CHAPTER 3

SOFTWARE PROJECT PLANNING AND RISK

S/w project planning is a collective set of activities to start s/w project management

Software Project Planning encompasses five major activities:

1. Estimation
2. Scheduling
3. Risk analysis
4. Quality management planning
5. Change management planning

OBJECTIVE:

Objective of project planning is to provide a framework that enables the manager to estimate resources, cost and schedule. Although few uncertainty, project is developed on a plan that has been established after planning tasks. So plan must be updated and followed with progress within the project.

Steps of planning:

1. Establish project scope
2. Determine feasibility
3. Analyze risk
4. Define required resources
5. Estimate cost and effort

6. Develop project schedule.

SOFTWARE SCOPE:

Scope describes the function and features that are to be delivered to end users, the input and the output data, outcome of the software used by the users and the performance, constrains, interfaces and reliability.

Scope can be determined using:

1. A set of use-cases developed by users.
2. A narrative description of s/w scope is developed after communication with all stake holders.

RESOURCES:

Reusable s/w:

1. Off the shelf components/part component:

It may have already been developed or can even be asked the third party to develop it.

2. Full experience components:

Members of current team have had full experience in the application area represents by these components. So, modification will be less risky.

3. Part-experience components:

Needs large changes compared previous project components.

Members have limited experience.

More degree of risk is observed.

4. New component:

Needs totally new components

SOFTWARE PROJECT ESTIMATION:

Estimation is the process of determining the cost, effort, resource, and the time required to develop a specific s/w based system. Estimation is done early actually. Estimation is form of problem solver.

To achieve reliable cost and effort estimates a number of option arise like:

1. Delay estimate until late in the project.
2. Base estimates on similar projects that have already been completed.
3. Use relatively simple decomposition technique to generate project cost and effort estimated
4. Use one or more empirical models for s/w cost and effort estimation.

DECOMPOSITION TECHNIQUES:

The problem to be solved is generally too complex if considered as a problem. Hence we decompose the problem into smaller set of smaller problems.

Decomposition is done in two major areas

1. Decomposition of problem
2. Decomposition of process

First of all size is to be determined before estimation. Size refers to quantifiable outcomes of the software projects. Direct approach says LOC gives size. Indirect approach says FP gives size.

Different factors affect estimations like:

1. Degree of size of the product to be built
2. Convert size requirement from size of product as human effort, calendar time.
3. Degree of reflection of abilities of the s/w team with project plan.
4. The stability of product requirements and the environment the support.

Problem based estimation contains LOC based estimation and FP based estimation. Process based estimation is decomposition of process into smaller set of tasks. And use case based estimation done using use-case diagram developed by end user.

EMPERICAL ESTIMATION MODELS:

It is a typical model that is derived from past projects using regression (relation between the mean values) on data collected. Empirically derived formulas are to predict effort as a function of LOC and FP. The result of LOC based or FP based estimations are plugged into the estimation model. Empirical data that supports most of the models are derived from limited example projects. So no model is suitable for all types of projects. Local desires and conditions may effects the models.

Structure:

General form is: $E=A+B*(e_v)^c$

Where A, B and C are empirically derived constants.

E is effort in person-month and e_v is the estimation variable either LOC or FP.

In addition to above relationship adjustment/modification are done on the basic of other characteristics like problem complexity, staff skills, environment etc.

Among many LOC oriented estimation models are:

1. Walton-Felix model: $E=5.2*(KLOC)^{0.91}$
2. Bailey-Basili model: $E=5.5+0.73*(KLOC)^{1.16}$
3. Boehm simple model: $E=5.288*(KLOC)^{1.047}$

- FP oriented estimation models some are

1. Albrecht and Gaffney model : $E=-9.14+0.355FP$
2. Kemerer model : $E=-37+0.97FP$

THE COCOMO II MODEL:

Introduced by Barry Boehm COCOMO II was more comprehensive and addressed following areas:

1. Application of composite model
2. Used at early stages
3. For consideration of s/w/system interaction
4. Evaluation of technology maturity

Sizing information is also required for COCOMO II and three sizing factors are FP, LOC and object points. The COCOMO II uses object points as indirect measures that is computed using counts of number of

1. Screen at UI
2. Reports

3. Components likely to be required to build the application.

- Each object is categorized into three complexities levels -> simple, medium and difficult criteria suggested by Boehm. The complexity is determined using table, the object point count is determined by multiplying the original number of object instances by weighting factor and summing to obtain a total object point count. When reuse is to be done the reuse percent is to be estimated and the object point count is adjusted. NOP=object point*[(100-%reuse)/100] where NOP is defined as a new object points.

  Productivity rate is used to estimate effort based on the computed NOP value

  PROD=NOP/effort

Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
Components			10

THE SOFTWARE EQUATION MODEL:

It is multivariable model that assumes a specific distribution of effort over the life of a s/w development project. 4000 s/w projects were used to collect data from which estimation model has been formed.

$E=[LOC*B^{0.33}/P]^3 *(1/t^4)$

Where E= effort, t= project duration,B=special skill factor, p=productivity

Programs for KLOC=5 to 15->B=1.6

KLOC > 70->B=3.9

Typically, P=2000 for real time embedded system

P=10000 for telecommunication and system software

P=28000 for business system

P=12000 for scientific software

Software equation has two independent parameters i.e estimate of size and indication of project duration.

A set of equation derived from software equation from which minimum development time is defined as, $t_{min} = 8.14(LOC/P)^{0.43}$ in months and E= 180 Bt^3 in P-M for E>=20pm

RISK:

"Risk": Possibility of loss or injury - Webster

Risk Exposure =(Probability of unsatisfactory outcome) X (Loss if unsatisfactory outcome)

Quotes:

"In architecting a new software program, all the serious mistakes are made on the first day."- Robert Spinrad, VP-Xerox, 1988.

"IF YOU DON'T ACTIVELY ATTACK THE RISKS,

THE RISK WILL ATTACK YOU" - -Tom Gilb

RISK ANALYSIS AND MANAGEMENT:

Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty. Risk is a potential problem- it might happen, it might not. But it is really good idea to identify, assess its probability of occurrence, estimate its impact, and establish a contingency plan in case the problem really occurs.

Everyone involved in the software process – managers, software engineers, and customers participate in risk analysis and management. Understanding the risk and taking proactive measures to avoid or manage them is a key element of good software project management.

RISK MANAGEMENT STRTEGIES:

Risk involves changes in mind, opinion, actions or place. Risk converts future happenings.

Steps

- ✓ Risk Identification
- ✓ Risk analysis
- ✓ Risk ranking
- ✓ Risk management

TYPES:

1. Reactive Risk :
   Reactive risk strategies have been laughingly called the “Indiana Jones School of risk management”. In the movies that carried his name, Indiana Jones, when faced with overwhelming difficulty, would say, “Don’t worry, I will think of something!” Never worrying about problems until they happened, Indy would react in some way.
   Sadly, average software project manager is not Indiana Jones and the members of the team are not too trustworthy. Yet the majority of software teams rely solely on reactive risk strategies.
2. Proactive Risk :

   Considerably this is more intelligent strategy for risk management. This strategy begins long before technical work is initiated. Potential risks are identified, their probability and impact are assessed and they are ranked by importance.

   Software team establishes a plan to manage risk. The primary goal is to avoid risk, but not all risk can be avoided, so team works to develop a

contingency plan that will enable it to respond in a controlled and effective manner. Proactive risk strategies consist of:

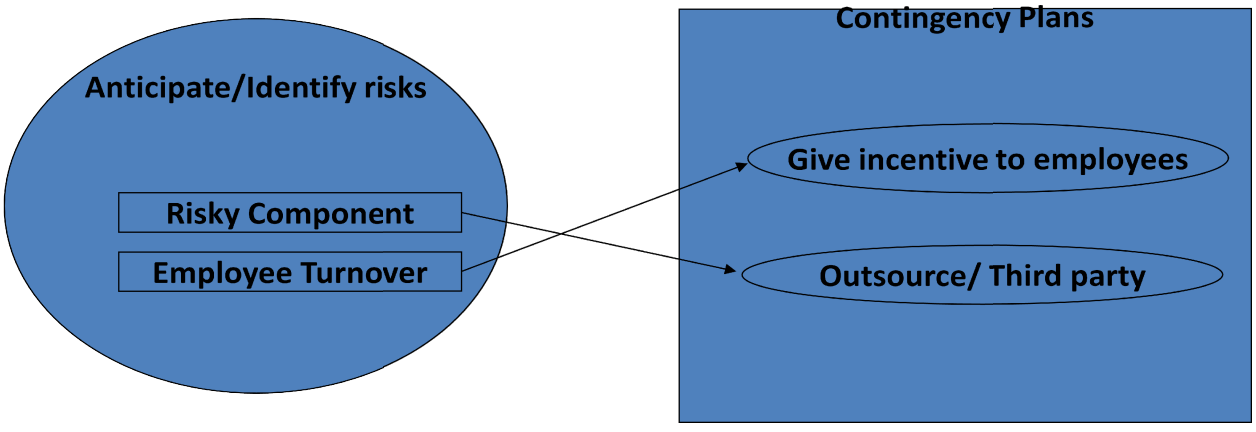
- Risk Avoidance
- Risk Transfers
- Risk Reduction
- Risk Acceptance

IMPORTANCE OF RISK MANAGEMENT:

1. Addresses Complex Software Systems
2. Focuses Projects on Critical Risk Items
3. Provides Techniques for Handling Risk Items
4. Reduces Software Costs by Reducing Rework
   - Usually 40-50% of software costs

SOFTWARE RISKS:

Anticipated unfavorable event during software development is called software risk. When risk turns to reality it hampers successful and timely completion of project. When risks are analyzed it is a must to confirm the level of uncertainty and degree of loss associated with each risk.



CHARACTERISTICS OF SOFTWARE RISK:

1. *Uncertainty*: Risk may or may not happen.
2. *Loss*: If risk becomes a reality *losses* will occur

TYPES OF RISKS:

1. PROJECT RISK:
    a. Affect the project schedule or resources or cost.
    b. Budgetary
    c. Schedule
    d. Personnel
    e. Resource
    f. Customer related

    SOLUTION:

    Increase visibility through documentation.

2. TECHNICAL RISK:
    - Ambiguous, incomplete, changing specification
    - Potential Flaws in Design
    - Implementation
    - Interfacing
    - Testing, maintenance
    - Technical uncertainty

3. BUSINESS RISK:
    Affect the organization developing or procuring the software. Some business risks are:
    a. Excellent product but no one wants!! (Market risk)
    b. Building a product that no longer fits into the overall business strategy for the company. (strategy risk)

c. Building a product that sales force doesn't understand how to sell. (Sell risk)
d. Losing the support of senior management due to change in focus or a change in people. (management risk)
e. Losing budgetary or Personnel commitment (budget risk).

RISK IDENTIFICATION:

Risk Identification is a systematic way of discovering or specifying the risks or threats. Generally, risks are of two types:

1. Generic→ Potential threats to all software projects.

2. Product–specific → Potential threats that may occur while developing the specific software.

- So, early identification is important. To systematically identify important risks categorize them into classes. Risk can be identified by creating the checklists.
- Generic risks can be of following types:
  - Product size
  - Business impact
  - Customer characteristics
  - Process definition
  - Development environment
  - Technology to be built
  - Staff size and experience

  For eg: Checklist is prepared as a set of Components and driver of risks.

Risk Components and Drivers

Components are	Impacts Maybe
Performance	negligible
cost	marginal
support	critical
schedule	catastrophic

RISK PROJECTION:

It is often called risk estimation. It attempts to rate each risk in two ways, namely: probability of occurrence & impact/consequence of problem.

For this four projection steps are followed, which are:

- Establish a scale of probability of occurrence of risk.
- Impact of the risk is just described in detail.
- Impact of the risk in project and the product is estimated.
- Overall accuracy of the risk projection is noted.

It helps to prioritized risks. Resource is allocated in mitigation of this risk on the basis of prioritized list. Only risks above certain level are focused.

Three factors affect the consequences that are likely if risk occur i.e.

- Its nature – indicates problem

- Its scope – indicates distribution
- Its timing – when and how long?

STEPS OF DETERMINING RISK EXPOSURE:

1. Determine probability of occurrence for each risk component
2. Determine the impact for each component based on the criteria shown from the table
3. Prepare a risk table and analyze the result i.e. overall Risk Exposure (RE)
4. RE = P*C where, P is probability and C is cost to the project if the risk occur.

NUMERICAL QUESTIONS: FROM COPY

RISK RREFINEMENT:

It is mostly done when risk is occurred during outsourcing.

During the early stage of project planning, most of the risk is stated in general. As time passes and more is learned about the project, so project risk can be further refined so that it can be easy to mitigate, monitor, and manage. One method to do this is to represent the risk in CTC (condition-transition-consequence) format. i.e. is risk is stated in the following form:

Given that <condition> then there is concern that (possibly) <consequence>

- Example:

Given that all reusable software components must confirm to specific design standards and that some do not confirm, then there is concern that possibly only 70% of the planned reusable module may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30% of components. the above general condition can be refined in the following manner:

- Subcondition1: certain reusable components were developed by a third party with no knowledge of internal design standards.
- Subcondition2: The design standard for component interfaces has not been solidified and may not confirm to certain existing reusable components .
- Subcondition3: certain reusable components have been implemented in a language that is not supported on the target environment.

Hence refinement helps to isolate the underlying risks and might lead to easier analysis and response. But the refined sub-conditions remains the same (i.e. in the above example, 30% of the s/w components must be custom built.)