

PROJECT PLANNING: AN ANALYSIS

Dr.S.S.Riaz Ahamed

Principal, Sathak Institute of Technology, Ramanathapuram,India.

Email:ssriaz@ieee.org, ssriaz@yahoo.com

ABSTRACT

The purpose of the Planning stage is to analyze the project in terms of work breakdown, cost, resources, and timing. At the end of this stage all team members should be clear on the sub tasks and deliverables with the project, the time constraints they are working too and the roles and responsibilities that are expected from them. Software Project Plan defines what the work is, and how this work can be completed. This plan is developed at the beginning of the software project and is continually refined and improved as the work processes. It can be useful to management as a frame work for review and control the process of developing the software. Additionally, the Software Project Plan can define each of the major tasks and estimate the time and resources that are required to complete these tasks.

Keywords: Work Breakdown Structure (WBS).

1.1 SOFTWARE PROJECT PLANNING

The software planning begins with a statement of the work to be performed and other constraints and goals that define and bound the software project (those established by the practices of the Requirements Management key process area). The software planning process includes steps to estimate the size of the software work products and the resources needed, produce a schedule, identify and assess software risks, and negotiate commitments. Iterating through these steps may be necessary to establish the plan for the software project (i.e., the software development plan).

This plan provides the basis for performing and managing the software project's activities and addresses the commitments to the software project's customer according to the resources, constraints, and capabilities of the software project.

During planning, a project is split into several activities which may be pursued in parallel or in series. The questions are

- How much effort is required to complete each activity?
- How much calendar time is needed?
- How much will the completed activity cost?

The Plan define a strategic model to devise the project more effective and execute based on the strategic plan. Though there are many strategic models are recommended, each has its own Importance.

Milestone Planning Process

This process focuses on identifying the key project objectives or milestones. Each milestone defines a condition or state that the project must occupy to meet the final objective. The milestone planning process forms a powerful method of structuring a project and providing a framework for progress reporting.

The Milestone Plan Deliverable

The milestone plan should list all the milestones identified as necessary to meet the project requirements. Each milestone must exhibit the following qualities:-

- Be measurable – Must include a quality statement.
- Be robust to changing circumstance – A unique state that is not revisited.
- Be specific about what is needed not how it is to be achieved.

Resource Analysis Process

The stage is concerned with identifying and quantifying all resources considered necessary for successful delivery of the project. .

- Identify all resource requirements
- Quantify resource requirements.

Completion of this stage leads to the Resource Plan deliverable.

Resource Plan Deliverable

As a guideline the Resource plan should list the following.

- Internal manpower – describe the internal disciplines required eg) hardware, DSP, RTE, project management, procurement. Do not attempt to show time estimates at this stage, this is done in the activity planning stage.
- External resources – details any resources required that are not present within Concept eg) specialist skills, approval bodies and detail a supplier for each.
- Tools & Equipment – details specialist test or development equipment required. Detail which organisation is responsible for providing the equipment or list suppliers if externally supplied.
- Training Requirement – detail training requirements, training supplier and candidate for training.

Risk and Contingency Analysis Process

An important aspect of effective planning is the analysis of risk inherent in the project. The objective is to identify all critical areas of risk and then to develop procedures and contingency plans to minimize the impact of strategic and conditional risks. Contingency analysis consists of examining the risks that threaten the critical success factors of the project, ranking the issues to focus management effort and developing responses. This process leads to the production of the Risk Matrix.

Risk Matrix and Contingency Deliverable

The Risk Matrix deliverable has the responsibility for.

- Identifying all risks considered.
- Ranking the risks in terms of impact to the project and the probability of occurrence.

This is achieved through the use of a Risk Matrix diagram

Activity Planning Processes

The purpose of the activity planning stage is to :-

- Identify the activities required to achieve the milestones.
- Assign roles and responsibilities to project team members.
- Estimate activity durations and dependencies.
- Schedule the activities.
- Produce accurate costing for the project.
- Establish reporting and communication levels.
- Establish the acceptance criteria.

Project Specification Deliverable

This document details how the requirements of the project are to be implemented. It should outline the skill bases, technologies and strategies employed to deliver each of the milestones. It should also define any limitations / caveats that have had to be made to the final deliverable.

Roles and Responsibility Matrix Deliverable (Internal)

This is an internal document that references the responsibility of each team member with activities. It should also detail exactly what the responsibility level is such as advisory, executor, consultant.

The Project Plan Deliverable

The project plan shows tasks, milestones, dependencies and schedules. It is usually implemented by way of the Gantt chart or Pert chart. The project plan should clearly show the effort requirements for each task and the team members allocated to the task. This plan is used in multi-project environments as a method of controlling and balancing resource requirements. The project plan is the starting point for the live project plan that is used in the Driving stage.

Costing Plan Deliverable

The costing plan should show the breakdown and explanation of the costs of the project in terms of effort and equipment. It is recommended that spreadsheet package be used for this process

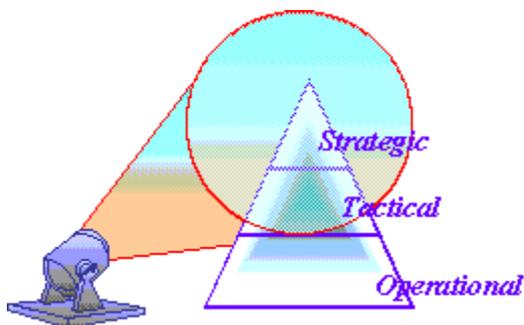
The Communication Plan Deliverable

This deliverable is responsible for :-

- Agreeing the methods and contents of the communication between project manager and customer contact.
- Agreeing the frequency of communication.
- Agreeing escalation procedures.

1.2 STRATEGIC PLAN

A Strategic Plan is a planning tool for management of larger companies to formulate their high-level business strategy. It starts with the brainstorming sessions that are part of the Strategic Planning Workshop. After the Strategic Plan has been conducted, more meaningful "business plans" can be conducted for each division, i.e., division plans or line of business plans.



The Workshop to develop the Strategic Plan involves executives and decision-makers in the analysis of the business's environment and evaluates the organization's competitive advantage. It helps planners to understand the environment, corporate culture and core competencies.

1.2.1 Benefits of a Strategic Plan

It identifies the best high-level strategies by:

- Addressing and prioritizing external opportunities and threats to the firm.
- Addressing and prioritizing internal strengths to be capitalized on and weaknesses to be overcome.
- Crystallizing individual executive's plans and preferences for the company.
- Bringing everyone together on issues of what types of activities the firm potentially will and will not do, now and in the future.
- Defining the precise mission of the business.
- Identifying key objectives and strategies for achieving the mission.
- Formulating strategies that will achieve/maintain a competitive position.

1.3 PLANNING

It is found that software projects span a broad range of application domains. It is valuable but risky to make specific planning recommendations independent of project context. It is valuable because most people in management positions are looking for a starting point, a skeleton they can flesh out with project-specific details. They know that initial planning guidelines capture the expertise and experience of many other people. Such guidelines are therefore considered credible bases of estimates and instill some confidence in the stakeholders.

Project-independent planning advice is also risky. There is the risk that the guidelines may be adopted blindly without being adapted to specific project circumstances. Two simple planning guidelines should be considered when a project is being initiated or accessed.

Default distributions of effort and schedule by phase

| Domain` | Inception | Elaboration | Construction | Transition |
|----------|-----------|-------------|--------------|------------|
| Effort | 5% | 20% | 65% | 10% |
| Schedule | 10% | 30% | 50% | 10% |

An important point here is that this is cost allocation, not effort allocation. To avoid misinterpretation, two explanations are necessary.

- The cost of different labor categories is inherent in these numbers. For example, the management, requirements, and design elements tend to use more personnel who are senior and more highly paid than the other elements use. If requirements and design together consume 25% of the budget (employing people with an average salary of \$100/hour), this sum may represent half as many staff

hours at the assessment element, which also accounts for 25% of the budget but employs persons with an average salary of \$50/hour.

- The cost of hardware and software assets that support the process automation and development teams is also included in the environment.

Top Down Planning

The first is a forward-looking, top down approach. It starts with an understanding of the general requirements and constraints, derives a macro-level budget and schedule, then decomposes these elements into lower level budgets and intermediate milestones. From this perspective, the following planning sequence would occur:

- The software project manager (and others) develops a characterization of the overall size, process, environment, people and quality required for the project.
- A macro-level estimate of the total effort and schedule is developed using a software cost estimation model.
- The software project manager partitions the estimate for the effort into a top-level WBS using guidelines. The project manager also partitions the schedule into major milestones dates and partitions the effort into a staffing profile, using guidelines. Now there is a project-level plan. These sorts of estimates tend to ignore many detailed project-specific parameters.

At this point subproject managers are given the responsibility for decomposing each of the WBS elements into lower levels using their top-level allocation, staffing profile, and major milestones dates as constraints.

The advantage is that the top managers, who are the most knowledgeable about the firm as a whole, drive the development of the plan.

The software planning begins with a statement of the work to be performed and other constraints and goals that define and bound the software project (those established by the practices of the Requirements Management key process area). The software planning process includes steps to estimate the size of the software work products and the resources needed, produce a schedule, identify and assess software risks, and negotiate commitments. Iterating through these steps may be necessary to establish the plan for the software project (i.e., the software development plan).

This plan provides the basis for performing and managing the software project's activities and addresses the commitments to the software project's customer according to the resources, constraints, and capabilities of the software project.

Bottom-up-Planning

The second perspective is a backward –looking , bottom-up-approach. This approach begins by analyzing the micro-level budgets and schedules, then sum all these elements into the higher level budgets and intermediate milestones. This approach tends to define and populate the WBS from the lowest level upward. The following planning sequence would occur:

- The lowest level WBS elements are elaborated into detailed tasks, for which budgets and schedules are estimated by the responsible. WBS element manager. These estimates tend to incorporate the project-specific parameters in an exaggerated way.
- Estimates are combined and integrated into higher level budgets and milestones. The biases of individual estimators need to be homogenized so that there is a consistent basis of negotiation.

Comparisons are made with top-down budgets and schedule milestones. Gross differences are assessed and adjustments are made in order to converge on agreement between the top-down and the bottom-up-estimates.

The primary advantage is that the people closest to the operating system, customers, and suppliers drive the development of the plan. The hierarchy of the planning process is from bottom and the direction of movement of the game plan is upwards.

The following are to be born in mind:

- The software engineering group reviews the allocated requirements before they are incorporated into the software project.
- The software engineering group uses the allocated requirements as the basis for software plans, work products, and activities.
- Changes to the allocated requirements are reviewed and incorporated into the software project.

That is in this type of planning, the modules are designed first and then it is been incorporated in the main module.

Which is better top-down or bottom-up planning depends upon the goals

- Software estimates are documented for use in planning and tracking the software project.
- Software project activities and commitments are planned and documented.

The Iteration Planning Process

Planning is concerned with defining the actual sequence of intermediate results. Planning the content and schedule of the manor milestones and their intermediate iterations is probably the most tangible form of the overall risk management plan. An evolutionary build up plan is important because there are always adjustments in build content and schedule as early conjecture evolves into well-understood project circumstances.

| Engineering Stage | | Production Stage | |
|-------------------|-------------|------------------|------------|
| Inception | Elaboration | Construction | Transition |

Feasibility Iterations Architecture Iterations User Iterations Product releases

- A generic build progression and general guidelines on the number of iterations in each phase are described next. Iteration is used to mean a complete synchronization across the project, with a well-orchestrated global assessment of the entire project baseline. Other micro-iterations, such as monthly, weekly, or daily builds, are performed en route to these project-level synchronization points.

The general guideline is that most projects will use between four and nine iterations. The typical project would have the following six iterations profile.

- One iteration in inception: an architecture prototype
- Two iterations in elaboration: architecture prototype and architecture baseline.
- Two iterations in construction: alpha and beta releases.
- One iteration in transition: product release.

Project Planning:

Even though good planning is more dynamic in an iterative process, doing it accurately is far easier. While executing iteration N of any phase, the software project manager must be monitoring and controlling against a plan that was initiated in iteration N-1 and must be planning iteration N+. The art of good project management is to make trade-offs in the current iteration plan and the next iteration based on objective results in the current iteration and previous iterations. This concept seems and is, overwhelming in early phases or in the project s that are pioneering iterative development. But if the planning pump is printed successfully, the process becomes surprisingly easy as the project progresses into the phases in which high-fidelity planning is necessary for success.

Aside from bad architectures and misunderstood requirements, inadequate planning (and subsequent bad management) is one of the most common reasons for project failures. Conversely, the success of every successful project can be attributed in part to good planning. Planning, requirements and architecture are three attributes that needs to be highlighted. The end products associated with these perspectives (a software development plan, requirements specifications, and architecture description document) are not emphasized. On most successful projects, they are not very important once they have been produced. They are rarely used by most performers on a to-day basis, they are not very interesting to end users, and their paper representations are just the tip of the iceberg with respect to working details underlie them.

While planning document is not very useful as an end item, the act of planning, is extremely important to project success. It provides a framework and forcing functions for making decisions, ensures buy-in on the part of the stakeholders and performers, and transforms subjective, generic process frameworks into objective process. A project's plan is a definition of how the project requirements will be transformed into a product within the business constraints. It must be realistic , it must be current, it must be a team product, it must be understood by the stakeholders, and it must be used.

Plans are not just for managers. The more open and visible the planning process and results, the more ownership there is among team members who need to execute it. Bad, closely held plans cause attrition. Good, open plans can shape cultures and encourage teamwork.

1.4 PLAN AND DEVELOPMENT TEAM

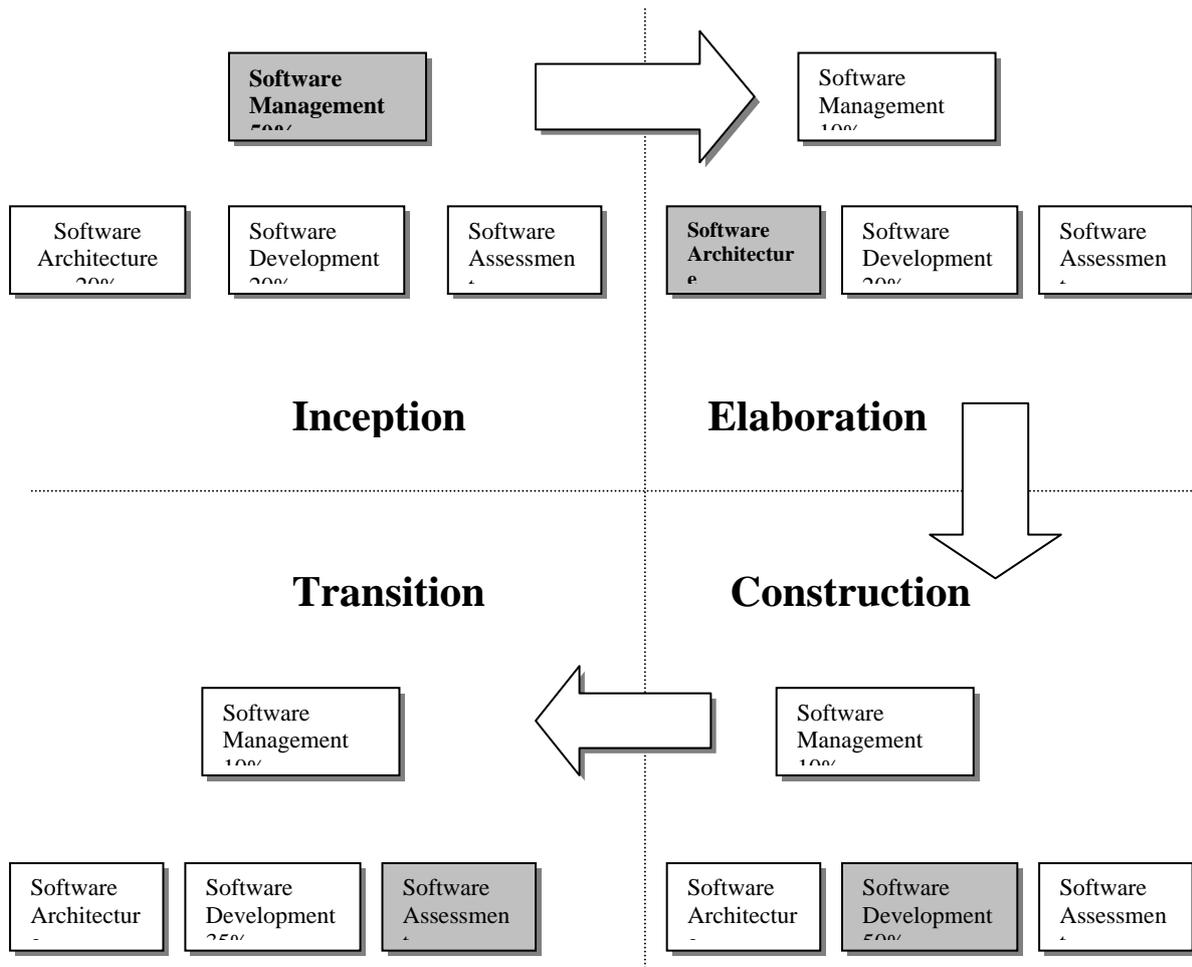
The Architecture of the team is recognized by the Project Organization and has to execute the plans designed with the team. A Different set of activities is emphasized in each phase, as follows:

Inception team: An organization focused on planning, with enough support from other teams to ensure that the plans represent a consensus of all perspectives.

Elaboration team: An architecture-focused organization in which the driving forces of the project reside in the software architecture team and are supported by the software development and software assessment team as necessary to achieve a stable architecture baseline.

Construction team: A fairly balanced organization in which most of the activity resides in the software development and software assessment teams.

Transition team: a customer-focused organization in which usage feedback drives the deployment activities.



1.5 THE WORK BREAKDOWN STRUCTURE

The most fundamental building block of a project plan is the work breakdown structure (WBS). The WBS contains a list of tasks to be accomplished, and you can implement a template WBS in a spreadsheet as a list of task names, task percentages of effort and task descriptions.

The Deliverable Plan

On most projects, you will be required to create technical documentation. Typically, this facilitates communication during development, assists in maintenance and provides milestones for payment.

The first step is to create another template, this time of document titles and descriptions. Templates of documentation are often referred to as standards; however, we can go beyond this initial step. It's valuable to include an estimate of the page count for each document in your project plan. This information helps set client expectations, defines the document scope for the developers who will write the documentation and acts as an estimate of review time.

I've found that the correct number of pages of documentation can be predicted using the staff months of development effort as the input. This is accomplished using the following formula:

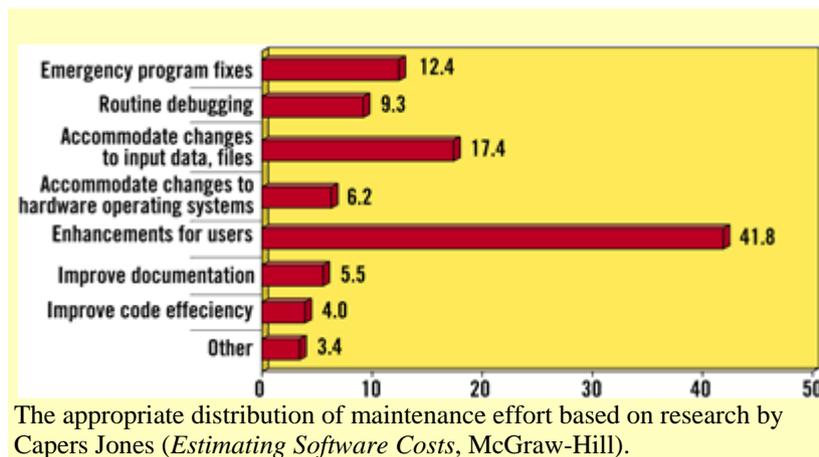
$$\text{Pages} = A * B (\text{Person Months}^C)$$

To apply this formula, you raise the person months of effort to a variable (C), multiply the results times another variable (B), then add a third variable (A).

Estimating Maintenance Effort

It's often necessary to estimate maintenance effort as part of your planning process. You may need to plan for an adequate maintenance staff, or you may need to provide a warranty on the software. Both tasks require you to estimate the effort required to provide these services.

First and foremost, you must ensure that you share a common understanding of maintenance work with your customer. At the CostXpert Group, we divide maintenance activities into three categories: corrective, adaptive and perfective. Standard bug fixes are considered corrective maintenance, while adaptive fixes involve modifications to the software required by such changes in the operating environment as database or operating system upgrades, compiler version changes and so on. Perfective maintenance is the most complex, in that it involves changing the code to allow the software to meet the same requirement but in a significantly more acceptable manner. The largest example of perfective maintenance is modifications of the code to improve performance in trouble spots. Note that not all changes to the code to add new functionality are considered maintenance, nor are any changes that are based strictly on user preference.



| A Simple Work Breakdown Structure Template | | |
|--|---------|---|
| Task | Percent | Description |
| Project Planning and Management | 10 | Project estimating, planning and risk management. |
| Define Business Strategy | 4 | Define the business strategy and document it in a creative fashion. |
| Define Marketing Strategy | 4 | Define the marketing strategy and document it a creative fashion. |
| Define Requirements | 4 | Define system requirements in both documentation and prototype form, plus define a production or style guide. |
| Design Site | 30 | Prepare functional specifications, technical specifications, site architecture and testplan. |
| Build Site | 19 | Build the site, including the front end, middle-tier, agents and interfaces. |
| Test Site | 19 | Test the site, both at the unit level and the integration level. |
| Deploy Site | 10 | Deploy the site, including acceptance testing and user documentation when required. |

| Table . A Sample Deliverable Planning Standard | | | | |
|--|------|------|------|---|
| Deliverable | A | B | C | Description |
| Software Development Document Plan (SDP) | 5.00 | 0.08 | 0.91 | Describes a developer's plans for conducting a software development effort. This is expressed in the form of a project Gantt chart, supplemented by one or more Project Work Orders covering the various phases of work. |
| Strategic Brief | 2.00 | 0.90 | 0.91 | Documents strategic business objectives, measures of business success, stakeholders and stakeholder objectives. |
| Creative Brief | 2.00 | 0.90 | 0.91 | Defines target audience(s), objectives, market and product positioning, branding and relationship between Web strategy and overall market strategy. |
| Software Requirements Specification (SRS) | 3.00 | 1.21 | 0.91 | Specifies the requirements for a computer program and the methods to be used to ensure that each requirement has been met. |
| Prototype | 1.00 | 0.00 | 1.00 | System prototype. |
| Functional Specifications | 0.00 | 4.00 | 0.91 | Defines back-end and middle-tier functional requirements from a business perspective. |
| Technical Brief | 0.00 | 4.00 | 0.91 | Defines the technical details of the system design, including object design. |
| Site architecture | 0.00 | 2.14 | 0.91 | Describes the site architecture with sufficient detail to allow the site to be configured for deployment. |
| Software Test Plan (STP) | 5.00 | 0.12 | 0.91 | Describes plans for acceptance testing of a computer program. It describes the software test environment, identifies the tests to be performed and provides schedules for test activities. |
| Software Design Description (SDD) | 0.00 | 8.00 | 0.91 | Describes the program-wide design decisions, the software architectural design and the detailed design needed to implement the software. It includes the middle-tier object and application server design, pseudocode for all key algorithms and business rules, field mappings for all screens, sort, select and display requirements for reports, and detailed data format specifications for all interfaces. |
| Software Test Description (STD) | 5.00 | 2.27 | 0.91 | Describes the test preparations, test cases and test procedures to be used to perform acceptance testing of a computer program. |
| Software User Manual (SUM) | 15.0 | 2.10 | 0.91 | Instructs a hands-on software user how to install and use a computer program. It includes a tutorial describing how to accomplish specific user work tasks identified as Operational Scenarios in the Software Requirements, Specification, and it also offers a comprehensive reference to all screens, reports and menu choices. |

1.6 PROJECT SCHEDULING

A schedule is a planning document, but it can be used effectively only if it is rich enough in useful and accurate information. A schedule lacking sufficient information is no more than an obnoxious administrative hurdle that does not contribute to the successful completion of the project. The process of creating a schedule begins with the identification of tasks that must be completed in order to finish the project. This step is primarily the responsibility of the Director since it is intimately related to the architecture of the system as conceived by the Director.

The basic unit of scheduling is the task. Coding tasks involve development of a module and documentation tasks involve development or editing of a chapter or section. To schedule a task requires specification of at least three items: who is responsible for the task, when should work on the task begin, and when is the completed task due. In many cases, these items require simple, arbitrary decisions and may be decided by the Producer alone. Sometimes there is a "best" decision, if not a "right" one, and the Producer must exercise good judgment. For example, it might make sense to assign all of the modules for a particular small subsystem to the same person. It usually makes sense for the documentation for a module to be written by the author of the code, though some effective arguments can be made for avoiding this strategy.

Sometimes there are more complex constraints on how and when modules are developed that require cooperation in scheduling between the Producer and Director. The Director understands relationships between modules that require sequencing of activities. For example, "task A must be completed before task B begins" or "task A may begin before task B ends" and so on. Such sequential dependencies can be used by the Producer to determine the minimum time required to complete the project. The Producer and Director must work together to develop a schedule that satisfies the fixed constraint of the time available for the project.

Several specific goals should be met when creating a schedule. First, a good schedule will create a reasonable (high) degree of parallelism at all stages of the project. Parallelism is one of the keys to obtaining concentrated effort from the project team. At any point in the project, each team member should be assigned several tasks. For example, a team member might be assigned to document the module just completed and to write code for two new modules. This kind of parallel assignment allows a team member to be productive at all times, even if one of the tasks hits a snag. It also enhances morale since no one feels stuck with a particular scut job, and each individual has some control over how to use working time. Many decisions are made by the boss or the Producer, but decisions that need not be made by them to order the affairs of the team should be left to the discretion of the individual team members. This usually includes where and when team members work on their tasks. Another motivation for parallelism comes from the intertwined nature of your fates. There are no winners on losing teams. Therefore, it is useless to shout, as the waves lap up the mast, that the leak was in somebody else's part of the boat. By maintaining parallelism the Producer ensures that everyone is contributing to the project goals and there is no wasted or idle time or effort.

A good, complete schedule will allow the Producer to identify the longest sequential chain of tasks in the project. This longest chain is the "critical path" that determines how long the project is actually going to take. Any slippage in the tasks on the critical path will directly affect the delivery date of the product. Thus, identifying the critical path allows the team to pay close attention to the tasks on that path. This is not to say that tasks not on the critical path are unimportant or can afford to ignore deadlines; indeed, slippage in a task not on the critical path can cause a rippling of schedule changes through the project, resulting in a change in the critical path. There exist computer tools such as MacProject that can help to define a schedule, compute the critical path, keep up with allowable slack times, and maintain task lists for team members. Such tools are available for use in this course, but will not be discussed further in this text.

The third goal in scheduling is to include in the milestones a presentation of a prototype to the client as early as possible. A prototype is an early version of the system that demonstrates the architecture of the system Ñ mainly its user interface and control strategies Ñ without all of its substance, such as error checking, speed, much of its functionality, its real production platform, and so on. It is entirely appropriate to "fake up" a user interaction sequence using MacDraw or some other tool and show the "storyboard" session to the client before any code is written. As code is developed, it is appropriate to create a prototype in which one particular typical interaction works, in a jury-rigged fashion, and no more. Prototyping is valuable to obtain feedback on the architectural design from the client, to build morale in the team, and to prioritize further development of the product.

The fourth goal for a schedule is to permit team members to obtain a global view of where the effort is heading and how. The schedule is maintained by the Producer, but it should be available to the whole team. Engineers do not work just for a paycheck. An engineer's motivation is to see his product used; to see that what he is doing contributes to the whole. The public schedule can provide this motivation concretely.

The fifth goal of a schedule is to document progress in the project. Marking milestones as achieved and tasks as completed provides a sense of "closure" that is great for morale and fulfills psychological needs to sense progress in a long project. Some project teams have taken great pride in including a version of their master schedule in their final report with the dates of accomplishment of milestones or task completions noted. Also, some projects have left in the final report the schedule items that were abandoned to show how the project's goals evolved. One group's Producer prepared this kind of display biweekly and distributed it to the team, client and Boss.

Initial and Final Project Schedule plans

A Project plan is to be scheduled at initial level and later it has to be monitored. Finally again plan has to be delivered for checking the progress.

| Document | Responsible | Phase |
|---|--|-------------------|
| Problem definition | Customer | Definition Phase |
| Requirements specification (with test cases for each requirement) | Developer in cooperation with customer | Contract Phase |
| Project plan | Developer | Contract Phase |
| Design specification | Developer | Development Phase |
| Test specification | Developer | Development Phase |
| Acceptance document | Developer | Development Phase |
| Project evaluation | All | - |

Requirements on documents

| | |
|---|---|
| Requirements on requirement specification | Requirements must be numbered. Requirements must be prioritized. The document shall be understandable, i.e. contain all necessary explanations and background information. |
| Requirements on project plan | It must contain a plan regarding calendar time. It must contain a plan regarding man-hours. A description of the different tasks/actions/phases (e.g. what shall be done in the "beta test phase"). A description of who in the group does what (who is project leader, in charge of testing, etc.). |
| Requirements on design specification | It must be understandable, i.e. one should be able to implement the system from it. It must contain an architectural description. It must contain a class diagram. |
| Requirements on test specification | For every test case: Name of the test case Which testdata that should be used Expected result Outcome of the test case |
| Requirements on acceptance test specification | It should include: The test cases from the requirement specification The test cases from the test specification Any test cases given by the customer |
| Requirements on project evaluation | Report on used resources What in the project (not the course) has gone as planned What in the project hasn't gone as planned Other experiences from the project |

1.7 ACTIVITIES OF PLANNING

Top level activities:

1. The software engineering group participates on the project proposal team.
2. Software project planning is initiated in the early stages of, and in parallel with, the overall project planning.
3. The software engineering group participates with other affected groups in the overall project planning throughout the project's life.
4. Software project commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.
5. A software life cycle with predefined stages of manageable size is identified or defined.

6. The project's software development plan is developed according to a documented procedure.
7. The plan for the software project is documented.
8. Software work products that are needed to establish and maintain control of the software project are identified.
9. Estimates for the size of the software work products (or changes to the size of software work products) are derived according to a documented procedure.
10. Estimates for the software project's effort and costs are derived according to a documented procedure.
11. Estimates for the project's critical computer resources are derived according to a documented procedure.
12. The project's software schedule is derived according to a documented procedure.
13. The software risks associated with the cost, resource, schedule, and technical aspects of the project are identified, assessed, and documented.
14. Plans for the project's software engineering facilities and support tools are prepared.

Types of Activity Relationship

1. A documented software development plan is used for tracking the software activities and communicating status.
2. The project's software development plan is revised according to a documented procedure.
3. Software project commitments and changes to commitments made to individuals and groups external to the organization are reviewed with senior management according to a documented procedure.
4. Approved changes to commitments that affect the software project are communicated to the members of the software engineering group and other software-related groups.
5. The size of the software work products (or size of the changes to the software work products) are tracked, and corrective actions are taken as necessary.
6. The project's software effort and costs are tracked, and corrective actions are taken as necessary.
7. The project's critical computer resources are tracked, and corrective actions are taken as necessary.
8. The project's software schedule is tracked, and corrective actions are taken as necessary.
9. Software engineering technical activities are tracked, and corrective actions are taken as necessary.
10. The software risks associated with cost, resource, schedule, and technical aspects of the project are tracked.
11. Actual measurement data and replanning data for the software project are recorded.
12. The software engineering group conducts periodic internal reviews to track technical progress, plans, performance, and issues against the software development plan.

Formal reviews to address the accomplishments and results of the software project are conducted at selected project milestones according to a documented procedure

Software planning data are recorded

Affected groups and individuals agree to their commitments related to the software project

Ability to make high-level organizational changes in response to general threats and pressures.

A bottom-up approach works best when success is dependent upon sensitivity to customer needs and demands.

1.8 CONCLUSION

The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. Project management includes developing a project plan, which includes defining and confirming the project goals and objectives, identifying tasks and how goals will be achieved, quantifying the resources needed, and determining budgets and timelines for completion. It also includes managing the implementation of the project plan, along with operating regular 'controls' to ensure that there is accurate and objective information on 'performance' relative to the plan, and the mechanisms to implement recovery actions where necessary.

1.9 REFERENCES

- [1] David I. Cleland, Roland Gareis (2006). Global project management handbook. McGraw-Hill Professional, 2006. ISBN 0071460454. Pp.1-4.
- [2] Martin Stevens (2002). Project Management Pathways. Association for Project Management. APM Publishing Limited, 2002.
- [3] Morgen Witzel (2003). Fifty key figures in management. Routledge, 2003. ISBN 0415369770. Pp. 96-101.
- [4] Bjarne Kousholt (2007). Project Management –. Theory and practice.. Nyt Teknisk Forlag. ISBN 8757126038. p.59.
- [5] F. L. Harrison, Dennis Lock (2004). Advanced project management: a structured approach. Gower Publishing, Ltd., 2004. ISBN 0566078228. p.34.
- [6] Stellman, Andrew; Greene, Jennifer (2005). Applied Software Project Management. O'Reilly Media. ISBN 978-0-596-00948-9.
- [7] Albert Hamilton (2004). Handbook of Project Management Procedures. TTL Publishing, Ltd. ISBN 07277-3258-7.
- [8] Roger S. Pressman, Software engineering : A practitioner's approach, MCGraw-Hill publication, 1997, ed.4
- [9] Edward Kit, Software testing in the real world, Addison-Wesley publications, 2000, ed.1
- [10] Ian Sommerville, Software engineering, Addison-Wesley publications, 1996, ed.5
- [11] Pankaj Jalote, An integrated approach to software engineering, Narosa publications, 1997, ed. 2
- [12] Shari Lawrence Peleeger, software engineering theory and practice, Pearson education, 2001, ed. 2
- [13] Richard Fairly, Software engineering concepts, McGraw-Hill Inc., 1985
- [14] Myers, Glenford J. (1979). *The Art of Software Testing*. John Wiley and Sons. p. 145-146. ISBN 0-471-04328-1.
- [15] Barry W. Boehm, Software Engineering Economics, Prentice-Hall Inc., 1981.
- [16] Dustin, Elfriede (2002). *Effective Software Testing*. Addison Wesley. p. 3. ISBN 0-20179-429-2.
- [17] Carlo Ghezzi et al : Fundamentals of Software Engineering (Fifth edition) PHI 1991.