1. General Introduction

1.1 Introduction

The word “Risk” derivation was dated back to the seventeenth century, from Italian origin. Previously the common term was “hazard” (Charette, 1989). Webster’s online dictionary defines “risk” as “the possibility of loss or injury”. From the software perspective, it will be loss of revenue and injury to a company, career, and so on (Tilley & Muller, 1993). Yau (1994) defines the project risks as a loss associated with a project if the project fails to meet its objectives, weighted by the possibility of not meeting these objectives. Sherer (1995) pointed out that the risk in software development is the expected loss that may take place when software is developed, used, or maintained.

The confusion between the meaning of word “problem” and word “risk” urged Gemmer (1997) to present the following distinction between the two words. “A risk is not a problem. A risk is a potential problem over which we have some choice. If we are dealing with the consequences of a past decision or event or will have no other choice than to deal with them, it’s a problem, not a risk”.

In the computer world, the hardware was the key role at the beginning of the computer technology revolution, but the shift from hardware to software reduced its ability to act as a key role (Chittister & Haimes, 1996). This happened because of the portability, and compatibility of the software. Consequently, it implies that the door of a huge investment in software industry is opened. Shifting from the hardware to the software brought a lot of benefits but in conjunction with new problems and risks as well,
these risks were made unknown before. However, as the software have become an integrated, significant, and critical part in most aspects of our lives (e.g. finance, chemical factories, and medical life-support systems), it is increasingly becoming a risk source for those systems (Azuma, 1996; Devanbu & Fong & Stubblebine, 1998).

In many cases software failures are often life-threaten or massive loss (Bryan, & Siegel, 1988; Tran & Liu, 1997). These software risks have made software developers pay more attention to cope and manage these risks. Risk management increases the likelihood that the software products will meet their planned schedule, budget, performance, and quality (Chittister & Haimes, 1994). Otherwise the risk may cost human lives or lead to project bankruptcy (Ghezzi & Jazayri & Mondrioli, 1991). After all, risk management is routinely practised in many daily activities e.g. car driving, and engineering constructions. Nevertheless, it is a new concept in software industry (Boehm & DeMarco, 1997; Gemmer, 1997).

1.2 Software Disasters

Software projects are risky adventures. Software development projects are difficult to manage and many of them end in failure (Keil & et. al., 1998). According to the General Accounting Office or Comptroller General, only 2% of software are used as delivered, 30% after changes, 20% after extensive rework, 45% never successfully used after delivering, and 30% paid for but never delivered (Charette, 1989; Plant & Tsoumpas, 1995).
In software development there are many risk elements involved. For example key personnel resigning before the product can be fully documented, project critically depends on hardware but the manufacturer of the hardware becomes bankrupt, the hardware could not meet the software project needs, spending little or too much effort and time for testing and quality assurance, and competing package announced lower price before the project product becomes ready to be marked (Schach, 1990; Schach, 1997).

Due to the involved risks and software risk management weaknesses, software developers have to be concerned about the high frequency of software project disasters (Keil & et. al., 1998). One of software surveys illustrated that 35 percent of 600 firms had at least one software project runaway (Boehm, 1991). Another example is the C-17 transport aircraft, which is known as the most computerized aircraft. This aircraft has sophisticated software. Because of software problems its project was two years behind schedule and around $1.5 billion over its initial cost estimate $4.1 billion (GAO, 1992). In 1995, US organizations spent about $59 billion in cost overruns on IS projects and about $81 billion on canceled software projects (Johnson & Chaos, 1995).

Boehm (1988) pointed that more than 25 projects have used the Spiral model (the model is supported to include risk analysis), only the projects which fully used the model have increased their productivity about 50 percent.

1.3 Basic Concept

Boehm (1989-a) defined software risk management as "an emerging discipline whose objectives are to identify, address, and eliminate software risk items before they
become either threat to successful software operations or major source of software rework". Hence, the main goal of software risk management is to find out what may go wrong, and doing some positive things to prevent or mitigate its consequence (Charette, 1989; Hashim & Keshlaf, 1997-b).

Due to the absent or poor software risk management, many software development projects missed the chance to reach their goals of delivering acceptable software products within scheduled time, planned budget, good quality, and with high performance (Fairley, 1994; Greer & Bustard, 1997-a). It is obvious, if there is an early concern regarding identifying and managing the software risk, most of software projects’ problems would be solved or reduced before undesirable outcomes occur (Boehm, 1991; Madachy, 1994, Wiegers, 1998). Generally, optimal software risk management consists of two primary parts which are risk analysis (identify, analysis, and prioritize the risks) and risk management (plan, control, and monitor) (Boehm, 1989-a). Indeed, most of the available tools concentrate on risk analysis with some touch on the other management parts (Hashim & Keshlaf, 1997-a).

1.4 Risk Monitoring

Risk monitoring is aimed at two objectives: Firstly monitoring the risk situation itself, secondly keeping track of deviations of the controlling actions from the risk management plan as they take place in order to correct them within enough time (Boehm, 1989-a; Down & Colman & Absolon, 1994). Risk monitoring is a key role in keeping track of the risk reduction progress, hence positive and negative actions can be detected
and weak points can be avoided in the future, as well as during abrupt inspections and resolution processes (Boehm, 1991).

1.5 Software Risk Management History

Software risk management is still in its early growth stage and very slow in the adoption by software firms (Greer & Bustard, 1996; Keshlaf & Hashim, 1998). This is clear from the few numbers of related publications and approaches, which briefly covered some aspects of software risk management. Moreover, the number of the available tools is too small and not all of the existing techniques and methods are fully automated (Hashim & Keshlaf, 1997-a). In fact, U. S. National Aeronautics and Space Administration (NASA) and U.S. Department of Defense (DoD) are the earliest organizations that are committed to manage software risks. Both of the organizations have developed and introduced approaches to analyze and manage software risks. Firstly, NASA included risk management program in its flight safety procedures, where DoD in 1981, recommended spending more effort to quantify the technical risks and allocate funds to deal with. The most important approaches in the software field were established in 1985, by invoking DoD's Software Development Standard DoD-DTD-2167. Meanwhile, in the same year, the U.S Air Force started generating a number of pamphlets used for software risk analysis and management. The most important pamphlet was the one which is coded AFSC/AFLC 80045 (AirForce, 1988; Charette, 1989). Boehm introduced the Spiral model which was the first software development model that considers risks (Boehm, 1986; Boehm, 1988; Boehm & Belz, 1988; Pressman, 1997).
1.6 Motivation

The word "RISK" itself is a great motive to make software developers worry until the end of their projects. In software industry field there are many motives to establish and run software risk management program and develop managing tools (e.g. SoftRisk). Some of these motives are expanded below:

- Software risks are increasing, as long as software industry is growing (Romano & Palmer, 1998). Poor risk management makes many software development projects facing the risks without weapons. These projects fail in producing satisfied product within the planned schedule, budget, and quality (Boehm, 1989-a; Keil & et. al., 1998).

- Software risks are not threatening to the project's budget, time, and effort only, but in many critical situations they are threatening to human lives (Bryan, & Siegel, 1988; Tran & Liu, 1997).

- Running the software risk management program is not free. Only large firms can afford the cost of managing tools. Therefore, small and medium software firms are facing software risks more frequent (Bryan, 1988). So it is the time to design and develop a simple, effective, and cheap software risk management tool to match the needs of all software projects' types and sizes (Hashim & Keshlaf, 1997-a; Hashim & Keshlaf, 1997-b).

- Available software risk management tools are too few. They are designed for specific aspects and purposes (e.g. designed for requirement phase, or
covers risk analysis only) which limit them for specific applications only (Keshlaf & Hashim, 1998).

- Software risks are changeable based on the software industry development. Discovering new risks and enhancing in the risk assessment formulas are vital to enhance the risk management techniques.

1.7 Benefits of Risk Management

Running software risk management brings many benefits to software development projects. Hereafter, some of these benefit (Charette, 1989; Boehm, 1989-a; Boehm & DeMarco, 1997):

- Software risk management increases the chance of meeting project goals, in delivering the product within the planned schedule, cost, and quality.
- Risk management maximizes the organization profit, reputation, and saves development time. Furthermore, it minimizes development problems, and programs rework.
- Software risk management increases the reliability of software development plans, and makes decision-makers more confident in taking decisions regarding their software projects.
• Efficient risk management technique can identify risks very early and then guide the developer to the shortest way of dealing with them within enough time, and lower cost.

• Protecting other systems which embeds the software from indirect software risks.

1.8 Project Aims

This project has been conducted in order to improve and tackle the shortages of the existing tools. The main aims of the project are listed below:

• Attempting to detect some new top risks that have high impacts on software industry.

• Increase the efficiency of the software risk management, by simplifying and improving the software risk management techniques.

• Build a prototype tool to manage the software risks in order to cover some risk management aspects (e.g. risk monitoring) that have not been covered well by existing approaches.

• Risk probability and magnitude are difficult to estimate. Therefore, the research is supposed to establish a mechanism to assist software developers in estimating the risks' probabilities and magnitudes.
• Due to the importance of risk documentation in predicting the future risks, the prototype is supposed to give some attention to the risk documentation.

1.9 Methodology

The methodology of achieving the research aims can be summarized in the following points:

• Understand and study existing software risk management approaches in order to determine the negative and positive aspects.

• Conduct a software risk management survey to know the practicing level of the software risk management in Malaysia. The survey’s result could be used to improve the proposed prototype design.

• Evaluating the existing software risk management tools in order to capture their good features and tackle their shortcomings in building the proposed prototype.

• Practicing the traditional software development steps to develop the proposed prototype.

• Finally, testing and validating the prototype.

1.10 Thesis Overview

This thesis covers most of the software risk management aspects by concentrating on the software risk monitoring, which has not been covered well in the literature. The thesis
consists of eight chapters including this introduction chapter. The next chapter will touch on theoretical background of the software risk management along with a review of the existing risk management approaches. As a continuity of the literature review, a software risk management survey was conducted in Malaysia. One of survey’s aims is to evaluate the existing top ten risk items and an additional seven top risk items that have proposed in this study. The survey’s result and its analysis will be discussed in chapter 3. Actually, developing any software tool should follow a special development lifecycle, which starts with a requirement phase and ends with a retirement phase. Therefore, the development of this research’s prototype (SoftRisk) started with the requirement specification. Hence, the SoftRisk analysis and requirements identification is described in chapter 4. As well as the components of a mechanism to estimate probabilities and magnitudes of risks is described in the same chapter. Chapter 5 clarifies design of SoftRisk (i.e. its main classes, interaction diagrams and user interface design). The next chapter (Chapter 6) describes SoftRisk development, e.g. the JAVA programming language used, diagrams, algorithms, classes, implementation problems and others. Chapter 7 discusses testing and validation of SoftRisk, and indirectly shows some shots of SoftRisk frames. Finally, chapter 8 concludes the thesis, states the main contribution, and gives future directions.