Software Risks and Control

Application software is becoming inseparable from the services which the software support and becoming basic infrastructure of modern by organisations as role of IT is moving from being supportive to the core of the business. It is therefore necessary that application software may be developed properly. However, the software development suffers from different types of risks. These risk threaten the quality of the software and thereby the business to the software support. It is therefore necessary to identify these risk and control them.

Software Engineering is fast emerging as a, disciplined which combines management techniques and technical skills to develop the software and manage this software' risk. There are many models suggested by practitioners in the field who have converted the process of software to science and later applied this scientific approach to introduce a new discipline of software engineering. There are two most popular models used to minimise this risk.

- SPR model suggested by software Productivity Research group promoted by Capers Jones.
- SEI-CMM model suggested by Software Engineering Institute, Carnegie Mellon University.

The SPR Model for Risk Control:

The SPR model further identifies a list of about 60 common risk factors as indicated in the annexure. Many of these risk factors may be related to each other whereas occurrence of one of the risk may lead to other risks. These risk factors can be classified broadly into risks arising out of Lack of proper measurement and estimation standards, inadequate planning, creeping user requirements. Inadequate management and technical tools and methods, lack of quality control etc.

The model classifies the software projects as under:

- MISprojects -Accounting system claims handling, etc.

- System software' projects such as operating systems, telecommunication systems or control software.

- Commercially marketed software projects such as spread sheets, word processors, CAD/CAM packages.

- Military software projects which are used by defence and are constrained to follow the standards laid down for this purpose.

- Contact or outsource projects undertaken by software organisations as per specifications of the user organisations.

- End user software projects where the software is developed in house rather than by a
professional programming staff. Due to their inherent organizational structure used in these projects, the susceptibility of these projects as well as their resistance changes.

Some of these risks can be controlled using technical and management tools but some other factors are resistant for such controls. following is the list of controllable and uncontrollable risks.

<table>
<thead>
<tr>
<th>Controllable risks</th>
<th>Uncontrollable risks</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Creeping use requirement</td>
<td>i. Excessive proper work</td>
</tr>
<tr>
<td>ii. Schedule pressure, long schedules and excessive time to market.</td>
<td>ii. Inadequate user documentation</td>
</tr>
<tr>
<td>iii. Cost overruns</td>
<td>iii. Low user satisfaction</td>
</tr>
<tr>
<td>iv. Low quality and error prone modules</td>
<td>iv. Friction between clients and contractors</td>
</tr>
<tr>
<td>v. High maintenance costs</td>
<td>v. Legal issues and litigation risks.</td>
</tr>
</tbody>
</table>

The organizational structure and skill levels of software organization changes as per the projects taken up by them. Therefore, these organizations have different levels of resistively as well as susceptibility to these risks.

Although organisational preparation is necessary to control all the risks, the limitation of controlling techniques have to be understood properly so that the probability of risk becoming reality can be minimised. The risk analysis and assessment methods are effective enough to identify significant problems and once the problems are identified and examined, solutions can be developed.

Out of 60 most common risks, Capers Jones has identified following risks as having serious impact on software projects.

- Inaccurate Metrics
- Inadequate Measurement
- Excessive Schedule Pressure
- Low Quality of Software
- Management Malpractice
- Inaccurate Cost Estimation
- Silver Bullet Syndrome
- Creeping User Requirement
- Low Quality of software
- Low Productivity of Development Team
- Cancelled Projects

The risks are discussed in brief hereafter.

1. **Inaccurate Metrics:**

In the earlier days of software engineering (prior to 1978), lines of code was used as the common measure of software projects which was bound to be inadequate and inaccurate for following problems:

- Lack of standardisation for any single language
- Lack of conversion rules across different languages
- Paradoxical behaviour while object oriented languages or automated code generators are used. (large code is generated with lesser efforts.
- Inability to measure non-coding items of work such as plans, specification, user document which constitutes nearly half the project cost for larger projects.
- Inability to measure software defects or defects in requirements, specification and non-code deliverables.
- Difficulty in determining source code size during requirements when the requirements is needed for estimation.

The function point metrics which is based on functionality was therefore found to be more suitable to economic studies, quality studies and value studies. Function Point measure is becoming more popular although its limitations are well accepted. Its limitations include subjectivity for deciding systems characteristics, omissions for important input, output and query requirements, non consideration to algorithmic or Logical complexities, etc. However, it can still be used as improvement over LOC and also as more reliable metrics for software project.

2. **Inadequate Measurement**

Whatever may be the metric used for measurement, there are associated problems with properly measuring the size or any other attribute of the software like cost, defects, efforts, etc.

**Leakage in effort measurement**

More often than not the overtime of programmers, managerial, efforts, administrative efforts of specialist, groups such as, Quality Assurance group, technical writers and technical writing are committee from the efforts estimation. Similarly efforts for project tracking are also not taken into account. Thus many of the efforts are not measured and not available for next projects for reference.

**Leakage in costing**
All the efforts which are not measured or recorded are not costed. eg. unpaid overtime, management efforts, users involvement during requirement analysis time for project management, review by top management, etc. are not costed at all. Whatever historical data available is seldom validated.

For in-house projects, the cost of the developers, users, testers, etc. is taken for granted and seldom included in costing. Similarly often the cost of changing or creeping user requirements is seldom taken into account as such changes are considered to be too trivial to have any impact on cost.

**Excessive Schedule Pressure**

SPR study indicates that irrational schedules and excessive schedule pressures are common project risks in over 65% of the large projects. This happens because the top management do not have proper notion about the workload or activities involved in the project. Unrealistic schedules are imposed on the development team and this leaps to many other risks such as:

- Poor quality of end product due to short circuiting procedures and hurried implementation.
- Low, morale, excessive fatigue burnout and high attrition rate among software personnel.
- Cancelled projects due to lack of proper estimation and cost involved in ... project development, etc.

The main factor contributing to the excessive schedule pressure are:

(i) Ad-hoc and arbitrary method of scheduling and (ii) Ever changing requirements of the user till the completion of the projects.

The in-house project development is most susceptible to such excessive schedule pressure. larger projects are more susceptible than smaller ones proper requirement analysis and user commitment, proper understanding of tasks involved and proper use or scheduling techniques are important tools for controlling these risks.

4. **Management Malpractice**

Although rarely discussed in risk management techniques, this factor is one of the critical problems. The root cause for this risk is that the managers are seldom trained for job of managing software projects and rarely having, basic technical skills needed for undertaking software projects following’ six skills are identified by Dr. Jones for proper management practices to be followed.

I. Software sizing
II. Software effort and cost estimation
III. Software planning
IV. Software project tracking
V. Software ‘effort and cost measurement
VI. Assessment of project deliverables

The organisations which train their managers and adopt proper techniques in these areas can control this risk to substantial extent. The impact of this factor is normally not assessed in the software literature or models suggested by the software engineering institutes.
5. Inaccurate Cost Estimating

Although there are commercial software estimation and planning tools, there usage is not common and most of the organisations do not have practice of project planning. This factor is responsible for cost overruns and schedule slippages that occur. The organisations may use automated or manual tools for software cost estimation. However, it is necessary to have well defined organisational procedures and practices to cost the software projects as well as system of reporting the consumption of resources (namely the cost incurred) which can enable the organisation to cost and control the software projects.

6. Silver Bullet Syndrome

This risk arises mainly in those organizations which adopt one of the models developed for the purpose of software development. Normally moving to CASE technology, TQM, Object oriented paradigm, etc. leads to improve productivity and quality. However, their implementation requires proper planning and applications within the organizations. Many of these techniques need to be applied parallely. Similarly, more often; the productivity gains made by the vendors are exaggerated and need careful assessment. Blind adoption of these tools and treating them as panacea for all the problems laced for software development itself is a major source of risk. More often, these tools tend to become end in themselves without any tangible achievement:

It is necessary for organization to adopt these tools but not treat them as panacea for all problems. Proper understanding of the it functionality and proper training are necessary while using such tools.

Creeping User Requirements

One of the killer risk for software projects is the continuously changing requirements of the user. It is empirically estimated that users requirement changes at the rate of 1% per month. Ambiguity regarding the nature of output and input as well as communication gap between the developers and the user are cause of this risk. It can be controlled following using for requirement analysis techniques and also, introducing proper change procedure. Documenting the impact of changing requirements in terms of function points and time/cost implications help controlling this risk for end-user or MIS projects, Joint Application Development or (JAD) act as important control techniques.

8. Low Quality

End software with number of detects poses a serious risk for software projects the 'Defect Removal Efficiency' or ability of the software development team to fix the bugs along with number of bugs found decides the quality of the software. Low level of quality of software leads to low level of user satisfaction as well as programmers productivity and loss of acceptability.

Following "Software Quality Assurance" and "Software Quality Control" (Software testing) can minimise this risk significantly.

Low Productivity

Low productivity of the development team is one of the serious risks to the software projects. This risk arises due to different factors such as skill levels of developer, proper training, exposure and experience usage or proper development tools, following proper development methodology, etc.
Low productivity leads to schedule slippage or extended schedules, increasing cost and even lower quality of final product. Proper development of the human resources and adoption HR practices using proper development tools and development methodology can control this risk.

10. Cancelled Projects
As the software systems become more complex and increase in size the probability of their getting cancelled also increases. Large systems in excess of 10,000 function points such as developing operating systems, telecommunications systems, major defence system show higher probability getting cancelled (around 50%). It is observed that an average cancelled project is normally one year behind the schedule or has over 100% cost overrun at the time of cancellation.

The major factors for projects getting cancelled are over ambitious large projects, inadequate estimation and costing inadequate planning and tracking. It is therefore advisable to take up projects which can be properly implemented with available technical and managerial resources.

Impact of Top 10 Risks
These ten factors are some of the many factors and can be considered to be 'tip of the iceberg' since manifestation of these problems or risks becomes possible due to "many other interrelated, factors which may be minor in nature. These risk factors indicate potential dangers which the organizations need to take care through vigorous, assessment of organizational processes, introduction of accurate metrics and measurement techniques and multi-threaded process improvement programmes. The adoption of software engineering is one of the answers for various risks faced by the organizations since risk-management is important subdiscipline of software engineering. The SPR assessment states that in the organization is threatened by any four of these risks, it will not be able to implement any software projects successfully.

SPR Approach for Risk Control
The Software Productivity Research Group has adopted a 'clinical approach' towards managing the software risk. This approach is used in encyclopedia of control of communicable diseases in Man" which adopts a uniform approach to identify the problems or symptoms, its probable impact, suggests short term and long term remedial measures. The structure of SPR clinical model adopts following structure for risk control

Definition of the risk
Severity
Frequency of risk occurrence
Occurrence of risk
Susceptibility and resistance
Root causes
Associated problems
Cost impact.
Methods of prevention.

The SPR model is useful in many ways to understand the risks and ways to control these risks.

**SEI Capability Maturity Models for Risk Control**

The Software Engineering Institute has proposed the popular capability maturity model in which the organization acquire maturity through improving their internal processes. The focus of the model is on improving the processes and introduce element of discipline in each activity and institutionalise the process of change. It seeks to effect the cultural change within the organization which would help handling the risks. The SEI model views the risk and processes as complimentary to each other and intricately interwoven like warp and welt of the fabric.

The risk relates to whether we have the right product whereas process addresses how we can get the product right. The risk concentrates on the end product whereas process focuses on the activity.

Whereas change management in the risk, the process is the way if which change should be initiated.

There is always a risk of failure associated with innovation and new paths. The process seeks to mitigate the risk by conforming to standards whereas the risk relates to uncertainty process relates to control over the situation.

The SEI model defines risks as a future events with a probability of occurrence and potential for loss, by similar definition a problem by similar definition a problem before the organization is the risk which has become the reality. The principle of software risk management here is that with a timely discovery, risks can be avoided, eliminated or have their impact lessened.

Informed decisions by consciously assessing what can wrong and what can be the severity of its impact.

The SEI ranks the software risks in a descending order as under:

1. Incorrect Resources estimation
2. Ambiguous requirements
3. User/Customer uncertainty
4. Inadequate management process.
5. Improper design risk
6. Development system and risk with development system
7. Improper work environment

The SEI identifies following risks associated with any technology related project:

- Lack of strategic framework or conflict over strategy.
- Lack of adaptation to technological change Supplier/vendor problems
- Supplier/vendor problems
- Poor management of change
- Too much faith in ability of the technology to fix the problems
The risk management paradigm suggested by the SEI is as under.

1. Identify actionable risks, prioritise the risk and manage the risks. For this purpose, organization should seek. View of all individuals and also seek information, through multiple sources.

2. Analyze the risk to decide which risks should be addressed and those of which risks are to be addressed first.

3. Plan for the risk by taking specific decisions about addressing risks. The plan also consists and assigns resources for risk management. The purpose of the plan is to take action, decide measuring success and assign responsibility for addressing and monitoring the risks. The planning therefore includes establishing due dates, fixing responsibilities, tracking and control system, identifying interdependence of tasks and people and definition of configuration of the system.

4. Risk tracking is to follow through the project and document the data on risks. This data acts as the basis for taking decisions. It should provide visibility of risks and mitigation also ensure that the risk is being managed.

5. Risk control includes specific decisions based on risk tracking data. The risk control acts as the repository of decisions made and action taken with reference to various difficulties/risks encountered.

6. Risk communication: Risk communication is the common thread passing through all the five risk control activities discussed alone. Communication is necessary to ensure that risks and mitigation plans are understood by all the information on risk is readily available. It also ensures effective on-going dialogue between the management and the project team.

To manage the software risk, following steps are suggested by SEI

1. Decide upon the measure for success of the project.
2. Identify top five or top ten issues which may prevent the project from being successful.
3. Decide importance of each of these issues.
4. Decide the actions necessary to address these issues.
5. Decide the timing for these decisions.
6. Decide the boundary as well as people needed to be involved in these decisions.
7. Decide information needed to ensure effectiveness of the decisions.
8. Openly share and communicate the issues involved.

**SEI Models for Project Management**

SEI considers process improvement as the main response to all the problems. It has therefore suggested following models to improve the organizational processes and thereby organizations capability.
- Capability Maturity Model (CMM) which identifies five evolutionary levels of capability maturity for organization. The organizations achieve higher level through improving its processes and practices. The emphasis of the model is not only on process improvement but cultural change through institutionalizing the process improvement and culture of continuous improvement.

- People Capability Maturity Model (P-CMM)

- This model is similar to process capability model but focuses on people within the organization. It seeks to suggest framework for improving the Human Resources - policies, or the organization so as to integrate the personal aspirations with organizational goals. As that incase of process improvement mode, the thrust is again on institutionalization of the processes' and cultural change towards people management.

- Personal & Team Software Process Model (PSP & TSP)

These models are aimed at improving the efficiency and productivity of individuals and build processes which would build effective teams. The focus here is on, sustained personal improvement through personal process with organizational support.

SEI models are initiated and adopted U.S. Department of Defense and have popularity all over the world. The Europe has adopted slightly modified version of SEI models. In India, SEI models are most popular and most of the software organizations are trying to get SEI certification. India also has largest number of SEI level - 5 companies. SEI models can be used not only for software development but for improving organizational processes.

Relevance of Risk Control Models for banks and financial institutions:

As the financial sector gets ready and takes up large investment in software development, it is desirable that they are aware of the risk factor involved in software development. The development could be 'in-house' or 'outsourced'. However, proper understanding of the issues and risks involved in the software would help the management to take remedial action in time.

Most Common Software Risk identified by SPR MODEL

1. Ambiguous Improvement targets
2. Artificial Maturity levels,
3. Cancelled Projects
4. Corporate Politics
5. Cost Overruns
6. Creeping User Requirements
7. Crowded Office Conditions
8. Error Prone Module
9. Excessive Paperwork
10. Excessive schedule Pressure
11. Excessive Time to Market
12. False Productivity Claims
13. Friction between Clients and Software Contractors
14. Friction between Software Management and Senior Executives
15. High Maintenance Costs
16. Inaccurate Cost Estimation
17. Inaccurate Metrics
18. Inaccurate Sizing of Deliverables
19. Inadequate Assessments
20. Inaccurate Quality Estimating
21. Inadequate Compensation Plans
22. Inadequate Configuration Control and Project Repositories
23. Inadequate Curricula for Software Engineering
24. Inadequate Curricula for Software Management
25. Inadequate Measurement
26. Inadequate Package Acquisition Method
27. Inadequate Software Policies and Standards
28. Inadequate Software Project Risk Analysis
29. Inadequate Tools and Methods for Project Management
30. Inadequate Tools and Methods for Quality Assurance
31. Inadequate Tools and Methods for Software Engineer
32. Inadequate Tools and Methods for Technical Document

33. Lack of Reusable Architecture
34. Lack of Reusable Code

35. Lack of Reusable Data

36. Lack of Reusable Design
37. Lack of Reusable Documentation

38. Lack of Reusable Estimation Template
39. Lack of Reusable Human Interface

40. Lack of Reusable Project Plan

41. Lack of Reusable requirements

42. Lack of Reusable Test Plans, Test Cases and Test Data

43. Lack of Specialization.

44. Long Service Life Obsolete System

45. Low Productivity

46. Low Quality

47. Low Status of Software Personnel and Management

48. Low User Satisfaction

49. Malpractices of Project Management

50. Malpractices of Technical Staff

51. Missed Schedules,

52. Partial Life Cycle Definition

53. Poor Organizational Structure

54. Poor Technology Investment

55. Severe Layoff and Cutbacks of Staff

56. Short Range Improvement Planning

57. Silver Bullet Syndrome

58. Slow Technology Transfer

59. Inadequate Research & Reference facilities

Prepared by Shri A.K. Hirve, FM
Reprepared by Shri V.G. Sekar, FM, CAB, Pune