**SOFTWARE PROJECT PLANNING**

**Introduction**

Due to an increasing **market demand** and a rapid **expansion of technological possibilities**, the pressure on the software industry is constantly growing.

In order for the *software companies to cope with frequently shifting conditions* and the *raising quality demands*, they need to constantly ***improve the quality*** of their work.

These improvement efforts must be reflected in the planning activities.

Project planning therefore has become of great importance for software development, which has forced the industry to make fundamental changes of the planning process.

**However**, although ***planning does not guarantee success***, at least a minimum degree of planning is required, since lack of planning most likely will guarantee failure (***lack of planning is planning to******fail***).

**The Project Plan**

A project **plan** should be **based on the project requirements** and **developed estimates.**

Itis a formal, consistent and approved document that provides for the essential project guidelines.

Generally, the **purpose** of the project plan document is to **support management** and **control of the project execution**.

The plan should cover **all phases of the project** and should ensure that all involved plans are consistent with each other.

A project plan usually contains several parts produced in order to help the project team with their project.

**Objectives of the project plan are:**

*-Guide project execution*

*- Document project planning assumption*

*- Document project planning decisions regarding alternatives chosen*

*- Facilitate communication among stakeholders*

*- Define key management reviews as to content, extent, and timing*

*- Provide a baseline for progress measurement and project control*

**The project plan is generally developed in the initial phase of the project and needs to be reviewed and agreed upon by all concerned persons.**

However, the plan is expected to change over time and is updated each time the actual progress differs from the plan or when project conditions changes, which require new approaches

A carefully prepared project plan if properly followed and committed to, should lead to a successful project and eliminate many of the pitfalls inherent in the project management process

It provides leadership vision and facilitates for management to utilize available resources efficiently

**Types of project plans**

In addition to the main software project plan, different types of specific plans may be developed to support the main plan in different areas. Example of such plans may be:

**- Quality plan** - Includes the quality procedures and standards that concern the project.

**- Validation plan** - Covers approaches, resources and schedule involved in the system

validation.

**- Configuration management plan** – Consists of the configuration management procedures and

structures to support the project.

**- Maintenance plan** - Predicts the maintenance requirements, maintenance costs and the effort

required.

**- Staff development plan** - Includes how available skills and experience will be developed.

**The form of a project plan**

The main project plan may be of various forms. The list below is a summary of What to put emphasis on:

* Scope and Objectives
* Work Breakdown Structure (WBS)
* Budget & Schedule
* Risks
* Monitoring and Reporting Mechanisms
* Resources
* Knowledge & Skills
* Stakeholder Involvement

**1. Scope and Objectives** - The scope and objectives of the project are generally set by extracted requirements. The scope is a statement defining the project and its deliverables and should clearly and concisely state project information such as, what it is, what it does, how much it will cost, and when it will be delivered.

The project scope has strong relationships to the project schedule and involved resources. Thus, modifications of the project may also affect the project scope.

**2. Work Breakdown Structure (WBS)** - A WBS is a **deliverable-oriented** grouping of project components that organizes and defines the total project scope. Thus, the WBS divides the total scope into major work packages, which are further subdivided into manageable work items to be accomplished in order to finish the project.

**3. Budget and Schedule** - The budget and schedule of the project are based on established estimates.

Schedule development implies to determine **start and finish dates** for concerned project activities, which for example may be performed through simulations or mathematical analysis.

The schedule may be presented by **Gantt charts, milestone charts**, etc. and may be supplemented by supporting detail documents that include identified assumptions and constraints

**Cost budgeting** implies to allocate the overall cost estimates to individual work items.

The budget should be based on and supported by the WBS, project schedule, and the cost estimates.

**4. Risks** - Planning for project risks should address issues that could jeopardize accomplishment of critical project objectives.

-The planning process involves identifying project risks, quantifying the risks, and developing risk responses. These must be documented in a **risk register**.

The risk planning process will be discussed later.

**5. Monitoring and Reporting Mechanisms** – The purpose of monitoring mechanisms are to provide an **understanding of the project progress** in order to **take appropriate corrective actions** if project performance deviates from established plans.

Monitoring involves monitoring actual values of planning parameters, such as **cost**, **effort**, and **schedule**. These values are compared to the estimates and possible deviations are identified.

Reporting mechanisms are concerned with collecting and disseminating performance information in order to provide involved stakeholders with **status, progress and information** about how resources are used to fulfill established objectives.

**6. Resources** - The resources and quantities required in order to carry out the project should be identified and described.

Project resources come in various forms such as, personnel, funds, equipment, facilities, material, information etc. and the selection of these resources should be based on the established estimates.

Establishing resource requirements *allow for several benefits such as, identification of resource shortage, identification of feasibility problems due to resource conflicts,* etc.

**7. Knowledge & Skills** - Planning for knowledge and skills involves both training of project team and acquisition of knowledge from external sources.

The knowledge and skills required to execute the project should be identified and the currently available knowledge and skills should be assessed.

With this information available, the deficiency of knowledge and skills is identified and mechanisms for providing this knowledge and skills are selected.

**8. Stakeholder Involvement** - Stakeholders involved in the project should be identified and their functions requiring representation in the project should be defined.

Furthermore, the level of interaction and the relevance of each involvement should be described. An appropriate technique to handle this effort is to develop a **two-dimensional matrix** with stakeholders along one axis and project activities along the other axis.

A project plan may take various forms depending on the needs and purpose. The

project plan is obviously developed when the developer has reached an agreement

with the customer.

Prior to such an agreement, the supplier must be certain that they are capable of undertaking a project.

If the supplier doesn't possess the knowledge, skills or time to undertake the project, it's a waste of time to initiate negotiations with the customer.

To find out whether this is the case or not, **effort and time estimates** must be established. When this is settled and negotiations are initiated, the supplier must come up with a reasonable price for the product.

The price must be compatible, cover the development costs and furthermore generate a desirable profit.

In order to achieve this balance, the **project costs must be estimated**. With all the estimates at hand, it’s possible to establish the project scope, a schedule and budget, resources required, and knowledge and skills required.

**Project estimation**

Project estimation implies to predict the effort required to successfully execute the project.

Lack of project estimates makes the project boundaries quite vague.

Estimates serve as a compass, navigating the project team throughout the project lifecycle.

Making estimations are not difficult, but to establish accurate and realistic estimates is one of the most challenging and important activity in software development.

**Why do we estimate?**

There are several reasons to establish project estimates.

Project estimates, generally, provide the project staff, management and other stakeholders with basic guidelines and settles the project scope.

Estimates are also an important instrument to verify that the supplier is capable of undertaking the project and moreover provides for a price indication.

Thus, accurate estimates are critical to both the software supplier and their customers.

**Underestimating the costs may lead to that the supplier may be forced to exceed the budget, abandon the time plan and decrease functionality and quality of the software.**

However, overestimating may result in too many resources allocated to the project or that the

supplier loses the contract in the bidding phase, because of a too high price.

**Importance of project Estimates:**

*- Estimates can help to classify and prioritize development projects with respect to an overall*

*business plan.*

*- Estimates facilitate to allocate resources to the project and how these resources will be used.*

*- Estimates make it easier to manage and control projects when resources are linked to real*

*needs.*

**What do we estimate?**

A project is based on specific requirements and preconditions that set the project scope.

Relevant project characteristics to estimate are factors that somehow limit the project and strongly influence the project and the outcome.

* *How much effort is required to complete the project?*
* *How much calendar time is needed to complete the project?*
* *What is the total cost of the project?*

In order to answer these questions, it is essential to first estimate the **size of** the project.

The overall size of the project is provided by estimating each work product generated by the project.

**Size – Deliverable and non-deliverable work products**

Everything produced and generated within the project are classified as work products.

Thus, work products may be **deliverables** as well as **non-deliverables** and may be both documents and software.

The size and complexity of these work products are required as inputs to further estimate effort, schedule and cost.

**The size may be such as number of functions, function points, lines of code, number of pages, number of classes, volume of data etc.**

Size estimates are the most important software estimates.

The main reason for that is the inherent dependence of cost and quality on the size.

Development of large software systems obviously costs more and also involves more quality

problems than smaller software systems.

Thus, Size estimation provides the basis for all further software estimation.

**As stated above, size may be estimated by predicting the amount of different software metrics, such as number of functions, function points, lines of code etc.**

Line of Code (**LOC**) is quite popular **software metric** to use in order to estimate the software size.

However, since **LOC** is strongly dependent on the programming language and the programming style, this approach may involve uncertainties.

Different languages require different amount of code to implement a specific function and different code standards and programming styles expect a certain amount of code on one line.

More reliable software metric, when it comes to size estimation, is function points, which measures the amount of software functionality that will be implemented in the software.

In order to estimate the number of function points, the occurrences of five system types are counted

**Effort – Predict the total project effort**

With the size estimates at hand it is possible to estimate the project efforts.

Estimating effort implies to predict the total project effort. How much time is required to produce the already estimated work products?

In order to establish these estimates it can be wise to define a **project life cycle** and a **Work Breakdown Structure** (WBS). A WBS is a product-oriented structure that divides the overall project into an interconnected set of small and manageable components.

The WBS makes it possible to identify and organize these components and facilitates to allocate efforts and responsibilities to each component.

A project life cycle defines how the different project phases are sequenced. The software community provides for several different **lifecycle models**, such as the waterfall model, XP, evolutionary model etc. Other relevant factors to consider when making effort estimates are knowledge, skill, and training needs.

**Time – Develop a project schedule**

When the effort estimates are available it’s possible to determine and develop a project schedule.

The project schedule should provide for the **start and finish time** of defined tasks, the number of people involved in each task, and what their assignments/roles will be.

The schedule should be translated into calendar time in order to make it more understandable and useful.

**Cost – Estimate the complete project**

In order to make a cost estimate of the complete project, several factors must be considered, such as labor, hardware and software resources, training, and other supporting infrastructure needs.

The effort costs, which are the dominant cost drivers in most projects, may for example be salaries, social- and insurance costs etc. of involved staff.

However, effort costs must also take overheads into account. Overheads may be such as, costs of building, heating, lighting etc.

**How do we estimate?**

It can be quite easy to come up with estimates, but to establish credible and accurate estimates requires experience and/or knowledge.

The software community utilizes a number of different models and techniques to develop and establish estimates.

Over time, several **estimation techniques** have been proposed, such as estimation by analogy, expert judgement etc.

However, there is no simple way to establish accurate estimates. Despite the extensive amount of experience with estimation techniques, the accuracy of the existing estimation techniques is not

adequate.

Initial estimates are usually based on inadequate information in a user requirements definition. Moreover, the software to be developed may run on unknown computers or use new technologies, and the people in the project may also be unknown.

Estimation techniques are divided into two different categories:

* *experience-based techniques and*
* *algorithmic* modeling.

**Experience-based techniques**: are based on the project manager’s knowledge and experience, and may take previous projects into consideration.

**Algorithmic models (also known as parametric models):** are based on mathematical algorithms that use variables considered to be major cost factors, such as the project size, resources, and other process and product factors.

**Experience-based techniques**

All experienced-based techniques rely on **experience-based judgements** by competent and experienced personnel.

However, there may be ***significant dissimilarities between past and future projects***.

Many new development technologies and methodologies have been introduced in the last 10 years and if the developers haven’t worked with these new concepts, their knowledge and experience may not be of any value in order to develop credible estimates.

**Most common experience-based techniques**:

These techniques can be applied using either a **top-down approach** or a **bottom-up approach**.

**A top-down approach:** begins at system level by examining the overall functionality.

The system-level activities such as integration, configuration management and documentation are considered.

**The bottom-up approach:** begins at the component level and decomposes the system into components. Thus, each component is estimated and together, these estimates add up to an estimate of the whole system.

**1. Estimation by Analogy**

This experience-based estimation technique *compares the current project* with one or more similar projects that are completed.

Thus, the characteristics, such as *effort, schedule and cost of the projects* are known.

Having done similar projects in the past and knowing its characteristics, estimates are established **through analogy reasoning**.

The strength of this technique is that it is based on actual experiences and real project data. Thus, current method only deals with issues that actually occur in practice whereas algorithmic methods must cope with all possible problems.

The fact that this method deals with *experienced and completed projects* it is also capable of dealing with poorly understood domains. It avoids the problems associated both with knowledge elicitation and extracting and codifying the knowledge. The weakness is obviously that it’s not always certain that there exists a similar project that is representative of the constraints, environment, functions etc. of the new software.

**2. Expert judgement**

Current method is the most commonly used method.

- It bases the estimates on knowledge and experience of one or several experts.

- Each expert establishes an estimate and the estimates are compared and discussed.

- This process is iterated until the experts have agreed upon an estimate.

**So, who is an expert?** People with knowledge in software development? People with experience within the estimation area?

***An expert is defined as a person that is familiar and has experience of software development in the current domain.***

*The strength with current approach is that an expert with adequate experience may provide good estimates quite fast and relatively cheap*.

**Note:** However, on the other hand, if the expert(s) doesn't possess sufficient experience and knowledge, the estimates will most likely suffer from inaccuracy.

**3. Parkinson’s Law**

This experience-based estimation technique is based on **Parkinson's law**, *"work expands to fill available time"* and implies that the project cost is determined, not estimated, by the available resources rather than on objective assessments.

For example, if the software to be developed should be delivered in 12 months and five people are currently available, the effort is determined to be 60 (12\*5) person month.

**Note:** This approach may result in large overruns.

**4. Pricing to win**

This approach estimates the cost with the intention to set a good price.

A ***good price implies a price that will win the project***. Thus, the estimates are based on the

customer’s budget rather than on their needs for certain functionality.

***The strength of this approach is obviously that the supplier often gets the contract****.*

However, the probability that the customer gets the software they require is quite

small, since the costs don’t reflect the work required.

**Algorithmic modeling**

Algorithmic estimation is based on the **application of a cost model**, i.e. mathematical formulas that have been derived through statistical analysis of completed projects.

In its most general form, the software cost may be expressed as:

**Effort = A × SIZE B × M**

in which A and B represent constant factors that depends on organizational practices and the type of software that is developed.

The **SIZE** represents the software size and may be expressed in either **code size** or **function points**.

M is an effort multiplier which is determined by combining several cost factors.

The different Cost factors are divided into four different categories,

***- Product factors****: reliability, complexity, size of database, reusability etc.*

***- Computer factors:*** *Execution time constraints, storage constraints, platform volatility etc.*

***- Personnel factors:*** *programming capability, analyst capability, application experience etc.*

***- Project factors:*** *use of software tool, required development schedule etc.*

These cost factors are applied in the **COCOMO** model, which is an example of an algorithmic estimation model.

**COCOMO model**

The **COCOMO model** is an empirical model that was developed by collecting data from a large number of different projects. The collected data was analyzed in order to find out formulas linking the software **size** and **project factors** to the **development effort**.

The **choice of estimation technique** obviously depends on various circumstances and resources, such as knowledge, experience, data access etc. However, project estimation should always be based on several methods in order to compare the result.

If the different approaches don’t generate approximately the same result, the information available is not sufficient and consequently, actions should be taken in order to retrieve more useful information.

**Risk Planning**

Estimations are a large part of project planning and naturally also a part of the project plan.

However, along with estimations come **risks**. Therefore, it is natural for us to **present risks after estimations**.