CSIT (September 2013) 1(3):243–252
DOI 10.1007/s40012-013-0022-4

ORIGINAL RESEARCH

A systematic approach for the estimation of software risk and cost using esrcTool

Mohd Sadiq · Mohd Shahid

Received: 27 November 2012 / Accepted: 29 June 2013 / Published online: 13 August 2013

Abstract Software risk evaluation is a process for identifying, analysing, and developing mitigation strategies for risk in a software intensive system while it is in development. This paper presents a systematic approach for the estimation of software risk and cost using esrcTool. This tool is based on software risk assessment and estimation model. In this model function point approach is employed as an input variable to estimate the source of uncertainty, i.e. measurement error, model error, and assumption error. To show the validity of our tool, we have considered the project developed by our students.

Keywords Software risk · Cost · esrcTool · FP · SRAEM

1 Introduction

Software risk evaluation (SRE) is a process for identifying, analyzing, and developing mitigation strategies for risks in a software-intensive system while it is in development. SRE process has been in evolutionary development at the Software Engineering Institute (SEI) since 1992; and it has been used on over 50 Department of Defence and civil (Federal and State) contractors and program offices. There are certain questions that are related to risk management like risk prioritization and analysis etc. It is systematic and continuous processes that can be best described by the SEI risk management paradigms which includes: (a) identify (b) analyze (c) plan (d) track (e) control and (f) communication [1]. The practice of risk management involves two primary steps each with three subsidiary steps, as shown in Fig. 1. The first primary step of risk management is risk assessment and it involves: (a) risk identification, (b) risk analysis, (c) and risk prioritization. The secondary step of risk management includes risk control and it involves: (a) risk management planning, (b) risk resolution, (c) and risk monitoring [2–4].

Risk assessment begins with team training. The team meets prior to the risk assessment for team building and training in the details of risk management paradigm and risk assessment process. This training includes instruction in the risk management mechanism to be applied during the assessment as well as practice exercises using the mechanism [5, 6]. Brief description about primary and secondary steps of risk management is given below:

1. Risk identification produces lists of the projects-specific risk items likely to compromise a project success. A typical risk identification technique includes examination of decision drivers, assumption analysis, and checklist.
2. Risk analysis assesses the loss probability and loss magnitude for each identified risk item and it access compound risk in the risk item interactions. Typical technique include performance models, cost models, network analysis etc.
3. Risk prioritization produces a ranked ordering of the risk items identified and analyzed. Typical techniques includes risk exposure analysis, risk reduction leverage (particularly involve cost-benefit analysis) etc.

M. Sadiq (✉)
Department of Computer Engineering, National Institute of Technology, Kurukshetra 136119, Haryana, India
e-mail: sadiq.jmi@gmail.com; msq_delhi@yahoo.co.in

M. Shahid
Department of Computer Science and Engineering, Mewat Engineering College, Established by Haryana Wakf Board, Government of Haryana, Gurgaon, Haryana, India
e-mail: shahid27.jmi@gmail.com

Fig. 1 Steps involved in the risk management

4. Risk management planning helps prepare you to address each risk item including the coordination of individual risk item plans with each other and with the overall project plan. Typical technique includes checklist for risk resolution technique, cost benefit analysis, and standard risk management plan outlines, form and elements.
5. Risk resolution produces a situation in which the risk items are eliminated or otherwise resolved (for example risk avoidance via relaxation of requirements). Typical technique includes prototypes simulation, benchmark, mission analysis, and design to cost approach.
6. Risk monitoring involves tracking the project's progress toward resolving its risk items and taking corrective action where appropriate. Typical technique includes milestone tracking and a top ten risk item list that is highlighted at each weekly and monthly [7].

The objective of this paper is to present a systematic approach to estimate the software risk and cost using esrcTool. This tool is based on software risk assessment and estimation model (SRAEM) [8] which is used to predict the possible results of software projects with good accuracy. SRAEM not only assess the risk but it also estimates the risk. In SRAEM, function point (FP) is used to estimate the measurement error, model error, and assumption error. Function point is an important software metrics which is used to calculate the approximate LOC, cost and effort of software [5, 9, 10].

The paper is organized as follows: In Sect. 2 we present the background and the related work of risk assessment and estimation model. Section 3 presents the method for the computation of FP. In Sect. 4, we present the systematic approach for the estimation of software risk and cost using esrcTool. An experimental work is carried out in Sect. 5; and finally we conclude the paper in Sect. 6.

2 Background and related work

This section presents the background and related work of software risk assessment and estimation models. Researchers, scientist and academician in the field of software engineering have developed a lot of models/tools according to their need and requirements. Gupta and Sadiq [8] have developed a SRAEM to predict the possible results of software projects with good accuracy. SRAEM model not only assess the risk but it also estimate the risk. In this model, the risk is estimated using risk exposure and software metrics of risk management. This metric is based on mission critical requirements stability risk metrics (MCRSRM) [11]. This software metrics is used when there are changes in requirements such as addition, subtraction, or deletion. The esrcTool gives the incremental risk for every phase and also the total cumulative risk as the software progress from phase to phase.

Gupta et al. [12] developed a software estimation tool based on software engineering metrics model. In literature, there are a few published models that evaluate the risk of software projects. In [13] Kashlaf and Hashim developed a model and prototype tool to manage software risks. Soft risk prototype is a tool prototype to manage software development risks. Java language has been chosen as development language of the prototype. Another model named Software engineering risk model (SERIM) focuses on three risk elements: (i) technical risk, (ii) cost risk, and (iii) schedule risk. This model does not take into account of the software complexity issues, which plays an important role in determining the risk for the software projects. It also does not account for issues related to requirements. In the series of risk assessment models there is another model called software risk assessment model (SRAM) [14]. This model considers the nine critical risk elements (i) complexity of the software (ii) staff involved in the projects (iii) targeted reliability (iv) product requirement (v) method of estimation (vi) method of monitoring (vii) development process adopted (viii) usability of software (ix) tools. The above models do not include the sources of uncertainty, i.e. measurement error, model error and assumption errors. Existing models have considered the prioritization as a single step of risk assessment but does not specify how prioritization would be done. In [15] we proposed the architecture of an esrcTool to estimate the software risk and cost. On the basis of systematic review of software risk assessment and estimation model [16], which is an

extension of the work of Georgieve et al. [17], we identify 11 primary studies that are related to software risk and cost estimation and its brief description is given in Table 1. The results of previous studies indicate that *SRAEM* [8] and *SRAEP using model based approach* [18] is the latest risk assessment model in the literature [16, 17].

3 Method for the computation of FP

At the beginning of the 1970s, researchers at IBM initiated studies aimed at determining what critical variables were involved in programming productivity. Instead of considering code volume or complexity, they discovered that a system would be better evaluated by analyzing the functions executed by programs and mapping pertinent questions to estimating and evaluating software's development productivity in heterogeneous environments [19]. Function Point is a well known established method to estimate the size of software system and software projects. Originally, the method was used in the early phases of the waterfall model such that the implementation effort could be estimated on the basis of input and output behaviour as defined in the functional documentation.

As the size and the complexity of software increases, it becomes increasingly important to develop high quality software and cost effectively within a specified period. In order to achieve this goal, the entire software development processes need to be managed based on an effective plan. The subjects of estimation in the area of software development are size, effort invested, development time, technology used and quality.

Function Point is a measure of software size that uses logical functional terms business owners and users more readily understand. Albrecht's model of functional specifications requires the identification of five types of components, namely input, output and inquiry elementary processes and logical internal elementary processes and logical internal and external interface files The actual calculation process itself is accomplished in three stages: (I) determine the unadjusted function point (UFP), (II) value adjustment factor (VAF) and (III) Adjusted function points(AFP).The unadjusted function points includes: (a) Data function and (b) Transactional functions.

The data function includes:

3.1 Internal logical file
3.2 External interface file.

The transactional functions are classified in the following manner:

3.3 External input.
3.4 External output.
3.5 External enquiry.

3.1 Internal logical file (ILF)

ILF are groups of logically related data or control information maintained by the application itself. For instance, the file that stores the customer data of a company is and ILF for the customer database system since such a system is responsible for customer maintenance.

3.2 External interface file (EIF)

EIF are groups of logically related data or control information whose maintenance is under the responsibility of another application. For instance, the file that stores employee data of a company is an EIF for the customer database system, assuming that it accesses employee data whose actual maintenance is accomplished by the employee database system.

3.3 External input (EI)

EI are elementary processes that involve data or control information that are input at the boundary of the application with the main objective of doing ILF maintenance, for instance, updating the personal data of customer of an organization.

3.4 External output (EO)

EO are elementary processes that send control information or calculated data to the end user or to other applications, for instance, a report that summarizes sales volume per month for a particular customer of a company.

3.5 External inquiry (EQ)

EQ are elementary processes that send control information or uncalculated data to the end user or to other applications. For instance, a personal data query operation for an employee of a company

Once we have identified EI, EO, EQ, ILF, and EIF then each function presented must be classified according to its relative functional complexity as low, average or high. Calculating the VAF, is an earmark of the general functionality provided to the user. The VAF is derived from the sum of the degree of influence (DI) of the 14 general system characteristics. The DI of each one of these characteristics ranges from 0 to 5 as follows:

(i) 0—no influence;
(ii) 1—incidental influence;
(iii) 2—moderate influence;

Table 1 Summary of various risk assessment methods [16]

S. no.	Year of publication	Method's name	Input data	Risk assessment technology	Type of evaluation		
					Qualitative	Quantitative	Hybrid
1	Sadiq et al. [18]	SRAEP using Model based approach	Identify context using use case diagram, sequence diagram and security requirements	Calculate risk exposure and compute degradation of key node safety metric		√	
2	Gupta and Sadiq [8], 2008	SRAEM	Measurement error, model error, and assumption error	Risk exposure and mission critical requirements stability risk metrics		√	
3	Young et al. [28]	Analysing Software System Quality Risk using Bayesian Belief Network	Project risk factors	Bayesian belief network, Delphi method	√		
4	Vucovich et al. [29]	Software Risk in Early Design Method	Software functionality, historical function failure	Function failure design method		√	
5	Yong et al. [30]	A Neural network Method for Software Risk Analysis	Software Risk Factors from Questionnaires	Principal component analysis, genetic algorithm and NN			√
6	Deursen and Kuipers [31]	Source based software risk assessment	Source code information	Code metrics, and Questionnaires		√	
7	Yacoub and Ammar [32]	A methodology for architecture level reliability risk analysis	Complexity and coupling metrics	Dynamic metrics and architecture elements		√	
8	Neumann [33]	An enhanced neural network technique for software risk analysis	Software metrics data	Principal component analysis and artificial neural network		√	
9	Nogueira et al. [34]	A Risk assessment model for software prototyping	Requirements, personnel and complexity metrics	Different software metrics		√	
10	Williams et al. [35]	Software risk evaluation from SEI risk management paradigm	Risk data	Questionnaires	√		
11	Chee et al. [36]	Influence diagram for software risk analysis	Software metrics data	Influence diagram		√	

(iv) 3—average influence;
(v) 4—significant influence; and
(vi) 5—strong influence.

The general characteristics (F_i) of a system are: (i) data communications; (ii) distributed data processing; (iii) performance; (iv) heavily used configuration; (v) transaction rate;(vi) online data entry; (vii) end user efficiency (viii) online update(ix) complex processing; (x) reusability; (xi) installation ease; (xii) operational ease; (xiii) multiple sites; (xiv) facilitate change. The third and the last stage is the final calculation of the function points. With the help of the following equation we can get the total points of an application.

$$AFP = UFP \times VAF$$

where AFP adjusted function points; UFP unadjusted function points; and VAF value adjustment factor [20, 21].

Function points are computed by completing the following Table 2. Five information domain characteristics are determined and counts are provided in appropriate table location [22].

Organization that use FP methods develop criteria for determining whether a particular entity is simple, average, or complex. To compute the AFP the following relationship is used.

$$AFP = UFP \times \left[0.65 + 0.01 \times \sum (F_i)\right] \quad (1)$$

where F_i is the complexity adjustment value, i = 1–14.

For example, it is given that EI = 5, EO = 2, EQ = 4, ILF = 2 and EIF = 1 and the complexity of the project is average then the UFP would be calculated as follows (see Table 3).

If we assume that all general systematic characteristics are absolutely essential, then FP in this case would be 98.55

4 Systematic approach

This section presents a systematic approach for the estimation of software risk and cost using esrcTool 1 [15, 23].

Table 3 Results

MP	Count	Average complexity	Results
EI	5	4	20
EO	2	5	10
EQ	4	4	16
ILF	2	10	20
EIF	1	7	7
UFP	73		

Systematic means to characterize by order and planning. Therefore, esrcTool first extract the source code of the program/software for the computation of the FP because it is used as an input to the measurements error, model error, and assumption error. In order to estimate the risk and cost of the software, we have implemented this tool in C language. The architecture of the esrcTool is given in Fig. 2.

The systematic approach for the estimation of software risk and cost using esrcTool involves the following steps: (a) estimation of the risk, (b) cost estimation. The detailed descriptions about these steps are given in the following sub-section.

4.1 Estimation of the risk

There are three dimensions of software risk i.e. technical risk, organization and environmental risk [16]. Each software models have some weaknesses and also have some advantages [22]. A technical report on the software risk evaluation method is available in [1]. The esrcTool estimates the risk on the basis of measurement error, model error, and assumption error.

4.1.1 Measurement error

This error occurs if some of the input variables in a model have inherent accuracy limitations. Kemerer [24] argue that, function points are assumed to be at least 12 % inaccurate. Thus, if we estimate a product size of 1,000

Table 2 FP computation

Measurement parameter (MP)	Count	Simple	Average	Complex	Result
EI	X	3	4	6	Results after simplification
EO	X	4	5	7	
EQ	X	3	4	6	
ILF	X	7	10	15	
EIF	X	5	7	10	
UFP					

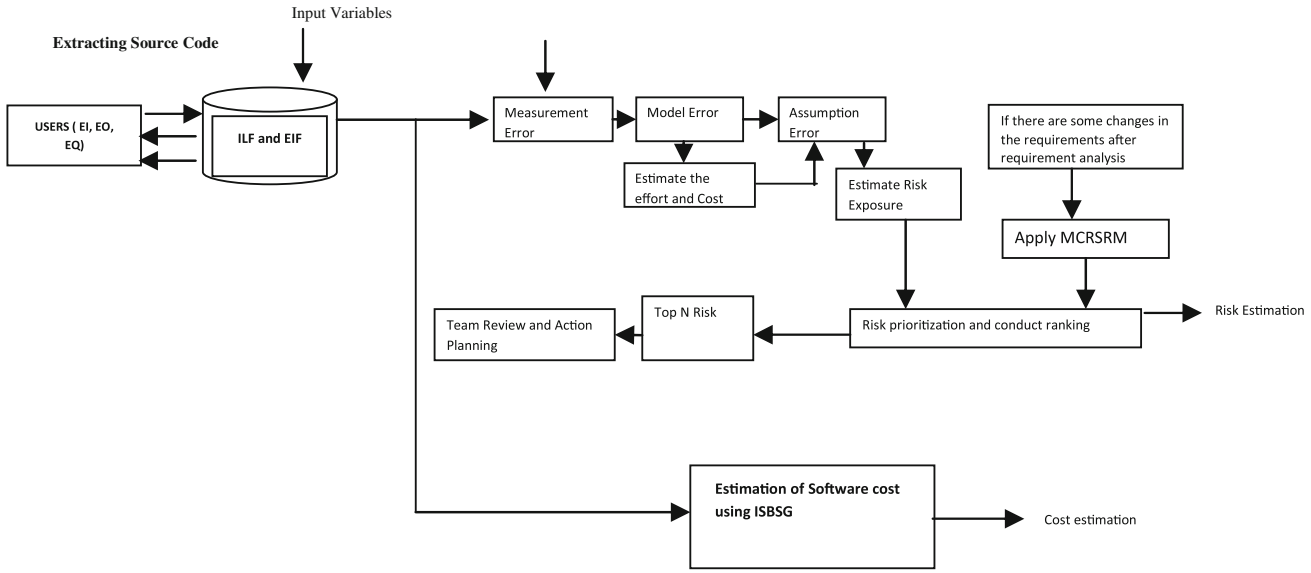

Fig. 2 Architecture of esrcTool (adopted from [23])

function points, measurement error could mean that the real size is anywhere between 880 and 1,120.

4.1.2 Model error

Factors that affect error but are not included explicitly in the model contribute to the model error. For example, 0.5 person-days per function point is usually obtained from results observed for recalled from previous projects. It is unlikely that any future projects will achieve the same ratio, but the model is expected to all right on average. If you base a model on past project data, you should calculate the associated inaccuracy by using the mean magnitude relative error. Thus, if you have estimation model with an inherent 20 % inaccuracy and your product is 1,300 function points in size; your estimate is likely to be between 208 and 312 person days.

4.1.3 Assumption error

This error occurs when we make incorrect assumptions about a model's input parameters. For example, your assessment that a product size is 1,300 function point rests on the assumption that you have correctly identified the customer requirements. If you can identify your assumptions, you can investigate the effect of their being invalid by assessing both the probability that an assumption is incorrect and the resulting impact on the estimate.

This is the form of risk analysis. For example, if you believe that there is a 0.3 probability that the requirement complexity has been underestimated and, if it has, you estimate another 390 function point. At this point the concept of risk exposure is used to calculate the effective current cost of a risk and can be used to prioritize risk that requires countermeasure. Mathematically it can be written as:

$$\text{Probability of risk occurring} \times \text{Total loss if risk occur} \quad (2)$$

total loss can be defined as E2 − E1. Where E1 is the effort, if the original assumption is true and E2 is the effort if the alternative assumption is true. Suppose E1 = 540 person days and E2 = (1,300 + 100) × 0.5 = 700 person days, then risk exposure = (700 − 540) × 0.3 = 48 person days.

The esrcTool also includes the MCRSRM in order to compute the risk when there are some changes in the requirements (addition, modification, or deletion). Therefore, total risk can be computed as [23]:

$$[b/a]_{i=1 \text{ to } n} + K\left[A[c/d]_i + B[d/b]_i + G[e/b]_i\right] \quad (3)$$

where:

(i) $[b/a]_i$ = (number of mission critical requirements)/(total number of requirements) at the input of phase number i,
(ii) K_i is the penalty for adding, modifying or deleting of requirements during phase number i,
(iii) a = total number of requirements,
(iv) b = total number of mission critical requirements (MCR),
(v) c = number of MCR added during phase i,
(vi) d = number of MCR modified during phase i,
(vii) e = number of MCR deleted during phase i.

Table 4 Information about probability of occurring and total loss, if it occurs

Risk	Probability of occurring (%)	Total loss, if it occurs (K)
Product recall situation	2	80
Significant product rejection	0.1	1,000
Competitive strike	10	25

Table 5 Prioritization of risk

Risk	Probability of occurring (%)	Total loss, if it occurs (K)	Risk exposure	Risk priority
Product recall situation	2	80	1,600	2
Significant product rejection	0.1	1,000	1,000	3
Competitive Strike	10	25	2,500	1

On the basis of the following parameters during phase 1, the risk at the input is 40/100 and the added risk during phase 1 is 42/40 due to adding, modifying and deletion of MCR. Where, a = 100, b = 40, c = 5, d = number of MCR modified during phase 1 = 10 (3 which were downgraded from MCR to just requirements, and the remaining 7 are still MCR, and e = 7.

It can be seen that esrcTool gives the incremental risk for every phase and also the total cumulative risk as the project progresses from phase to phase. Assumption error helps you to estimate the risk exposure. Now the next step is how to prioritize the risk: Suppose in a software project, we identified three different types of risk i.e. products recall situation, significant product rejection, and competitive strike. The information about probability of risks occurring and the total loss if it occurs are given in the Table 4.

The rank of risk is estimated using risk exposure and the value of the highest risk exposure indicate the most serious risk. Table 5 contains the calculated values of risk exposure and the ranking of risk.

In Table 5, Competitive strike contains the highest value of risk exposure i.e. 2,500, so it has the first priority. Similarly, product recall situation and significant product rejection have second and third priority respectively. Since risk exposure is not absolute but relative. There are several ways to compare different exposures with each other. One ways is to compare the exposure of a single event before and after managing the risk. We need a simple measure to assess risk reduction. So risk reduction leverage (RRL) is another quantitative means of assessing how risks are being managed. Mathematically we can write the RRL as:

$$(\text{Risk exposure before} - \text{Risk exposure after}) / \text{Cost of risk reduction} \quad (4)$$

4.2 Cost estimation

International Software Benchmarking Standards Group (ISBSG) is an international group of representatives from international metrics organizations who collect project data from countries like, India, Hong Kong Germany, Japan, and USA. ISBSG Release 6 Report provides the cost value for the software projects. Cost data is derived from 56 projects representing a broad cross section of the software industry. After going through these software projects, the ISBSG conclude that median cost to develop a function point is $US 716, and the average cost is $ US 849 per function point. For more information about the ISBSG please visit: www.ISBSG.org.au

5 Experimental work

This section presents the experimental work. In this paper, we have considered the projects developed by the students of Master of Technology (M.Tech.) of Computer Science and Engineering. The first project that we have considered is "A Mini Software for Numerical Integration (MSNI)". This Software basically calculates the values of the given Integrand after applying all the existing algorithms of the numerical computations like Simpson's 1/3 rule, Simpson's 3/8 rule and so on; and some proposed algorithm. Software requirements elicitation is a process which is used to identify the high level objective of an organization [25–27]. In MSNI, we have collected 21 different requirements and the prioritization of these requirements is shown in Fig. 3. For the computation of software risk and cost, we implement esrcTool in C language; and the GUI of the proposed tool is given in Fig. 4.

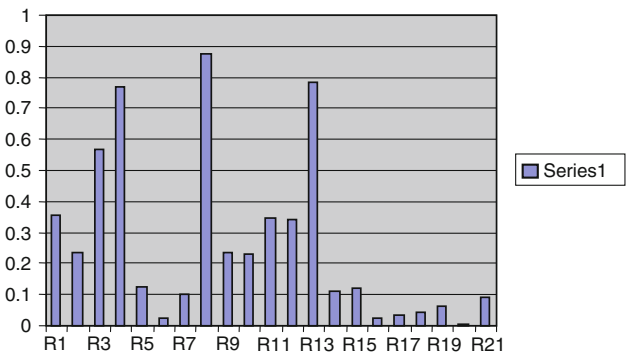

Fig. 3 Prioritization of requirements

Fig. 4 Snapshot of the GUI of the esrcTool [23]

Table 6 Results from esrcTool

Project (P) number	Values of FP	Measurement error	Model error	Assumption error	Risk exposure	Cost
P1	1,300	1,144 FP–1,456 FP	208–312 person days	Addition of 52 FP	10 Person day	1,144 × cost of 1 FP to 1,456 × cost of 1 FP
P2	1,200	1,056 FP–1,344 FP	192–288 person days	Addition of 48 FP	9 Person day	1,056 × cost of 1 FP to 1,344 × cost of 1 FP
P3	1,000	880 FP–1,120 FP	160–240 person days	Addition of 40 FP	8 Person day	880 × cost of 1 FP to 1,120 × cost of 1 FP
P4	750	660 FP−840 FP	120–180 person days	Addition of 30 FP	6 Person day	660 × cost of 1 FP to 840 × cost of 1 FP
P5	600	520 FP−672 FP	96–144 person days	Addition of 24 FP	5 Person day	520 × cost of 1 FP to 672 × cost of 1 FP
P6	550	484 FP−616 FP	88–132 person days	Addition of 22 FP	4 Person day	484 × cost of 1 FP to 616 × cost of 1 FP
P7	400	352 FP−448 FP	64–96 person days	Addition of 16 FP	3 Person day	352 × cost of 1 FP to 448 × cost of 1 FP
P8	250	220 FP−280 FP	40–60 person days	Addition of 10 FP	2 Person day	220 × cost of 1 FP to 280 × cost of 1 FP
P9	180	158 FP−202 FP	29–43 person days	Addition of 7 FP	1 Person day	158 × cost of 1 FP to 202 × cost of 1 FP
P10	90	79 FP−101 FP	11–25 person days	Addition of 4 FP	1 Person day	79 × cost of 1 FP to 101 × cost of 1 FP

In order to estimate the function point of the MSNI, we used the esrcTool as well as the tool proposed by [12]. The value of function point of MSNI found to be 50. According to the measurement error the actual size of the function point varies from 44 to 56. To find out the value of model error, we have assumed that the 0.2 person-days per function point. No estimation model can include all factors that affect the effort required to produce a software product. Suppose, we have an estimation model with an inherent 20 % inaccuracy and our product is of 50 function points in size, our estimation is likely to be between 8 and 12 person days.

The assumption error occurs when we have some incorrect assumption about the models input parameter. Our assumption is that the product size is of 50 function points rest on the assumption that we have correctly identified all the requirements. If we can identify our assumption, we can investigate the effect of their invalid by assessing both the probability hat an assumption is incorrect and the resulting impact on the estimate. This is the called the risk analysis. If we assume that there are 0.4 probabilities that the requirement complexity has been underestimated, so we estimate another 2 function point. We can estimate the risk exposure from the following

formula: Risk Exposure = E2 − E1, where E1 is the effort if the original assumption is true, and E2 is the effort if the alternative assumption is true and P2 is the probability that the alternative assumption is true. So in our case E1 = 10 person days, E2 = 50 × 0.2 + 2 × 0.2 = 10.4 = 11 (approximately). The risk exposure = 11 − 10 = 1 person days. We have summarized the results of esrcTool in Table 6.

6 Conclusion and future work

In this paper we present a systematic approach to estimate the software risk and cost using esrcTool. We have implemented esrcTool using C language. From the proposed tool, it is easy to estimate the risk in the software and also to estimate the cost of the software. The cost of the software depends on the value of the function point. In this paper we have employed the function point approach as an input parameter into the esrcTool. We have applied to the proposed tool on the MSNI; and on nine other projects that are based on software engineering and computer graphics. From the proposed tool, it is easy to find out the cost of software; and estimate the risk of those software's projects that were designed and developed by our graduate and post graduate students. Software risk and cost analysis is the part of non functional requirements (NFR). Therefore, such type of analysis helps to improve the understanding level of stakeholders during retirements elicitation phase. Future research agenda includes the following:

(i) To apply the esrcTool on real projects because projects developed by students have some kind of errors.
(ii) To develop a fuzzy based model for the computation of esrcTool.
(iii) To elicit and prioritize the software requirements using analytic hierarchy process (AHP) and quality function deployment.

References

1. Williams RC, Pandelios GJ, Behrens SG (1999) Software risk evaluation (SRE) method description (Version-2.0), Technical report December-1999
2. Sherer SA (2005) The three dimensions of software risk: technical, organizational, and environmental. In: 28th Hawaii international conference on system sciences, 1995, Wailea, IEEE
3. Van Scoy RL (1992) Software development risk: opportunity, not problem, Technical report
4. Demarco T, Lister T (2003) Risk management during requirements, IEEE Computer society, IEEE Software, pp. 99–100
5. International Function Point User Group (IFPUG) (1990) Function point counting practices manual, Release 4.0, IFPUG, Westerville
6. Zuse H (1991) Software metrics-methods to investigate and evaluate software complexity measures. In: Proc. second annual oregon workshop on software metrics, Portland
7. Firesmith D (2004) Prioritizing requirements. J Object Technol 3(8):35–47
8. Gupta D, Sadiq M (2008) Software risk assessment and estimation model. In: International conference on computer science and information technology, IEEE Computer Society, Singapore, pp 963–967
9. Low GC, Jeffery DR (1990) Function point in the estimation and evaluation of the software process, IEEE Trans Software Eng 16(1)
10. Tsoi H-L (2005) To evaluate the function point analysis: a case study. Int J Comput Internet Manag 13(1):31–40
11. Sherif JS (1996) Metrics for software risk management, ISMN#0-7803-3274-1, pp. 507–513
12. Gupta D, Kaushal SJ, Sadiq M (2008) Software estimation tool based on software engineering metrics model. In: IEEE international conference on management of innovation and technology, Bangkok, pp. 623–628
13. Keshlaf AA, Hashim K (2000) A model and prototype tool to manage software risks. In: 1st Asia pacific conference on software quality, pp. 297–305, IEEE
14. Foo S-W, Muruganatham A (2000) Software risk assessment model ICMIT 2000, IEEE, pp. 536–544
15. Sadiq M, Rahman A, Ahmad S, Asim M, Ahmad J (2010) esrcTool: a tool to estimate the software risk and cost. In: IEEE second international conference on computer research and development, pp. 886–890, Kuala Lumpur. doi: 10.1109/ICCRD.2010.29
16. Ahmad Khan MA, Khan S, Sadiq M (2012) Systematic review of software risk assessment and estimation models. Int J Eng Adv Technol 1(4), ISSN: 2249–8958
17. Georgieva K. Et al (2009) Analysis of risk analysis methods—a survey, LNCS- Springer, Heidelberg, pp. 76–86
18. Sadiq M, Ahmad MW, Rahmani MKI, Jung S (2010) Software risk assessment and evaluation process (SRAEP) using model based approach. In: IEEE international conference on networking and information technology, ICNIT-2010, pp. 171–177, Manila
19. Albrecht AJ (1979) Measuring application development productivity, Proc. IBM applications development symposium, Monterey. pp. 14–17
20. Low GC, Jeffery DR (1990) Function point in the estimation and evaluation of the software process, IEEE Trans Software Eng 16(1)
21. International Function Point User Group (IFPUG) (1990) Function point counting practices manual, Release 4.0, IFPUG, Westerville
22. Sadiq M, Rizvi DR, Aggarwal S (1997–2001) Weaknesses of software risk estimation models. In: 2nd National conference on emerging trends in computer science and information technology, AFSET, Faridabad, pp. 146–150
23. Sadiq M, Sunil, Zafar S, Asim M, Suman R (2010) GUI of esrcTool: a tool to estimate the software risk and cost. In: The 2nd IEEE international conference on computer and automation engineering (ICCAE-2010), pp. 673-677, Singapore
24. Kermerer CF (1993) Reliability of function points measurements, a field experiment. Commun ACM 36:85–97
25. Li Z, Wang Z, Yang Y, Wu Y, Liu Y (2007) Towards multiple ontology framework for requirements elicitation and reuse. In: 31st IEEE annual international computer software and application conference
26. Sadiq M, Ghafir S, Shahid M (2009) An approach for eliciting software requirements and its prioritization using analytic hierarchy process. In: IEEE international conference on advances in recent technologies in communication and computing, ACEEE annual world congress on engineering and technology
27. Sadiq M, Ghafir S, Shahid M (2009) A framework to prioritize the software requirements using quality function deployment. In:

National conference on recent development in computing and its application, organized by Jamia Hamdard, Delhi
28. H. Young et al (2007) Analysing software system quality risk using bayesian belief network. In: Proceedings of the international conference on granular computing, IEEE computer society, Los Alamitos
29. Vucovich JP et al (1995) Risk assessment in early software design based on the software function failure design method. In: Proceedings of the 31st annual international conference on computer software and applications. IEEE Computer Society, Los Alamitos
30. Yong H et al (2006) A neural network approach for software risk analysis. In: Proceedings of the 6th international conference on data mining-workshop. IEEE Computer Society, Los Alamitos
31. Deursen T et al (2003) Source based software risk assessment. In: Proceedings of the international conference on software maintenance, IEEE Computer Society, Los Alamitos
32. Yacoub SM, Ammar HH (2002) A methodology for architecture level reliability risk analysis, IEEE Trans Software, pp. 529–547
33. Neumann DE (2002) An enhanced neural network technique for software risk analysis. In: IEEE transaction on Software Engineering, pp. 904–912
34. Nogueira J et al (2000) A risk assessment model for software prototyping projects. In: Proceedings of the 11th international workshop on rapid system prototyping. IEEE Computer Society, Los Alamitos
35. Williams RC et al (1999), Software risk evaluation method description, CMU/SEI-99-TR, ESC-TR-99-029, Software Institute
36. Chee CL et al (1995) Using influence diagram for software risk analysis. In: Proceedings of the 7th international conference on tools with artificial intelligence. IEEE Computer Society, Los Alamitos