4. SoftRisk Analysis and Requirements Definition

4.1 Introduction

In order to achieve a success in software development projects, it is recommended to have a complete understanding of software requirements. Requirement analysis can be thought of as a task bridging the gap between system-level software allocation and software design. It is important to specify software functions and performances, and indicate the interface of the software (Pressman, 1997).

Having investigated and surveyed existing software risk management tools, several numbers of weaknesses were found. A prototype tool to tackle these weaknesses was decided. The main requirement will be detected in this chapter.

A technique to manage software risks has been designed based on: the Top Ten software risk items, proposed additional top seven risk items, risk exposure equation, and some new proposed formulas. It will be introduced in this chapter.

4.2 Information Description

4.2.1 Problems and Motives

According to the literature and the survey's results, there are many problems and motives that lay beyond building this prototype. Some of them are related to the existing risk management tools, while, the others are related to the software industry.
Some of these motives and problems are expanded below:

- The fast growth in the software industry brings many new risks into the field. Therefore, competing in this field becomes very difficult without effective tools to encounter and reduce the impact of the consequence risks.
- Due to the complexity and the size of the software projects they can be considered as risky projects especially in terms of time and cost. This has risen the demand for automated software risk management techniques.
- Only very few tools deal with the software risks. Most of these tools are concentrated on risk analysis with emphasising on the financial situations. The other topics of the software risk management are ignored or not covered enough.
- Due to the cost of developing and purchasing, the available software risk management tools are applicable to only large-scale software projects, which can cover the cost. Small and medium scale projects are still in trouble, because they can not cover the cost so they face the risks without effective weapons.
- Software risk management tools should be updated and improved to follow up the updates in the software industry.
- It is difficult to predict the future without the historical data and statistical analysis of previous experiments. However, existing tools do not support the use of the historical data and statistical analysis.
• Documentation is a very important issue and could help to avoid a lot of future crisis. Most of the available tools do not concentrate on this topic as an essential topic in the software risk management techniques.

• Representing the risk situation in a graphic form makes the monitoring task much easier and friendly. The existing tools do not support that.

• The techniques which are used by the most available tools are obscure or made complex, therefore, only few developers can understand and familiar with them (Greer & Bustard, 1996).

• Using RE values to prioritise and detect the top risks and then concentrating on them could save the developers time and effort. So, corrective actions can be taken earlier. However, existing tools face the problem of none-usage of the top risks and risk exposure technique.

4.2.2 General Requirements

The requirements of SoftRisk prototype have been specified after investigating existing software risk management approaches, interviewing and discussing with software developers, evaluating existing tools, and conducting software risk management survey. The general requirements of the SoftRisk prototype can be summarised in the following points:

• SoftRisk should be able to monitor software development risks.
• Before conducting risk-monitoring step, other risk management steps must be conducted. Therefore, SoftRisk is intended to perform all risk management steps.

• Risk monitoring should be supported with a line graph. The monitoring should always be concentrated on top ten risks at any given inspection-time. Top ten risk items are changeable depending on their impact on the projects.

• All identified risks should be documented.

• SoftRisk should provide assistance to the user to estimate the probability and the magnitude of the risk in a qualitative data form (from user stands point) and quantitative data form (for the internal calculations).

• SoftRisk should introduce reduction advice to prevent or at least reduce the impact of the risk. Reduction actions should be documented after being performed.

• SoftRisk prototype, should be able to prepare risks data in a form that can be used for statistical purposes. This is in order to:

  - Avoid such risks in future projects.
  - Identify positive and negative points in software development projects and software risk management approaches.
  - Help in detecting relations and similarities amid any sort of software risks.
- Determine the efficiency of reduction techniques and factors that lead risks to increase or decrease.

- It would be better if SoftRisk is provided with an on-line help and some passwords for security reasons.

4.3 Functional and Behavioural Description

SoftRisk can be broken down into four main components. Following, are the descriptions of what each component should contain:

4.3.1 Identification, Estimation, and Documentation

Identifying, estimating, and documenting any software risk should be carried out for any risk that threatens the lifecycles of software development projects. However, the probability and magnitude of any identified risk should be estimated, and documented for future needs. All related risk's data are considered as elements of the risk matrix, which should be documented too. The following components should be involved to cover this task.

4.3.1.1 Risks' Database (RskDB) Manager

Risks' database manager deals with the first part of the risk matrix elements. It should be provided with a special database called RskDB. The documented data in this database must be the most fixed risks' data, which can be used by any software project.
It should be provided with components which allow the user to enter data about any type of software risks, which are likely threats to software development project.

These data can be gathered from all available resources (e.g. Top Ten risks checklist, additional top seven risks, software surveys, projects being developed, developer experiences, historical or past project experiments data, publications, Internet, or even news).

The gathered data makes risk identification task much easier for future software development projects. Furthermore, it could be used for any statistical operations. The following features and components should be provided:

- Saving and retrieving functions (i.e. Adding, Selecting, Displaying, Updating, and Deleting).
- Pull-down menus to specify project type, and the phases that could be affected by the desired risk.
- Radio buttons to allow the user to detect the red lines for invoking contingency plan, and crisis plan. The user should be able to indicate the risk acceptable level and the highest allowed level for each risk.
- Along with entering risk’s data, there should be a call to risks’ probability and magnitude estimators in order to prepare estimation checklist with weight for each item.
- Reduction techniques (i.e. mitigation, contingency, and crisis plan techniques) should be entered concurrently with the other risk’s data.
4.3.1.2 Projects' Risks File (PRSF) Manager

This part deals with the second part of risk matrix elements. The main goals of this part are to identify, estimate, and document the risks, which threatens any given project. It should be supported with a file called PRSF to track and document all specific projects risks data. The documented data are changeable based on the project and risks situations. The following features and conditions must be supported:

- At entering time, a list of all registered risks in the RskDB should be displayed to the user if he/she wants to select some or all risks to deal with. In case of no selection (by default), all risks are considered to be selected. When the user starts entering data into PRSF, only those selected risks appear.
- SoftRisk should check the uniqueness of the project code.
- Each project should have its own password.
- Risk code should link PRSF to RskDB.
- If the desired risk is new, SoftRisk should automatically register it in RskDB before dealing with it. Then, the Risk-code should be automatically invoked into PRSF record. But in the case where it is already registered in the RskDB, then straightforwardly, its code is invoked into PRSF.
- Pull-down menus should be used to select project type, size, and phase.
- Determining risk probability and magnitude should be done in the easiest style, i.e. using qualitative form (Radio buttons are the best choice for this
purpose). Furthermore, the user must be able to get the probability and magnitude estimation assistance once he/she needs it (more details in coming section).

- Record oriented functions like adding, selecting, displaying, deleting, and updating must be supported.

- The prototype should not only specify the inspection number, but provide the user with three options: current, previous, and next inspection number to specify.

- Risk-code should be invoked automatically. The PRSF manager asks RskDB manager to show a list of risks based on the following options and conditions:

  - If the user has already selected certain risks to deal with, only those risks should appear in the list, otherwise, all risks will appear.

  - If the user selects any risk, the risk-code should be automatically invoked in PRSF record.

  - Based on the user’s request, he/she should be able to edit the risk record, before selecting it.

  - In case the risk is quite new and has not been registered in the RskDB, the PRSF manager should call RskDB manager to register it before using it.
4.3.1.3 Probability and Magnitude Estimator

Probability and magnitude estimation tasks are always very difficult to undertake. Some developers try to do so, but only in terms of qualitative data. Therefore, *SoftRisk* should provide an optional probability and magnitude estimation assistance. This option must be divided into two components; one for preparing this help while the other is for utilising it.

a) Probability and Magnitude Estimator Preparation

First, the probability and magnitude estimator checklists should be prepared in conjunction with entering RskDB records data. A probability and magnitude checklist must be prepared for each risk. The checklist items should be weighted. Table 4.1 shows a sample of the probability and magnitude estimation checklist, whereas, complete probabilities and magnitudes estimation checklists of the top risk items are presented in Appendix B.

Meanwhile, the prototype of *SoftRisk* should be initialised with estimation checklists about both the *Top Ten Risk Items* (Boehm, 1989-a) and *Additional Top Seven Risk Items* (Hashim & Keshlaf, 1997-a) (See Chapter 3).

b) Probability and Magnitude Estimator Utilisation

This component is similar to the preparation component, but it accepts user’s ticks on positive elements of the checklist and then the prototype evaluates and introduces an estimation value of the probability or the magnitude. This part should
be ready to accept and reply calls whenever the user needs this help (e.g. at PRSF or controlling part). However, the usage of this estimation service is optional.

Table 4.1: A Sample of Probability and Magnitude Estimation Checklists

<table>
<thead>
<tr>
<th>Risk Item: Personnel Shortfalls</th>
<th>Factor Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do the staff members have inadequate experience in similar projects?</td>
<td>30%</td>
</tr>
<tr>
<td>They did not attend any enhancement training courses, or teambuilding</td>
<td>10</td>
</tr>
<tr>
<td>They are not educated enough</td>
<td>20</td>
</tr>
<tr>
<td>The staff can not adapt to the advanced software very fast</td>
<td>10</td>
</tr>
<tr>
<td>The project needs special and critical skills</td>
<td>20</td>
</tr>
<tr>
<td>The management does not provide teambuilding programs for the project staff</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Magnitude Factor</th>
<th>Factor Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>The organisation can not support its staff with any kind of help may need</td>
<td>10%</td>
</tr>
<tr>
<td>The project requirements can not be handled by the staff ability</td>
<td>.</td>
</tr>
<tr>
<td>The problem recovering takes a long time</td>
<td>40</td>
</tr>
<tr>
<td>Will this risk cost the project more than 30% of its budget to recover?</td>
<td>5</td>
</tr>
<tr>
<td>Are there more than two parts that may be affected by the risk?</td>
<td>15</td>
</tr>
<tr>
<td>Is the project a real time project?</td>
<td>10</td>
</tr>
<tr>
<td>The project will not get enough people on time</td>
<td>15</td>
</tr>
</tbody>
</table>

4.3.2 Assessment and Prioritisation

After the risks are identified, estimated, and documented, it is the time for assessment and prioritisation steps. SoftRisk should be able to assess the projects risks based on their probabilities and magnitudes (RE equation must be used).

For the selected project all risks should be assessed and prioritised based on their RE values. Once the risks are assessed and prioritised, a list of top ten risks should be prepared for each inspection. The list will be the input of the monitoring and
controlling components. It is not a fixed list and risks could be different from inspection to inspection based on RE values.

4.3.3 Monitoring and Controlling

As mentioned above, the main task of SoftRisk is to monitor software risks. Previous components support this part by preparing the data. The following are the descriptions of monitoring and controlling components and tasks:

4.3.3.1 Monitoring

The monitoring in SoftRisk should be carried out on the priority list which is suggested by the assessment and prioritisation component. The monitoring should be supported with a line graph. The graph should be divided into three zones; green, yellow, and red, for negligible, medium and catastrophic risks respectively. The graph should represent project’s risks, but only top ten risks for each inspection are represented. The x-axis of the graph represents the inspection numbers whereas, the y-axis represents RE values. The graph should also be in scrolling form.

4.3.3.2 Controlling and Re-assessment

SoftRisk should introduce reduction advice to prevent or reduce each risks impact. After executing the risk reduction technique, a re-assessment operation should be performed to ensure that all of the risks are under control.
The controlling and re-assessment operation should involve the following approaches and options:

- A list of all inspection numbers should be displayed and then the user selects one to work on.

- Once the user specifies the inspection number, another list of top ten risk items of the specified inspection must be displayed. This list is considered as a priority list of the risks, which are registered under the desired inspection number.

- The project, duration, and risk are made known now, then the next step is to display all risk’s relevant data which are important for the controlling operations which could include:
  
  - Risk name, cause and description.
  - Mitigation, contingency, and crisis plans along with invoking red lines.
  - Existing risk exposure value.
  - Risk acceptable level.
  - Highest allowed level of the risk.

- After performing any of the reduction techniques the user should be able to re-enter the new estimated values of the probability and magnitude of the risk (the estimation help must be available).
Re-assessment operation should be performed after executing any of the reduction techniques. Risk change (RE_{change}) and risk change value (RCV) should be computed to obtain more evaluation of the risk situation. Following are the formulas proposed in this research to get much sensitive assessment for the risks (Hashim & Keshlaf, 1997-a):

\[ RE_{old} = LP \times LM \]  
\[ RE_{new} = LP_{new} \times LM_{new} \]  
\[ RE_{change} = RE_{new} - RE_{old} \]  
\[ RCV = RR_p \times RR_m \]  
\[ RE_{current} = RE_{old} - RCV \]

where;

- RE is Risk Exposure.
- LP is Loss Probability.
- LM is Loss Magnitude.
- \( RE_{new} \) is the RE after applying risk reduction technique.
- \( LP_{new} \) is the latest loss probability after applying risk reduction technique.
- \( LM_{new} \) is the latest loss magnitude after applying risk reduction technique.
- \( RE_{change} \) is the difference between \( RE_{new} \) and \( RE_{old} \).
- \( RE_{current} \) is the current RE value.
- RCV is a Risk Change Value (i.e. amount of reduction). It can be calculated by multiplying the RRₚ (i.e. Reduction amount in risk probability) by RRₘ (Reduction amount in risk magnitude).

In fact, these equations have been proposed to observe the efficiency level of the reduction techniques. The aims of the proposed formulas (formulas 2 to 5) are as follows:

- Formula (2) is to obtain the latest impact of the risk.
- Formula (3) is to observe the changes between the old risk impact and the latest one.
- Formula (4) is to get the risk change value. This value will be obtained after performing the reduction technique.
- Formula (5) is to get the current risk impact, which is the difference between the old risk impact and the risk change value.

The monitoring and controlling operations should continue until the end of the project lifecycle to ensure that all risks are under control and within the acceptable level.
4.3.4 Statistics Preparation

The preparation of statistical data will be very simple in this prototype. It must include the following parts:

4.3.4.1 Risks Threats

It is a significant approach because the risks, which threaten the projects, can be detected. The development phases that are infected by each risk will be detected as well, so the relation between the risks and phases will be observed. A table should be produced with entitled columns i.e., risk code, risk name, risk description, and risk phase. However any one of these titles can sort the table.

4.3.4.2 Risks Frequency

In this part the user can select all or some projects to see how many times each risk has threaten those projects. The risks with the highest frequently number can be highlighted to avoid or find effective reduction techniques to mitigate their impact. It is also important to know what sort of project they attack, and which projects that have the highest number of risks to detect the relation between the risks and projects types.

4.3.4.3 Risks Sorting

Once the user gets a sorted list of all probabilities and magnitudes of the risks (i.e. the risks sorted by RE values), he/she can realise the degree of severity of each
risk. So a suitable reduction technique can be prepared for future inspection and the risk can be taken in the account. Therefore, he/she can concentrate the effort in coping with certain risks rather than on random number of risks. This saves developers' time and effort. Therefore, SoftRisk should contain a part to handle this task.

4.3.4.4 Risks Infection Impact

A component allows the user to specify a sort of risks to trace their infection amongst the projects. Other risks that affect any one of those projects should be listed. This part is important to know the interaction amongst the risks, and their behaviour within the projects.

4.3.4.5 Risks Similarity

It is very important to determine the similarity amid the projects’ risks. Therefore, the resolutions of the risks will be much easier, and can be shared among the projects. SoftRisk should find this similarity amongst a group of projects, group of phases, or among all projects and phases.

4.4 Overview on SoftRisk Steps

As stated in chapter 2, there are various groups of risk management steps. SoftRisk introduces an improved group of software risk management steps
(technique). These steps are expanded below (Keshlaf and Hashim, 1998; Keshlaf & Hashim, 1999):

**Step 1:** Risk Identification

**Step 2:** Risk probability and magnitude estimation

**Step 3:** Risk Documentation

**Step 4:** Risk Assessment (computing RE)

**Step 5:** Prioritizing and highlighting highest ten risk items

**Step 6:** Monitoring (Graphical representation)

**Step 7:** Controlling and re-assessing (Perform a suitable reduction technique, compute risk change (RE_{change}) and risk change value (RCV), and documenting the new risk situation and reduction actions).

**Step 8:** Perform statistical operations if need be, then go back to step 1.

As can be seen in Figure 4.1 SoftRisk introduces a somewhat continuous risk management along the overall project lifecycle. So, all risks will be under control to enable the project to achieve its goals (i.e. no overrun, no over cost, and introducing a satisfied product within the required quality) the following is an explanation of each step.

4.4.1 Step 1: Risk Identification

Risk identification is always the first risk management step. Therefore, software developers have to identify software risks before starting any other risk management
steps. Two forms should be completed to identify any software risk. The first form contains a general risks data that can be used by any software development projects, whereas the second form contains specific project's risk data.

Once the two forms are filled out, the risk is already identified. It must be noted that the fields of the two forms can be considered as risk matrix elements.

![SoftRisk Steps Diagram](image)

**Figure 4.1:** *SoftRisk* Steps Diagram (Continuous Inspection and Monitoring)
4.4.2 Step 2: Risk Probability and Magnitude Estimation

The risk assessment is accurate as long as the risk probability and magnitude estimation are accurate. Probability and magnitude of each risk must be estimated. Hence, a suggested mechanism to estimate both probability and magnitude can be used. Based on the mechanism, for each risk there will be a special checklist whose elements are pondered (i.e. have weights). The developer will have to choose among the checklist elements, then, estimation value will be produced.

4.4.3 Step 3: Risk Documentation

Documentation is a very important element for risk management. It can be very helpful at least for:

- Tracking software projects situation from risk perspective
- Statistical operations
- Future risks predictions

The documentation operation should cover all risks data, be it specific project data, general risk data, and reduction actions.

4.4.4 Step 4: Risk Assessment

Risk assessment is an essential part for any software risk management approach. The simple and famous risk assessment formula (Risk Exposure formula) can be used (Boehm. 1989-a). Other formulas can also be used (Hashim & Keshlaf, 1997-a).
4.4.5 Step 5: Prioritisation and Highlighting Highest Ten Risks

Concentrating on the highest top risks can tremendously save developer’s time and effort (Greer & Bustard, 1997-b). The technique lists and all sorts of risks based on RE, then the highest ten must be highlighted at each risk management inspection cycle. However, they are changeable based on their RE.

4.4.6 Step 6: Monitoring (Graphical Representation)

Representing data by graphics is much better than using report form. Therefore, refined top ten risks should be represented by a line graph to ease their monitoring and comparison. The graph should be categorized into three zones representing the severity of the risk, green, yellow, and red for negligible, medium, and catastrophic impact respectively.

4.4.7 Step 7: Controlling and Reassessment

At this step, the developer should perform a suitable reduction technique to prevent or reduce the risk impact. After that, risk assessment equations should be performed to reassess the risk level. The equations could include, among others, risk exposure, $RE_{change}$, or risk change value (RCV).
4.4.8 Step 8: Performing Statistical Operations and going back to step 1

Any useful statistical operation can be performed at this step. If the project is still ongoing, it is highly recommended to continue monitoring risks in order to keep them under control and safe the project from any other consequences.

4.5 SoftRisk's Implementation Requirements and Environment

Developing the SoftRisk prototype requires some hardware, software, and support resources to achieve the prototype goals and satisfy its requirements. The required resources are stated below:

4.5.1 The Hardware

The primary hardware that is needed to implement the SoftRisk prototype involves the following items:

1- A Computer with 200MHz speed, 32 MB RAM, 1GB HD is preferred. It could be Sun workstation or PC.

2- Computer printer and Modem or network card for Internet use.

4.5.2 The Software

The primary software that is needed to develop the prototype includes the following items:

1- Internet browser.  2- Java language compiler.

3- Web browser, and web server.  4- Database server and database bridge.

5- Some Java classes library especially for graphics.
4.6 Conclusion

After investigating and surveying the existing approaches and tools of software risk management in the previous chapters, many weaknesses have been discovered. Therefore, it has been decided to tackle those weaknesses by building a prototype. The motives and problems that lay beyond the building of the prototype were stated in this chapter.

General requirements of the prototype have been specified. Meanwhile, the functional and behavioural descriptions of the proposed prototype were expanded. On the other side a checklist mechanism to estimate risk probability and magnitude have been presented and new formulas to improve risk assessment sensitivity were presented as well.

In this chapter a new group of steps (technique) to manage software risks is introduced and described. This technique includes a new step called Risk Documentation. In order to ease risk management and save developer time and effort, the technique depends on RE, top risk items and graphics representation to assess and monitor software risks. Based on the technique concept the risk management process should continue until the end of the project to ensure that all of risks are under the control and the project is safe. Finally, the implementation environment is also specified in terms of hardware and software specification. Once the requirements have been specified, the next step is the design of the system. The coming chapter will be assigned to analyse and design the prototype.