5. *SoftRisk* Analysis and Design

5.1 Introduction

This chapter describes the design of *SoftRisk* prototype that will manage software development risks. Their main classes and some user interface design will be described.

5.2 Description of *SoftRisk* Design

*SoftRisk* has mainly been targeted to monitor software risks. However, the user has to go through a set of software risk management steps before reaching the monitoring steps. Therefore, *SoftRisk* is intended to cover most of the risk management steps to reach the monitoring step.

5.2.1 Risk Matrix

For identification and assessment purposes, the design involves a risk matrix. The matrix can be seen as the kernel of the prototype. The user has to fill out some fields which are used by the prototype to identify the risk. These fields feed directly projects’ risks file, risks database or both. The Risk Matrix contains the most important data that is related to the risk. (Table 5.1 illustrates an example of the risk matrix elements). The matrix data will be gathered from any available resources e.g. projects under processing, past projects’ data, special surveys, expertise, publications, and Internet.
Table 5.1: Risk Matrix Example

<table>
<thead>
<tr>
<th>Risk Code:</th>
<th>TR14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Item:</td>
<td>System complexity.</td>
</tr>
<tr>
<td>Risk Type:</td>
<td>System risk.</td>
</tr>
<tr>
<td>Risk Phase:</td>
<td>Implementation phases.</td>
</tr>
<tr>
<td>Risk Cause:</td>
<td>Tool and language limitation, lack of expertise or lack of the information.</td>
</tr>
<tr>
<td>Prob. of Occurring:</td>
<td>Low</td>
</tr>
<tr>
<td>Risk magnitude:</td>
<td>High</td>
</tr>
<tr>
<td>Risk Responsibility:</td>
<td>Developer xxxxxxxx</td>
</tr>
<tr>
<td>Risk Mitigation technique and its requirements:</td>
<td>Buying special tool, which needs additional budget (15% additional budget).</td>
</tr>
<tr>
<td>Contingency Plan and its Requirements:</td>
<td>Call some experts from other companies to implement this part.</td>
</tr>
<tr>
<td>Contingency Plan Red Line:</td>
<td>e.g. RE is HIGH.</td>
</tr>
<tr>
<td>Crisis Plan:</td>
<td>Terminate the execution of the infected part.</td>
</tr>
<tr>
<td>Crisis Plan Red Line:</td>
<td>e.g. RE is Very HIGH</td>
</tr>
<tr>
<td>Highest Allowed Level:</td>
<td>e.g. RE is V. HIGH.</td>
</tr>
<tr>
<td>Risk Acceptance Level:</td>
<td>e.g. RE is Low</td>
</tr>
<tr>
<td>Current probability:</td>
<td>High</td>
</tr>
<tr>
<td>Current Magnitude:</td>
<td>Low</td>
</tr>
<tr>
<td>Reduction actions that have been taken:</td>
<td>Two programmers have been hired to perform some complicated parties</td>
</tr>
<tr>
<td>Person in charge:</td>
<td>Programmer number 3</td>
</tr>
</tbody>
</table>

5.2.2 SoftRisk Main Classes

The design of SoftRisk is based on Booch model (Booch, 1994). Figure 5.2 shows the main classes of SoftRisk and the relations amongst them. In the following, the main classes and their goals are briefly described. However, it should be noted that, for the sake of simplicity, many other secondary classes are omitted from the figure.

In addition Figure 5.1 shows the icons which are used for designing the SoftRisk classes diagram. The relationships include (1) Inheritance means that one class shares the structure and/or behaviour defined in one or more other classes (2) Association denotes a
semantic dependency and without specifying the direction of this dependency, or the exact relation. (3) Using relationship indicates that the client in some way depends upon the supplier to provide certain services. It also indicates which abstraction is the client and which is the supplier of certain services (Booch, 1994).

![Class Icon](image)

**Figure 5.1:** Class and Relationships Notation

### 5.2.2.1 SoftRiskStarter Class

**Class Name:** SoftRiskStarter  
**Attributes:**  
Tool_PassWord  
**Operations:**  
Entrance ()  
PassChecker ()

**Responsibilities:** Triggers the tool, and checks the entrance password

### 5.2.2.2 OperationsSelector Class

**Class Name:** OperationsSelector  
**Operations:**  
Start ()  
Finish ()

**Responsibilities:** Its main purpose is to select the target operation and then start and finish it.
5.2.2.3 Risk_Documentation Class

Class Name: Risk_Documentation
Attributes:
Risk_Code
Risk_Name
Risk_Description
Operations:
RiskDoc()
GetRisk()
CheckCode()
SpecifyCode()
PutRisk()

Responsibilities: The main goal of Risk_Documentation class is to handle the documentation of generic risk data (risk code, risk name, risk description, reduction
techniques, etc.) which can be used by any software development project. It accepts any risk data from any available resources, and document it in risks database (RskDB).

5.2.2.4 Projects_Risks Class

*Class Name:* Projects_Risks  
*Attributes:*  
Project_Code  
Current_Risk_Code  
Inspection_No  
Risk_Probability  
Risk_Magnitude  

*Operations:*  
RiskIdentification()  
GetProjectData()  
CheckProjectCode()  
DetectRisk()  
InspectionNoDetector()  
RiskEstimation()  
PutProjectRisk()  

*Responsibilities:* Projects_Risks is designed to identify document, and track risks amongst software development projects. It uses projects' risks file (PRS) to document projects' risks data. Projects' risks data include projects specific risks data, probabilities and magnitudes, inspection numbers and others.

5.2.2.5 Risk Assessment_and_Prioritization Class

*Class Name:* Assessment_and_Prioritization  
*Attributes:*  
Risks_Probabilities_Array  
Risk_Magnitudes_Array  

*Operations:*  
AssessAndPrioritize()
**Responsibilities:** This class handles all risks assessment and prioritisation operations. Object of this class prepares a priority list of the most top ten risk items for each inspection of the specified project. *Graphic Builder* and *Controlling* classes will use the prioritised list to monitor and control top risks. Figure 5.3 clarifies how this class works.

![Diagram](image.png)

**Figure 5.3:** Risk Assessment and Prioritization

The *Graphic_Builder* class job will be starting after finishing the assessment and prioritisation task

### 5.2.2.6 Monitoring_and_Controlling Class

**Class Name:** Monitoring_and_Controlling

**Attributes:**
- Risks_Codes_Array
- RES_Array
- Inspection_Numbers_Array

**Operations:**
- MonitorAndControl()
SpecifyProject()

Responsibilities: This object of this class starts monitoring and controlling tasks.

5.2.2.7 Graphic_Builder Class

Class Name: Graphic_Builder
Attributes:
RE_Ready_Array
Inspection_Ready_Array
Operations:
LineGraph()

Responsibilities: This class accepts the prioritised risks data (i.e. at most the top ten risks of each inspection) that are prepared by the assessment and prioritisation class. The ready data will be presented by a line graph for monitoring purpose. The graph is divided into three zones: green, yellow, and red. In the graph, the x-axis represents the inspection numbers, whereas, y-axis represents RE values. Furthermore, from this class, the Controlling class can be triggered.

5.2.2.8 Controlling Class

Class Name: Controlling
Attributes:
RE-Old
Prob-New
Mag-New
RE-New
REchange
RCV
Operations:
Control()
ShowLevelsAndReductionAdvice()
ReEstimate()
Perform()
ReassessAndDisplayNewLevels()
DocumentChanges()
Responsibilities: This class allows the user to get risks reduction advice, controlling levels and all required information to control the risk. Based on the risk controlling levels, the user can decide which reduction action that must be conducted.

This class incites some other classes to perform some supporting jobs such as providing the user with a probability and magnitude estimation assistance, and introducing the reduction techniques. The class handles the documentation of new risk situation and steps that have been taken against the risk, etc. In addition, this class handles re-assessment operations, and computes the new RE, risk exposure change rate (RE_change), and risk change value rates (RCV).

5.2.2.9 Prob_Mag_Estimation_Assistance Class

Class Name: Prob_Mag_Estimation_Assistance
Attributes:
Operations:
FrameBuilder()
Responsibilities: Builds a checklist frame.

5.2.2.10 Prob_Mag_Preparation Class

Class Name: Prob_Mag_Preparation
Attributes:
Risk_Code
Risk_Name
Checklists_Array
Weights_Array
Operations:
Prob_Mag_Est Prepare ()
PrepareChecklist ()
CheckWeights ()
Put_Ready_Checklist()
Responsibilities: It is designed to prepare checklists to assist the user in estimating probabilities and magnitudes of risks. However, Risk_Documentation class invokes this class whenever the user is entering risk data. The preparation task includes preparing the checklist and weights of each element of the checklist. To make the estimation task much easier and accurate the sum of the weights should be exactly 100. This class saves the prepared checklist and weights in a special table.

5.2.2.11 Prob_Mag_Utilization Class

Class Name: Prob_Mag_Utilization
Attributes:
Risk_Code
Risk_Name
Checklist_Answers
Operations:
GetProbOrMagAssistance ()

Responsibilities: This class is invoked in any case that the user needs the estimation. The object of this class introduces a checklist for risk’s probability and magnitude estimation. The evaluation of checklist answers produces an estimation of the probability and magnitude. Data in quantitative form (for internal use) are converted into their equivalent qualitative form (for external use).

5.2.2.12 Statistical_Data_Preparation Class

Class Name: Statistical_Data_Preparation
Attributes:
Operation_Type
Buffer_Vectors
Operations:
SpecifyStatisticalOperation ()
GetProjectsRisksData()
GetRisksData()
PerformStatisticalOperation()
DisplayResults()

**Responsibilities:** The aim of this class is to prepare risks data in order to make it ready for statistics purposes. The data will be about software projects risks. This class is supposed to incite some other classes to perform this operation.

Upon performing these operations, weak and strong points can be detected (e.g. in project development progress and software risk management efficiency). Frequency of risks and similarity amongst them can be detected as well.

### 5.2.3 SoftRisk Interaction Diagrams

An interaction is defined as an order list of messages between some objects (Platinum, 1996). By using the interaction diagrams the passing makes messages easier to read (Booch, 1994; Wilkie, 1994). The interaction diagram of SoftRisk can be divided into four main scenarios as can be seen in Figures 5.4, 5.5, 5.6, and 5.7.
Figure 5.4: Interaction Diagram Scenario-1 (Risk Documentation)
SoftRisk Objects Interaction Diagram

Scenario-2: Projects' Risks Identification

Figure 5.5: Interaction Diagram Scenario-2 (Projects' Risks Identification)
SoftRisk Objects Interaction Diagram
Scenario-3: Monitoring and Controlling

Figure 5.6: Interaction Diagram Scenario-3 (Monitoring and Controlling)
Figure 5.7: Interaction Diagram Scenario-4 (Statistical Operations)
5.2.4 User Interface Design

A well screen design should reflect the needs, requirements and capabilities of its users. It should also effectively utilise the capabilities of its controlling software and achieves the business objectives of the system. However, all screen elements must have meaning to the users and serve the purpose of performing tasks. Furthermore, a visual clarity of the screen is required. It is influenced by some factors such as: consistency in design, a logical and sequential ordering, a visually pleasing composition, the presentation of the proper amount of information, groupings, and alignments of screen items (Galitz, 1997). The following are some frames design.

5.2.4.1 The Design of RskDB Data Entry/Retrieve Frame

A special frame has been designed (Figure 5.8) to enter and retrieve risks data to and from RskDB database. The frame has been designed with an ability to open some other frames to perform some special tasks. A user clicks on Risk-Code button is supposed to open another frame to handle risk-coding task. But if “Prepare Prob. Est. Checklist or Prepare Mag. Est. Checklist” is selected from the “Estimation HLP” drop-down list, another frame will be opened for preparing and updating the probability or magnitude estimating checklist.

Text fields will be used for short data items, e.g. risk name, risk description, and risk cause. For the items which need long text to describe them (e.g. mitigation plan, contingency plan, and crisis plan), the frame design includes text-area fields to allow the user to write whatever he/she wants.
The frame design contains radio buttons to detect some critical levels, for instance, the contingency plan invoking red line, crisis plan invoking red line, the highest allowed risk level, and acceptable level. The radio buttons have been selected because they enable an easy access to choices and easy comparison of captures. They are also preferred by the users (Galitz, 1997).

Managing the screen space is a very important issue. Therefore, drop-down/pop-up list boxes have been chosen in the frame design to conserve the screen space, and remind the users of the available options. Drop-down/pop-up boxes have been used to indicate risk type, and risk phase.

![Risks Database Frame Design](image)

**Figure 5.8:** RskDB Data Entry/Retrieval Frame Design
5.2.4.2 The Design of PRSF Data Entry/Retrieve Frame

A special frame is supposed to support the data entry and retrieve projects risks data to and from the PRSF. Figure 5.9 depicts the PRSF data entry/retrieve frame.

The frame includes Project Code button which opens a small frame to specify a project code, and to insure the uniqueness of the project and the correctness of the password.

![Projects' Risks File](image)

**Figure 5.9: PRSF Data Entry/Retrieve Frame Design**

More frames are supposed to be opened by clicking on some other buttons. For example:

- **Risk-Code** button specifies and invokes risk code
- *Inspection-No* button detects the inspection number of the desired risk inspection.

- *Probability-Estimation-Assistance* and *Magnitude-Estimation-Assistance* buttons are supposed to create objects of the risk Prob_Mag_Utilization class.

Radio buttons have been chosen in the design to enter probability and magnitude of risk in qualitative form. They are selected because they are suitable in such situation. Other data such as project type, size, and phase will be specified using drop-down lists.

### 5.2.4.3 Estimation Assistance Utilization Frame Design (probability side)

Figure 5.10 illustrates the design of the probability estimation assistance (utilization frame). In order to make the frame easy to use, the simplicity was considered when designing it.

![ProbHelper window](image)

**Figure 5.10:** A Probability Estimator Frame
The developer just ticks on the provided checkboxes where the checklist items are positive to the risk situation. Once the user finishes the ticking he/she will get optional estimated value.

5.3 Conclusion

In this chapter the design of SoftRisk prototype has been described. However, A Risk Matrix has been presented in this chapter, which is considered as a kernel of the prototype design.

The chapter describes the design of main classes of SoftRisk along with the some user interface design. Furthermore, the interaction diagrams are also presented. Finishing design phase means starting implementation phase. Next chapter will be on the SoftRisk implementation.