and Presents a series of steps you can follow in developing a project schedule plan for your project.

A Project Tale

The following scenario illustrates how an unrealistic project schedule plan can snowball into a major mess. Doing a poor job developing a project schedule plan is akin to scheduling a project for failure. During the first month of the project, an unofficial, rough scheduling estimate allows for a 12- to l&month product development cycle. **A** month later, after the product requirements have been written and approved, a preliminary schedule is developed that suggests **12**-month duration. Everyone in touch with the project knows that this 12-month schedule is quite optimistic; however, no one is overly. Concerned because the schedule is only preliminary A final schedule with all the necessary detail is forthcoming. In the meantime, the product objectives are completed and all the participating groups agree on the direction the product should take. The writing of the product specifications has begun. The high-level design is also beginning. There is an overall good feeling about the project. Staffing of people with the required skills is under way. Coming on board are programmers, for both developing and testing the product, publications writers, and support personnel. Everyone understands that this project is extremely important to the company. Everyone also understands that the product must be delivered to customers as soon as possible. Two factors are offered as major reasons for supporting a nearly customer delivery. The first reason is to bring in revenue in the next fiscal year, which starts in one month (and, therefore, ends in 13 months). The second reason is to help ensure that several key customers choose this product over competitive products. No one doubts that the product must be delivered as soon as possible. The project planners commit to having detailed project schedules in place within the next two weeks. One planner remarks, "The detailed schedules really could be available within just one week. This project is a lot like others that I have planned, and there really isn't much to laying out the schedules." The planners decide not to include the technical people in the project scheduling activities. The rationale is "Leave the technical people alone as much as possible so they can get some 'real' work done."

A draft of the detailed schedules is ready in three days, when it is reviewed by a few members of the management team. Everyone agrees that the schedules are aggressive, but they also believe that this is the "right" schedule plan if the product is to hit the marketplace on time. Some minor changes are made and a final draft is produced. The planners survey most of the technical leaders about the schedules. Some technical leaders express opinions that the schedules are too aggressive and could be met only "if the wind is at our backs and no surprises occur, of which both are unlikely events." Also, there is concern that the projected staffing might be too aggressive to be achieved, or, if staffing levels are achieved, the skill level of the new people might be lower than required. If the project is staffed with people with a lower skill level than anticipated, more time will be necessary to bring these people "up to speed." The reply from the planning department is "Don't worry about staffing. That is our turf and we are working on it."Some of the technical leaders feel that vacations and holidays have not been planned into the proposed schedule sufficiently, especially since some people haveacaued excess vacation days from the previous aggressive project. The response is -- . ."Overtime was not planned. Therefore, overtime acts as a buffer should it be needed. Besides, there are many months before the end of the year when major holiday and vacation periods could occur. We should be in pretty good shape by then."Another concern is that the expected programmer productivity rate is on the high end of the range rather than in the middle or on the low side. The response from planners is that most of the technical leaders are seasoned veterans whose high skills skewed the productivity rate higher. The planners add, "If this project is to be successful, we will have to achieve the higher productivity rate." A few other concems are raised, but are essentially shrugged off. However, to address(appease) the concems of the technical people, schedules are lengthened by two weeks. While this does not significantly satisfy them, it does help to reduce some of the tension that is building. The technical leaders rationalize, "The product will be done when it's done, so why make a big deal of it now? Anyway, nearly every project seems to follow this pattern. So what's new?"The planners grumble, "It's too bad the lead technical people don't have more business sawy and realize that, to make it big in this business, you have to take big risks. That's why the technical people don't run the business. We will just have to drive the organization to achieve this schedule."There seems to be an unsettled feeling across the project, something of a standoff. No mutual meeting of the minds ever took place where each side could work through the concerns of the other side. (In fact, there should not be a notion of "sides." Instead, the business and technical objectives should be shared by all. **A**we-are-in-this-together spirit should be the prevailing goal.) The general conclusions that perceived standoffs like this have never stopped forward progress before. This is "business as usual."The product specifications are written, then reviewed. There are more review comments than expected.

The reason appears to be that the specifications were not quite finished (nor was the high-level design), but in the interest of maintaining project schedules, the product specifications were distributed on time. The planners are happy. The technical people are not happy, but recognize the importance of meeting schedules. The product specifications are updated to address there view comments and then redistributed for everyone's use. More comments are generated from the updated specifications. The most recent comments are all but ignored. It seems that the schedules did not allow for a second review of the product specifications. It was expected (and hoped) that the review cycle could be shortened to a single, three-week review period, rather than two, two-week review periods with a document update period in between. Since more time is required to perform the single update of the product specifications than has been planned, the high-level design is slipping farther behind schedule. The developers are working overtime in an attempt to finish the high-level design as soon as possible. Furthermore, one of every two design inspections is failing. The schedule had allowed for design inspections but not for recovery time for inspections that failed. Even with the overtime that people are working, the schedule is slipping slowly and frustratingly. High-level design is finally completed yet is two weeks behind schedule. The perception is "Not too bad," especially when the two weeks that were added to the project are taken into account. As low-level design, coding, test plans, and publication content plans are being developed, a disturbing realization takes hold. The product specifications are still not sufficiently complete. The prevailing thought is that they would have been complete if they had been through two review cycles, with a document update period in between. Unfortunately, it is too late to turn the clock back. All developers do their best to update their personal copy of the specifications as they proceed with their low-level design and coding. However, there is no time for them formally to update and redistribute the product specifications to other groups within the project. Now the testers and the publications writers begin to fall behind their schedules. Many options are offered, but only one will really solve the problem: The developers must update and redistribute the product specifications. They do just that. Several weeks pass. The product specifications have been updated and distributed throughout the project. The low-level design has been completed. The coding, unit testing, and function testing, however, are farther behind schedule. Also, the testers and writers discover that they are not able to recover all their lost time. They perceive that the real critical path of the project lies not with them but with the developers-in getting the code ready to enter the first phase of the formal test period. It seems that, when the schedules originally were developed, no one thought to include the writing of the unit test and function test plans. In the interest of protecting schedules, the unit test is abandoned and the function test is now the focus of activity.

The project is now six weeks behind the planned schedule. If any additional slips occur, the project will not be able to bring in revenue for the next fiscal year, as planned. (The two-week buffer, plus starting one month before the fiscal year began, accounts for the six-week leeway.) In response, the code is declared ready for the first independent test phase--the component test. Plans are put in place to

complete the function test in parallel with the component test. A medium disaster develops. Two groups, the developers and the testers, are performing tests on the same code and at the same time. Many of the same bugs (defects) are being discovered by both groups. To make matters worse, the developers' response time in fiing the bugs found by the testers is much longer than desirable. The reason: The developers are busy trying to complete their own function testing. To "fix" this problem, the developers are instructed to correct bugs - - according to priorities set by the testers. The project leadership stresses the importance of containing the duration of the component test to its original plan. How- - - - ever, in spite of this mandate, the component test takes four weeks longer than planned. The code that the developers delivered was just too "buggy." The schedule is now 10 weeks longer than was originally planned. The developers have been working a heavy dose of overtime for many months. People are weary, frustrated, and edgy. The major holiday and vacation period has arrived, but few are able to take any appreaable time off. The most repeated phrase is "I don't want to go through this again!" The second most repeated phrase is "But it will happen again. It always is the same way. We never seem to learn from our mistakes!"

If this scenario were to run its course, the project would be nearly four months late. This would cause the product cost to come in significantly higher than planned. Even though the "planned project schedules were apparently too aggressive to begin with, the expected, and missed, delivery date promised to the project leadership, marketing, customers, and other interested groups can only cause disappoint ment. Disappointment, however, may be too kind a word, since the plans that outside groups have made based on the product's expected delivery date will need to undergo serious modification. Other negative "fallout" also will be felt. Some expected customer sales will have disappeared, the next release will be available much later than planned (if at all), and a sometimes irreversible toll on the personal lives of the project team might have occurred. Whether the blame for lateness rests with the planners, the project technical people, or should be shared by both is not a point for debate here. The point is, the more a project schedule plan is thought out and mutually agreed to by all the key participants as being aggressive but *achievable,* then the greater the likelihood that *everyone* will win as the plan is executed successfully.

A project schedule plan should guide the project on a successful journey toward delivery of the product on time, within cost, with the expected quality, and without demoralizing the project's personnel. Three primary ingredients are necessary for developing an effective project schedule plan. The plan must:

- Be well thought out
- Have the commitment of the participants
- Be aggressive yet achievable

Many factors contribute to the successful implementation of these primary ingredients. Before focusing on the actual series of steps to follow in developing a project schedule plan, the first order of business is to understand the essential con- ;e$s that must be applied when developing a project schedule plan. These concepts are presented in the forthcoming sections. Afterward, the steps to developing a project schedule plan are described.

When to Start the Project Schedule Plan

While waiting for the needed information to develop the full project schedule plan, a near-term schedule plan should be established.

For a new project, broad, project-oriented schedules should be identified within weeks of the project's conception. Although these schedules will be preliminary, they will set the pace and level of expectation for the project until better schedules can be developed. A *near-term* set of schedules, however, should be available almost immediately. The near-term schedule plan should address the activities that must be worked within the next four to six weeks. The development of the full project schedule plan cannot be started seriously until after the product objectives have been completed and are available for review. The information in the product objectives becomes the basis from which a project schedule plan can be developed. The project schedule plan should be completed a short time after the high-level design and product specifications are approved. After the high-level design and product specifications activities have been completed, there should be sufficient understanding of the product to support a definition of a full set of schedules.

The progress being achieved on a project cannot be measured unless a plan first has been established against which progress can be measured.

Cyril Northcote Parkinson, British historian and author, once wrote that "Work expands so as to fill the time available for its completion." Schedules are critical to maintaining a healthy productivity rate for the members of a project, whether for a small group of people or for a large organization.

It is difficult to measure progress unless there is a plan to track against. Resist waiting until all imaginable details about the product content and the project plans are known before working the full project schedule plan. Each person has a variable productivity potential. Analogous to a variable resistor, a person has considerable flexibility to vary his or her productivity to match the needs of the task at hand. Aggressive but achievable schedules will help to harness the energies of project personnel so that an acceptable level of progress can be achieved throughout the duration of the project. Also, a "busy" person is much more content than a partially busy or idle person.

A top-down plan is developed without the participation of all project members and, therefore, should not be construed as a committed plan.

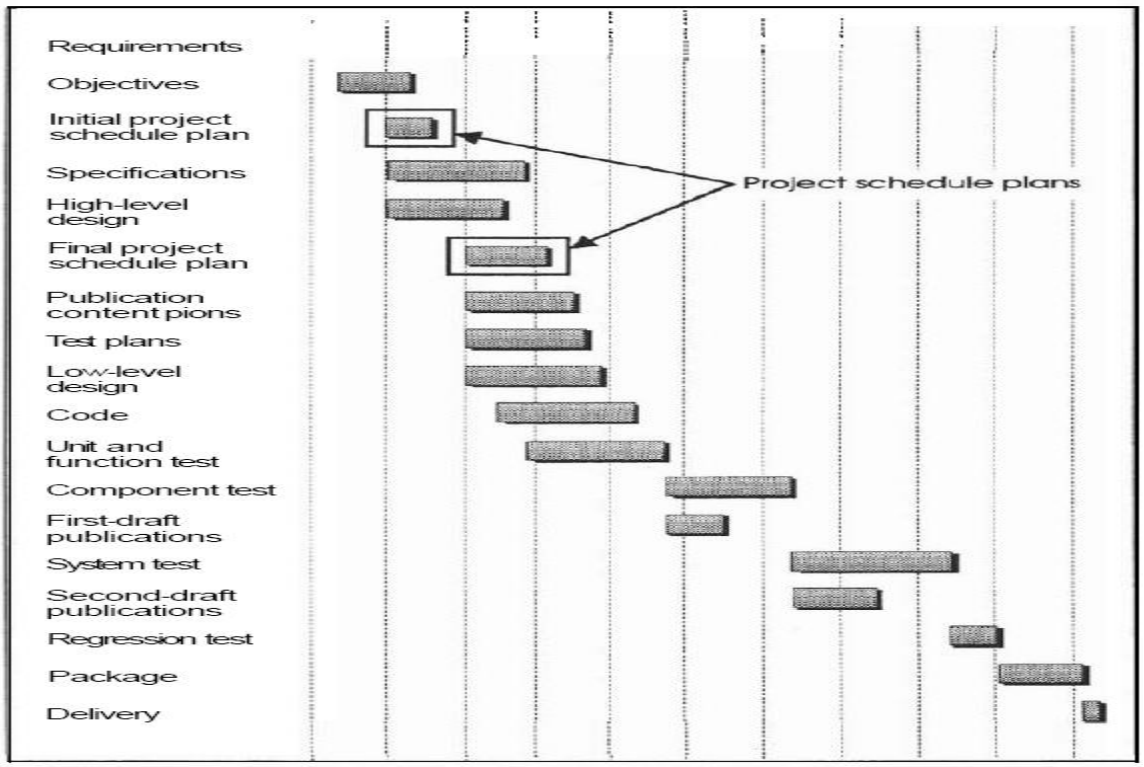
Project schedule plans come in two varieties: top-down and bottom-up. A top down plan typically is developed by one person (or a small subset of people on a project). The top-down plan lists the more significant activities of a project and suggests a set of schedules that might be achieved for it. A top-down plan does not involve the participation of all project members and, consequently, should not be conveyed to management as a committed schedule, but only as a target schedule. Top-down plans are strictly estimating techniques to help people get a better feel for the size, cost, and complexity of a project. Top-down planning is an essential business exercise and should be viewed as such, but with the understanding that the resulting schedules are typically high risk. In contrast to a top-down plan, a bottom-up plan is developed with the participa & son of virtually all members of a project. All activities are identified (not just the significant ones), along with the activity owners, the activity durations, and the dependencies among the activities. A bottom-up plan, if final, is ***the*** plan that should be committed to management. This chapter focuses on creating the bottom-up project schedule plan.

All participants in the product development cycle must feel ownership for their piece of the total project schedule. Only then will an individual's optimal productivity be possible. You might have observed instances where an individual did not feel ownership of the schedules for his or her activities. When the going gets tough, this individual might not put the extra effort forward. This hurts not only the individual but the project as well. Personal commitment from each participant is a must. Bottom-up planning helps to ensure the personal commitment from every member of a project.

Is there any value in creating both an initial project schedule plan and a final project schedule plan? Yes. A bottom-up project schedule plan usually is best developed in two stages: the initial project schedule plan and the final project schedule plan.

Never commit to a detailed schedule to build something unless the something is well defined.

As shown in Figure, the initial project schedule plan, hereafter called the *IPSP*, can be created after the product objectives have been written fully and are available for review. The IPSP should not be considered complete until the product objectives are approved. However, the IPSP, although a bottom-up plan, should not be committed to management. Why? Because it is based on product objectives and not on the more definitive product specifications. You should never commit to a detailed schedule to build something when, in fact, the something is not defined. The final project schedule plan *(FPSP),* as shown in Figure, can be started after the product specifications have been written fully and are available for review. However, the FPSP should not be considered complete and approved until the product specifications are approved. This is important because the product being built must be well defined before an FPSP can be considered complete and credible. The FPSP is the plan to commit to management.



Initial and final project Schedule plans

Project members need a plan to follow in order to achieve their greatest productivity.

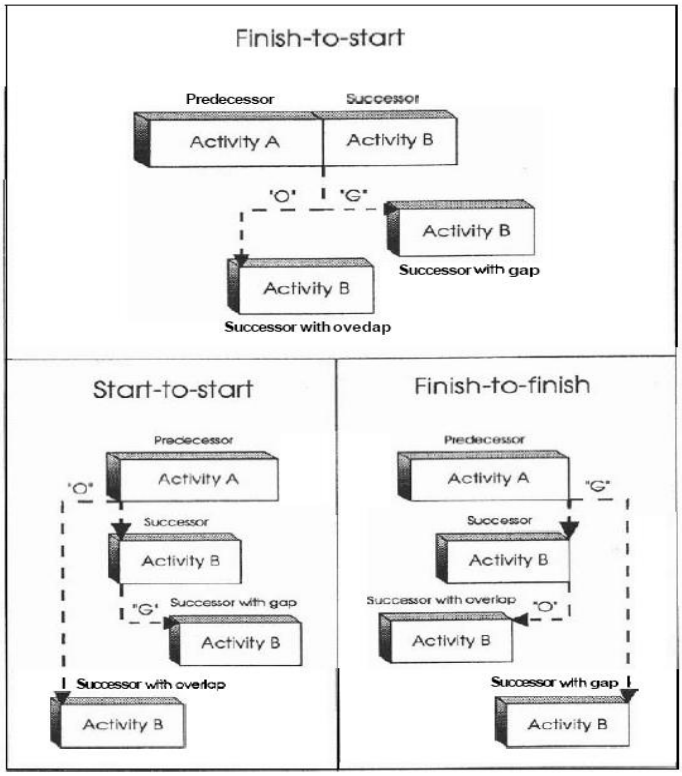
You might ask, "Why bother to even create an IPSP? Why not just wait for the product specifications to be available and then build the FPSP?" The reason is simple. If a project does not have a plan in place, even a relatively basic bottom-up plan, then precious time will be lost in getting to the FPSP. Why? Because everyone will be working hard on something, but not necessarily the **right** something. The members of a project need plans not only to ensure the right activities are being acted upon first but also to help the project members pace themselves.

For every day that a project delays in completing a plan, one-half day of poorly invested time is lost from the project that can never be regained.

Experience suggests that, once a project schedule plan is able to be developed, for every day that a project delays in completing the plan, at least one-half day is lost from the overall project cycle time. This is time that can never be recovered. If you delay by two calendar months in completing the plan, for example, you will have lost at least one calendar month of time from the project. People need a plan against which to pace their speed and priorities in order to achieve the most efficiency.

Types of Activity Relationships

Figure shows the three basic activity relationships and three variations per relationship that most projects require and most project management tools recognize. This means that at least nine variations of activity relationships are possible. These three basic activity relationships and their corresponding variations are briefly described here. **A** fourth is acknowledged.



Activity Relationships

1. **First to start**

   This is the most common relationship. In its simplest form, finish-to-start says that activity B will start *after* activity *A* finishes.

   Another variation is finish-to-start with a *gap* of *"G"* days. This means that activity B will start "G" days *after* activity *A* finishes.

   The third variation is finish-to-start with an *overlap* of "0" days. This means that activity B will start ***"0*** days *before* activity A finishes.

2. **Start – to – start**

   The most common form of start-to-start says that activity B will start *at the same time* as activity *A.*

   Another variation is start-to-start with a *gap* of " G days. This means that activity B will start "G" days *after* activity A starts.

   The third variation is start-to-start with an *overlap* of *" 0* days. This means that activity B will start *"0"* days *before* activity A starts.

3. **Finish – to – finish**

   The simplest form of finish-to-finish says that activity B will finish *at the same time* activity *A* finishes.

   Another variation is finish-to-finish with a *gap* of "G" days. This means that activity B will finish "G" days *after* activity A finishes.

   The third variation is finish-to-finish with an *overlap* of *" 0* days. This means that activity B will finish *" 0* days *before* activity A finishes.

4. **Start – to – Finish**

   This is a rarely used relationship and, therefore, is not shown in Figure. Start-to- finish says that an activity cannot finish **until** another activity has started.

Estimating the Duration of an Activity

Estimating the duration of an activity is perhaps the most difficult task in developing the project schedule plan. It is difficult because often you are estimating how long it will take to do something that you have not done before. Oh, sure, you have designed or coded or tested or written documentation before, but to the exact characteristics of this project? Probably not. Most new activities to be performed are unique enough to make estimating their duration something

other than an exact science. Let's look at some things to consider that can help in estimating activity durations.

Use a work breakdown structure.

When estimating the duration of an activity, break the activity into smaller work pieces and account for the duration of each work piece.

The more the estimated work is divided into smaller pieces, the better your estimate will be. For example, say you must prepare a product specifications document for review and you guesstimate it will take two months to write. If you spend a little more time to break up the activity into more discrete activities and then estimate the time to perform each of the discrete activities, your estimate will be more credible than the first approach. Specifically, you might decide that there will be 12 chapters and three people will each write four chapters. Then the three people will review each other's work for completeness, accuracy, and consistency and make any necessary changes. Then the product specifications will be distributed outside the three person team for review. It is much easier to estimate the time to perform discrete activities that last one week or less than to estimate activities that last several weeks or months. It is important to note that the planning performed here on the product specifications would fall under the category of low-level planning as discussed earlier. It is not recommended that the low-level activities be added to the intermediate-level project schedule plan. Doing so would only increase the complexity of the plan and unnecessarily make it more difficult to grasp and work with. Instead, the durations of the discrete activities should be tallied and the resulting value used in the project schedule plan for the activity called "product specifications prepared."

For best estimating results, break an activity into work pieces of approximately one week's duration.

The breaking of *big* activities into smaller and smaller activities defines the work breakdown structure (WBS). Every activity can be pared down to smaller discrete activities. For the most part, the intermediate-level activities that should make up the project schedule plan should be defined already in your organization's software development process. This is one level of the WBS. Each of these activities, in turn, can be broken into smaller pieces. And so on. If you own an activity, you probably will want to create the WBS to the level of detail that makes sense to best support our overall estimate. It is recommended that the WBS be defied to multiple discrete activities lasting one week (or less).

Use historical data.

Use historical data, if available, from past similar projects to help in estimating activity durations. For example, determine the productivity for performing high-level design, low-level design, coding, unit test, writing test scripts, writing a page of a user's manual, getting a design change request (DCR) approved, fixing and verifying a bug found in test, and so on. Include contingency buffer. In general, all activities should have contingency buffer built into them. There is a tendency to estimate the time required to complete an activity as if there will be absolutely no interruptions. The buffer ensures there is time to complete the activity and perform other activities or interruptions that are often overlooked. If a comprehensive list of ground rules (see the final bullet) has been accounted for, the buffer might be minimal or perhaps not be necessary. (For more on the subjects of buffers, see "Planned Contingency Buffers" later in this chapter.)Solicit others as a sounding board. Use others as a sounding board to help add credibility to your estimates. Any problems identified while scrubbing your estimates can save many headaches later as you are implementing your activities and held accountable for meeting your estimates. Develop a set of ground rules from which to estimate.

A project should have a common set of ground rules from which everyone plans.

When the project members are assembled to develop their pieces of a total project schedule plan, it is important that everyone begin the journey on the same footing. That is, a common set of assumptions, or round rules, needs to be developed.(These ground rules typically are developed by the project schedule coordinator as' part of Step 3 of developing a project schedule plan and are delivered to the project members in Step 4.

Examples of items that might appear on the list of ground rules include:

No overtime

First shift only

Account for vacation and holidays

State activity durations in workdays (not weeks)

Account for skill level of team members

Account for education training needs

Account for participation in design and code inspections

Account for participation in reviewing project documents

Account for participation in project meetings, such as team/project tracking meetings

Account for identify, recording, and displaying various quality- and process related measurements

Account for travel time

Account for availability of hardware, tools, and the like that are required

Account for realistic projections of personnel staffing

Base estimates to design, code, and test on productivity rates of; explain any deviations

Include an X percent contingency buffer in each activity

Account for ongoing support of products that have already been delivered to customers

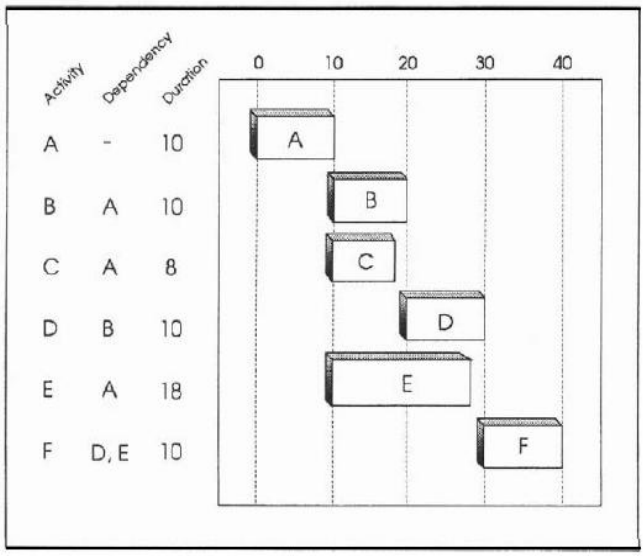
List any special assumptions made that are in addition to those in the ground rules

Critical Path

To shorten a project's duration, focus on what can be done to those activities that make up the critical path.

You should know the activities that make up the **critical path** of the project schedule plan. The critical path is the sequence of activities that, collectively, define the starting and ending dates for the project. The path of these activities has no slack time (excess time) in accomplishing the schedule. Said another way, if you were to reduce the total project's duration, the activities in the critical path are the initial ones upon which to focus. Furthermore, if you are to achieve your committed schedules, it is the activities on the critical path that typically need the most immediate attention. Typically 20 percent or fewer of all a project's activities are on the critical path. Activities (including documents) commonly in the critical path are the product requirements, product objectives, product specifications, high-level design, low-level design, code, unit test, function test, component test, system test, and packaging and stocking the product for delivery to customers. These activities are described in the sample software development process defined in Chapter 1. Keep in mind that the activities in the critical path can be different in your project depending on many factors, such as the software development process you employ and the characteristics of the product you are building. You might ask, "That's a lot of activities. What activities are missing from this list?" **A** lot. Some examples are: a "quality plan," build and integration plan, various test plans, writing test scripts for the various tests, publication content plans, writing the product publications drafts, and many other activities. However, it is important to note that these activities can be in a project's critical path, although typically they are not. Figure shows a simple example of a critical path in a primitive network. The example depicts six activities and their corresponding dependencies and durations. The critical path is the serial flow of activities A-B-D-F. To shorten the critical path by gone day, the day must be removed from among the A, B, D, and F activities. Activities C and E each have slack time of 2 days. Note that, if activity D were reduced to seven days, a new critical path would emerge: **A-E-E** This is because activity **E** now would end one day after the combined serial durations of activities B and D. **A** project might easily have 200 or more activities defined in its network.

Reducing the critical path on most projects is not a trivial exercise and can, in fact, be downright arduous.



Example of critical path A-B-D-F

What Should Be Tracked?

Obviously, a man's judgment cannot be better than the information on which he has based it. - *Arthur Hays Sulzberger, American newspaper publisher*

Before we list the items that should be tracked, let's take a look again at the primary objectives for tracking:

1. Identify potential problems before they occur.
2. Put recovery plans in place before unrecoverable harm occurs.

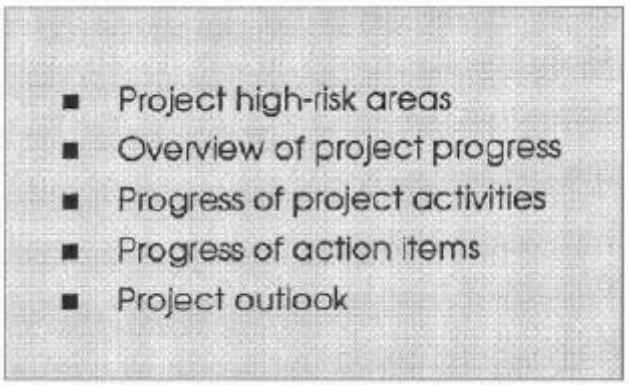
Notice that the focus is on identify and resolving problems. This means that

there are two general categories of information to be tracked:

Plans for project activities

Known problems

Project activity plans are tracked so that potential problems can be identified as early as possible. Once problems are identified, they are tracked individually to ensure they are resolved. "But," you might ask, "what exactly should be the subjects of focus at project tracking meetings?" The major subject areas for presentation and analysis at project tracking meetings are listed in Figure 5.1. These subject areas are so key that each will be discussed in a separate section that follows. However, before proceeding to these sections, three key terms used throughout this chapter must first be introduced: project tracking team, project tracking team meeting, and project tracking team leader. The project tracking team, or *PTT*, is a group of project members who routinely meet to present their progress and problems. On a small project, every project member.



Topic to be tracked

might participate. On a larger project, a subset of members attend who, collectively, can represent all the activities of the project. When the PTI members get-together, it is called a project tracking team meeting, or PTT meeting. The primary purpose of PTT meetings is to serve as a communications forum for the project members to accomplish the three items listed in Figure. The project tracking team leader, or PTT leader, is the person chosen to lead the PlT. The PTT leader is fully responsible for coordinating all the activities necessary in creating and using a successful tracking and problem-management process for use throughout the project's development cycle. The project schedule coordinator (PSC), described in, and responsible for developing the project schedule plan, might be chosen to act as PTT leader. (More can be found on the role of the PTT leader later in "The Steps in Tracking a Project Schedule Plan" section.)

Project Outlook

After the project status has been presented, it is helpful to predict the project's progress during the coming 30 days (or whatever length of time feels right for your project). The outlook is project-oriented, not activity-oriented as discussed in an earlier section. This information might be presented by assurance but could be handled by the PTT leader. In either case, both parties should work together so as not to be surprised by the information and the conclusions. The project outlook chart(s) should be prepared before the PlT meeting.

However, because the latest status is presented during the PTT meeting, an attempt should be made to adjust, real *time*, the project outlook as new information becomes available throughout the meeting. It is recommended that the project's outlook include the assignment of a ***risk value*** that attempts to gauge the likelihood that the project will achieve at least the following major milestones:

Next major milestone

Public announcement date

Product delivery date

Definitions of risk values that can be assigned vary widely in practice. An example of one approach that might be used is the assignment of any one of three risk values: low, medium, and high.

Low

A condition where the risk consequence would have insignificant impact unplanned schedules, function, costs, and quality; and/or the probability of occurrence is sufficiently low as to cause no concern. Normal monitoring and control is required to ensure continued low-risk status.

Medium

A condition where the risk consequence would have a noticeable and disturbing impact on planned schedules, function, costs, or quality; and/or the probability of occurrence is high enough to be of concern. The level of risk requires:

--Close control of all contributing factors.

-Establishment of a recovery plan.

-Efforts to identify a contingency plan.

High

A condition where a high probability of occurrence and/or the consequence would have significant impact on the overall project (schedules, function, costs, or quality). The level of risk requires:

--Close control of each high-risk area.

--Close control of each contributing factor.

-Establishment of a recovery plan.

-Identification of the contingency plan and the factors that, when reached, will trigger its implementation.

Assigning a risk value to a project has little significance if the project has no committed project schedule plan against which to evaluate.

Project

The assignment of a risk value is a method of predicting, based on historical and current data, the likelihood that the project schedule plan will be achieved for certain upcoming major milestones. If a project has no approved project schedule plan, assigning a risk value has little or no significance; if you don't know where you are supposed to be, then how can you possibly gauge your likelihood of getting there?

When Should Tracking Occur?

PTT meetings should be conducted weekly to achieve optimal productivity against the approved project schedule plan.

It is strongly recommended that the PTT meeting occur weekly throughout the full project life cycle. Meeting more frequently can cause unnecessary and undesired overhead that can interfere with the overall productivity of both the PTT members and all the project's members. Meeting less frequently than once a week often will allow problems to fester unchecked for longer than the project can bear.PTT meetings should be conducted on the same day, the same time, and the same place. Participants who are required at tracking meetings are usually very busy. They need to be able to plan their time and manage their calendars. Regularly scheduled tracking meetings allow participants to plan their other activities and to prepare for the meetings. Part of the preparation involves gathering the necessary status information for their own area's work activities and also might involve negotiating problem resolutions with other groups.

& **Lesson: PTT meetings serve as the primary driving force of a project.**

Another important reason for regularly scheduled tracking meetings is people’s need to pace themselves for effective productivity. Most people work more intelligently and with more conviction when they are being measured against their commitments. Infrequent or irregular tracking typically results in occasional spurts of improved productivity. When people's progress is tracked at frequent and predictable intervals, they are conditioned to maintain a fairly constant and predictably high level of productivity. Furthermore, this action helps to drive a critically needed level of discipline into an organization. In fact, the PTT meeting, and its derivative actions, serves as the primary driving force behind the project. It is suggested that PTT meetings occur on Tuesday, Wednesday, or Thursday of each week. Mondays are avoided to allow the PTT members sufficient time to gather status and update the appropriate charts for the PTT meeting. This also allows project members who worked the weekend to include reporting on their weekend progress more easily.

Fridays should be avoided for two primary reasons. One reason is that many people like to take Fridays off to create long weekends. Also, many holidays fall on Friday. The Friday disruption means that the PTT meeting members might begin to view the PTT meeting as an "optional" event, which it is ***definitely*** *not.* The other reason why Fridays should be avoided is that a PTT meeting is very likely to cause certain problems to require immediate attention. You guessed it: Friday afternoon is a bad time to get news that will interfere with a person's planned personal weekend time.

PTT meetings are most effective if conducted on a Tuesday or a Wednesday-and the following day is reserved for work meetings and escalation meetings.

The author favors Wednesday as the preferred day for the PTT meeting. It is strategically placed in a week and allows ample time both to prepare for the meeting and to recover from its aftermath. Furthermore, it is recommended that all-day Thursday be left open on the calendars of all the project's members. Thursday is reserved for work meetings and for escalation meetings (discussed in an upcoming section) for the problems identified as needing immediate attention in the PTT meeting. By reserving the day after the PTT meeting, the problems are assured of getting attention immediately. Use of the day after the PIT meeting for this purpose helps to avoid problems lingering from week to week and helps to force a structured method upon a project to ensure that these project-level problems get the attention they deserve and require. Should PTT meetings occur during mornings or afternoons? Most people seem to favor mornings because there is a belief that most people are more alert in the morning than they are in the afternoon. Also, there is a belief that afternoon PTT meetings are more likely to have to compete with other events for the full attention of the PTT members. The notion is that, if the meeting is conducted in the afternoon, then PTT members are more likely to come to it late, or leave early, or skip it altogether due to other demands on their limited time, demands that appear to increase as the day progresses.

Who Should Attend Tracking Meetings?

The PTT leader identifies the areas of the project that must be represented at PTTmeetings. The PTT leader might even recommend the specific people from those areas which should be represented in the PTT. The members chosen should be approved by their managers. - -

If the project membership is small, say five to 10 persons, all project members might participate on the PTT. For somewhat larger projects, say 10 to 25 persons, the lead person in each group (planning, development, test, publications, assurance, and others) might be the appropriate representative at the PTT meeting. For projects larger than 25 members, PTT members might be team leaders (even if more than one per group) and persons performing independent roles who need to represent themselves (the sole person interfacing to a vendor group, for example). Collectively, the project members chosen to attend the PTT meeting must be able to represent the status of all the project's activities and action items.

In the interests of encouraging accountability at the lower levels of an organization, PTT members typically should not include management personnel.

Project Tracking:

Due to the popular and vital concept of empowerment and moving the authority to the point of responsibility, management personnel typically should not be members of the PTT. However, managers will find the PTT meetings to be the best source of project status and often might choose to attend. An optimal number of full PTT members (versus occasional guests) seems to be in the realm of 10 or fewer people; however, larger meetings are possible and, in fact, are necessary for larger projects.

PTT members must be able to represent their areas adequately.

A project leader is usually the best person to attend tracking meetings. However, there are times when some things must be delegated. In these cases, project leaders should work with their representatives to give them the authority and support they need. It is essential that PTT members be able to represent their area adequately by fielding expected questions, negotiating solutions to problems, and making commitments for their area. There will be a serious, negative impact on the success of the PTT meeting if PTT members weakly represent their areas. In the section "What Should be Tracked?" the major subjects of focus at tracking meetings were discussed and listed in Figure 5.1. They are repeated here for convenience.

The PTT meeting agenda is centered around these major subjects.

Project high-risk areas

Overview of project progress

Progress of project activities

Progress of action items

Project outlook

A well-planned PTT meeting agenda can have a significant positive impact on the control maintained, not just for the meeting but for the project as well.

The following guidelines will help you develop an effective meeting agenda. Figure shows a sample PTT meeting agenda format that will help to demonstrate these guidelines. After the guidelines are presented, the sample agenda in the figure will be examined closely. Each item to be presented should have a person's name assigned (versus no name or an organizational or functional name). This technique helps to drive ownership and accountability upon those persons responsible for coming to the meeting fully prepared. It also helps to avoid: "What? I didn't know you expected me to present that!"

PTT Meeting Agenda: 04/15/9x

Time	Item	Presenter
8:00	Project high-risk areas	Turner
8:15	Overview of project progress	Turner
	Progress of project activities	
8:20	Change control	Yamato
8:30	Low-level design and code	Grow
8:40	Unit and function test	Miller
8:45	Component test	Berry
8:50	System test	Berry
8:55	Publications	Vandegrift
9:05	Performonce	Vila
9:10	Usability	Smith
9:15	Progress of action items	
	41	Casey
	37	Kaptsan
	5, 12, 19	Madison
	38	Miiler
	39	Short
	27, 44	Yamato
9:30	New action items	Ail
9:45	30-day outlook	Rosenman
9:50	Plan work/escalation meetings	Turner, oil
10.00	Adjourn	

Sample Project Agenda Format

Each item and presenter in an agenda should have a time limit stated. For example, the "project high-risk areas" might be presented in 15 minutes. (If this appears too long, remember that these are the most critical project problems in the project. The PTT members need to comprehend fully the impact (potential or otherwise) that these yet unresolved problems have on the project.)The time limit for presenting progress for a given project activity will, of course, vary widely for each activity. However, if each person is given somewhere between five and 10 minutes to present the status for all his or her activities, this might be adequate. Depending on the activity and its complexity, either five or 10 minutes is chosen. By contrast, the time to present each action item might be one to two minutes. The order of subject areas should follow some logical pattern. For example, when the "progress of project activities" is being presented, the sequence of activities might be in the order that the project is proceeding; that is, present progress on the high-level design before the low-level design, the low-level design before the coding, and so on. Furthermore, if an iterative development process is being followed, present progress on the first iterative piece before addressing the second piece, and so on.

Sound obvious? Maybe, but this technique of logical sequencing is not always followed. The discipline with which the PTT meeting is planned and executed will have a marked impact on the discipline exercised throughout the project. The sequence of presenters can be adjusted for logical reasons, but it should be predictable and planned. For a case in which a person from outside the project has something to report each week, but has no need to sit through the entire meeting, this

person could present at the beginning of the meeting and then leave. The **PTT** meeting agenda should be available several days before the meeting.PTT members need time to prepare for the meeting. Also, they need to be certain of the areas that they are responsible for addressing. The **PlT** meeting agenda helps the PTT members know what is expected of them. Armed with these guidelines, let's examine the sample project agenda format in Figure. Project high-risk areas and overview of project progress. Notice the same person is presenting both items. This would imply that "Turner"is the PTT leader.

Progress of project activities

Because low-level design is the first development activity presented, this means that the product requirements, product objectives, product specifications, and high-level design have completed and are now under change control. The inclusion of “change control" in the agenda means that someone (Yamato) must address the current status of **requirements/function/design** changes being proposed for the project. Although there are many names used to describe these change requests, some projects call them: requirements change requests (RCRs), design change requests (DCRs),or simply change requests (CRs).Notice that the development activities appear in the list in the order they are performed in the development cycle--until "publications" appears. The activities centered around publications, performance and usability relate across many areas of the development cycle and, therefore, usually are presented after the so-called development activities.

The progress of activities performed by vendors and subcontractor6should be tracked just a5 routinely as that of other members of a project.

Activities can be added, merged, or deleted from this list. This depends on the size and scope of your project. For example, if a vendor is performing the low-level design, the progress for that activity must be presented. It could be presented underflow-level design or as a separate activity referred to simply as "vendor" or "componentABC." No allowances for skipping status should be made for vendors or sub164Managing Software Development Projects contractors. Their activities should be tracked on the same regular basis as if it were being performed under the direct control and proximity of the product manager.

Progress of action items

Notice that start times are not shown for each presenter. This is because one to two minutes usually should be sufficient for presenting status for an action item. Setting time increments to this level of accuracy adds no value to the meeting agenda. A typical way to decide the order of presenting the action items is by alphabetical order of the action item owner's last name.

If a person owns more than one action item, then his or her action items might be presented in chronological order as shown

in the agenda.

New action items

Anyone with a new action item can log it now. It is expected that each new action item will have a person assigned to own it and, if possible, a target close date committed before this portion of the PTT meeting ends.

30-day outlook

Because the name (Rosenman) is different from the name on the "project high risk areas" and "overview of project progress" items (Turner), it might be assumed that assurance (or a person with an equivalent perspective) is presenting this item. However, the PTT leader is an alternative choice to address this item.

Plan work/escalation meetings

The PTT leader spends the last moments of the meeting declaring what project activities and action items require special attention over the next two to three days. For best results and proactive project control, the meetings to address these special project areas should be scheduled now, preferably for the following day. These meetings become priorities within the project.

Scheduling the Tracking Meeting

The meeting room should be large enough to accommodate the expected participants and be comfortable enough for the **PTT** members to spend the designated length of time. Ensure the necessary accessories are in the room, such as projector (with backup lamp bulb), wall boards to write on, flip-chart paper and holder, blank transparencies, markers, Scotch tape, and so on.

Schedule the PTT meeting room for 30 minutes longer than the planned meeting.

Project Tracking:

Reserve the room for the planned duration of the PTT meeting, plus another half-hour. Whv? Because sometimes a topic will be so pressing that a few more minutes past the scheduled meeting time is required. Attempt should be made to avoid extend in^ the meeting time (after all, it can wreak havoc with everyone's cal- ., -endar integrity), the extra minutes can be a blessing at times. Don't forget to reserve the meeting room for the same day, time, and place each week for the duration of the project. You might want to consider arranging the chairs and tables in a manner that might help encourage a participative environment. For example, if only a handful of people are meeting, perhaps they can all sit around a single table where they have ready eye contact with one another. If the meeting has around 20 members, perhaps the tables can be arranged in the shape of a horseshoe. The opening in the "horseshoe"is reserved for the

transparency projector and writing boards. This arrangement allows everyone to have direct eye contact with every other person. The meeting invitation should be distributed with sufficient lead-time notice and should be distributed on a routine schedule. The PTT leader should ensure that the meeting has the full support of the product manager and that the invited participants are keenly aware of the high priority of this meeting as compared with their other duties.

A simple set of PTT ground rules will establish expectations.

A short, simple set of *PTT meetingrolmd* rules will help PTT members to understand what is expected of them at the meeting. These ground rules should fit on a single sheet of paper and be displayed at meetings when needed. An example of a set of ground rules is:

Come on time.

Not five or 10 minutes late.

Come prepared.

Not just in your head, but also with the media (transparencies, diskettes for computer screen projections) from which you will report your status and which you will provide to the PTT leader after you have presented.

If behind on an activity, then address:

-Why the activity is late.

-What other areas are/might become impacted.

-What recovery plan will be/is in place.

-Whether you need help.

It is in the best interests of the project and its members for a project member to seek help when needed.

Many project members avoid asking for help. The most effective projects encourage its members to request help when it is needed. Time-consuming problems are to be resolved outside of the meeting. If a problem can be resolved within two minutes, do it. Why? Because there is a unique opportunity with the breadth of groups and skills represented at the meeting to resolve some problems easily. However, if more than two minutes is required, log the problem for resolution outside the meeting."Raise hand" if the meeting is perceived to be off course. This can be a helpful mechanism for PTT members to employ in reminding thePTT leader and others that the meeting seems to be on an unproductive tangent. ThePTT leader can then decide to move on or complete the discussion. This form of empowerment helps to keep meetings in check.

Encouraged: Open and candid status reports and discussion.

This is essential for a successful, productive meeting.

There is a tendency for presenters to make their status appear more positive than it is. Participants should feel encouraged to present complete and accurate status, even if the news is negative. The project's health is in far better shape if the project's members are cognizant of the project's problems. Problems can only be resolved if they are visible and being worked. Don't be fooled by the simplicity and obvious subjects in this example set of ground rules. These are problems that commonly plague PTT meetings (as well almost other types of meetings).

A project's leadership must fully support the PTT meetings and related events if a project is to be as successful as possible.

There is no point in tracking a project if the project's leadership (including thePTT leader) will not enforce the ground rules. PTT members must recognize the critical importance of their role in PTT meetings in helping to ensure successful tracking meetings. Furthermore, the project leadership must support the PTT members' need to have time to gather, prepare, and present status, and to work on resolving problems. The PTI members must have a clear understanding of the priority that this task has relative to their other duties. The entire organization wants to be successful. Everyone looks to the project leadership to provide the environment necessary to attain this success. The disciplined tracking of a project helps to create an environment for success.

Recovery Plans

Recovery plane are for fixing a problem quickly and limiting the damage that can result.

An effective tracking process doesn't just discover problems, but also ensures that recovery plans are implemented and tracked before the problems have a chance to cause lasting damage to the project. The first rule of thumb is to protect the overall integrity of the project schedule plan. The goal of recovery plans is to **fix** the problem quickly and limit the damage to the smallest area. For example, an activity might be completed late by one week, but if the successor activities can maintain their completion dates and the next major milestone date is preserved, the recovery plan will probably be viewed as successful. Recovery plans should be put in place when a problem arises with an activity that has the perceived likelihood of causing unacceptable harm to some aspect of the project. This is not a time to ignore the problem or to delay in making a decision to begin the recovery. As Thomas J. Watson, Jr. had been known to say: "Better to do something--even the wrong thing-than to do nothing at all."

Recovery plans should address the following items:

Identify the owner of the recovery plan.

Identify the sequence of activities that must occur to complete the resolution.

Determine the dates when each activity of the plan will be started and completed, and identify the dependencies of each activity.

Ensure that the appropriate people or groups approve the plan. This includes those groups who have a dependency on the successful implementation of the plan.

The best recovery plans don't just show how the activity in trouble recovers; they show how damage is contained for those activities dependent on the troubled activity.

Also, recovery plans should be included as items tracked in the PTT meeting and should be presented when the relevant activity is being reviewed.