A Qualitative Approach for Software Project Risk Identification by KeyGraph

Gufran Ahmad, Daxin Liu

College of Computer Science
Harbin Engineering University
gufran.ahmad@hrbeu.edu.cn

***Abstract*— Software project risk identification is a crucial step in the process of software risk management. Software risk identification is significant not only for research intent but for industrial intent also. The current work is a successful endeavour to solve the software project risks identification problem with KeyGraph, a graphical tool of chance discovery. The procedure simplifies the software risks identification. We apply KeyGraph method in a realistic software project scenario to discover newly generated hubs which are of prime importance in the identification of software project risks that are associated with the software project. After a careful analysis of the software project scenario, we identify these significant hubs as software project risks within the concerned software project.**

***Keywords*— Software project risk, KeyGraph, Software Project, Project Management**

I. INTRODUCTION

Software project risk identification is an initial and integral process in every software risk assessment which is a part of software risk management. Without proper software risk identification, it is tough to control and manage the risk that is associated with the software project [1], [2].

So far, a number of literatures and articles [1-5] are contributed to this effort and a vast majority of techniques have been applied for the concerned problem to solve the issue but still most of them are basically, dependent on rigorous textual procedures to identify risks [4].

The first thing to note that risk management entails additional cost. However, risk management can be considered cost-effective only if the cost of risk management is considerably less than the loss incurred if the risk materializes. Actually, the cost of risk management should be less than the expected value of the loss. Therefore, it is again obligatory to identify software project risk efficiently [6].

In addition, the significance of software project risks has become so dominating in all software development areas that some recent trends have centred its entire efforts on software risks. One of the recent trends is based on project's particular risks with respect to the use of agile or plan-driven methods in software development. It is known as risk-based method or risk-driven method software development. It is considered as hybrid agile method and is applied to software projects to balance discipline and agility and it can craft big extraordinary change based on project risk patterns. Henceforth, it simplifies all the software development and software project processes and adds a value to software project management as well [3].

II. RESEARCH OBJECTIVE

The main objective of this research is to identify the software project risks which are always associated with every software project by means of KeyGraph method.

We have classified these software risks as per the technical report of CMU/SEI by Marvin J. Carr, et al [7].

III. RESEARCH METHOD

The software risk identification process begins with the creation of software project risk scenarios. These scenarios are at first, made by collecting essential set of data or nodes as input to the formation of KeyGraph [8].

Further, co-occurrence network is created by links between these frequent co-occurring item-pairs or node-pairs of the data-set. The fundamental islands are obtained by deleting links that separates the graphs and deleting the links of highest contribution. This situation is regarded as generation of hubs, candidates of rare event or chance, indeed, risks which are significant items, touching many node-cluster bridges of frequent co-occurrence [9], [10].

Finally, a deliberate analysis is done before the nomination of these risks. These risks are nominated with the aid of software risk categorization.

IV. RESEARCH DATA-SET COLLECTION

Research data-set is collected from one of the software project scenarios cited in Anne Fuller, et al [5] in their case studies. The cited case is mentioned as follows.

"A company has developed a very comprehensive software product for a particular target industry. However, most of the users in the industry cannot afford the costs of deploying the product. The software is too complex for simple users to handle, and potential clients must dedicate a team of people with knowledge of the industry and advanced computer skills if they are to successfully adopt the package.

Thus the developers need to scale down the original product in order to increase market share.

They are considering two options: to block out some modules in the original system and thus disable some functionality, or to write another small system from scratch, reusing some code from the old system."

On the basis of this case, we create software risk scenario with KeyGraph to identify the software project risks that may occur during such changes by applying KeyGraph method.

V. Creation of Software Project Scenario With KeyGraph Method

The data-set is generated on the basis of software case [5] as mentioned above to create KeyGraph.

Data-set D =
Company comprehensive software product particular software industry
Most users cannot afford cost of deployment
Software product too complex simple user
Industrial knowledge and advanced computer skilled Team of people dedicated
Increase market share scale down software product
Write new small system from scratch using existing and new codes
Block out some modules disables some functionalities

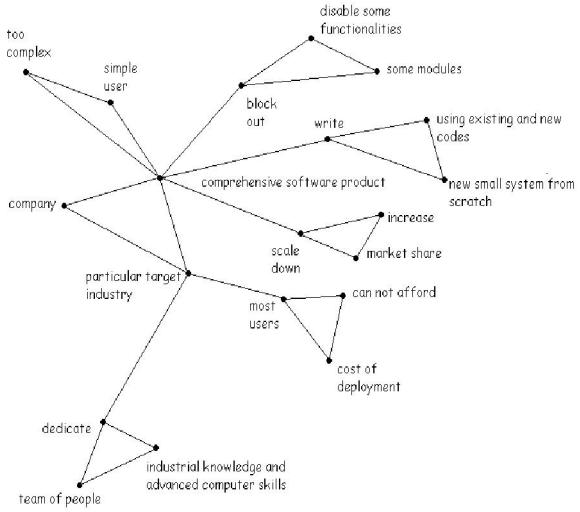

Figure 1. KeyGraph of software project scenario

After creation of KeyGraph for software project scenario as shown in figure 1, we create fundamental islands and in turn, bridges between islands. These bridges finally, generate significant hubs which are the software project risks themselves.

VI. Result

Finally, we got two sub-cases of software risks in the scenario as per the two options which were addressed in the software project case earlier.

If we consider the first option, i.e., to block out some modules in the original system and thus disable some functionality, this current scenario generates a significant hub discovered by the bridges of islands. This hub, considered as risk from the viewpoint of Chance Discovery, is recognized as performance risk and is shown in figure 2 below.

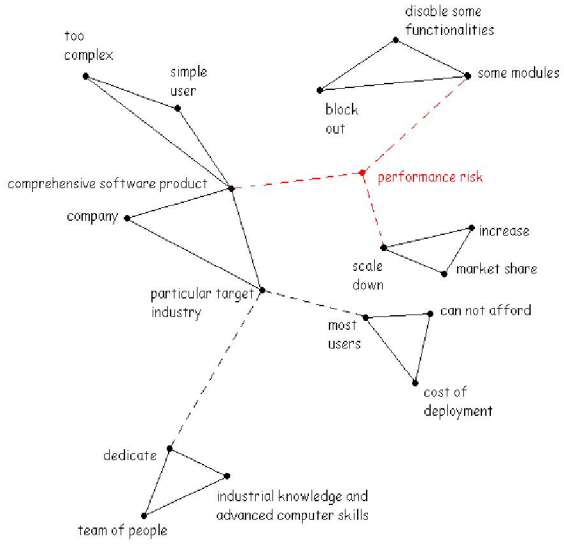

Figure 2. Identification of performance risk

Likewise, if we consider the second option, i.e., to block out some modules in the original system and thus disable some functionality, this scenario generates a significant hub discovered by the bridges of islands again. This hub, considered as risk from the viewpoint of Chance Discovery, is recognized as quality risk and is shown in figure 3 below.

Figure 3. Identification of quality risk

VII. Conclusion

We conclude our result by mentioning the successful identification of software project risk in the software project scenario.

The software risks scenarios are created graphically to simplify the problem domain. The graphical or indexed-based approach is moderately better approach to sort out the solution of such kind of software project risk identification problem than the other textual approach of solving the problem.

This makes user better understanding of the problem domain and problem itself to be approached to solve it because the problems can't be solved until it is identified. Henceforth, the main problem is always visualized by visual indexed-based approach during the entire analysis of the scenarios which is the most effective methodology. In other words, the current methodology is problem-centred or problem-focused rather than problem-dispersed approach.

This is done by the simplification of the data-sets that are associated with the respective scenarios. A proper investigation is always carried out during identification of the type of risks. This is the most critical task during the analysis of the problem.

The current tool is in fact, compatible to the need of our desire or purpose but still it is to be improved and enhanced so that better scenarios could be created and justified.

ACKNOWLEDGMENT

I am humbly thankful and acknowledge the Harbin Engineering University, College of Computer Science, and funding committee for their valuable supports and economic assistance.

REFERENCES

[1] Barry W. Boehm, "Software risk management: principles and practices", *IEEE Software, 8 (1), 1991, 32–41*.

[2] Berry W. Boehm, T. DeMarco, "Software risk management", *IEEE Software, 14 (3), 1997, 17-19*.

[3] Berry Boehm, Richard Turner, "*Balancing agility and discipline: a guide for perplexed*", Pearson education, 2004.

[4] Elaine M. Hall, "*Managing risk: methods for software systems development*", Addition-Wesley, 1998.

[5] Anne Fuller, Peter Croll, Limei Di, "A new approach to teaching software risk management with case studies", *CSEE&T 2000, IEEE, USA, 2002, pp. 215-222*.

[6] Robert N. Charette, "A risk of too many risk standards?" *INCOSE, USA, 2006*.

[7] Marvin J. Carr, et al, Technical report on taxonomy based risk identification, Software Engineering Institute, *CMU/SEI-93-TR-6, June 1993*.

[8] Y. Ohsawa, N. E. Benson, M. Yachida, "KeyGraph: Automatic indexing by co-occurrence graph based on building construction metaphor", *ADL'98, IEEE, 1998, pp. 12-18*.

[9] Yukio Ohsawa, "KeyGraph as risk explorer from earthquake sequence", *Journal of contingencies and crisis management, vol. 10, No. 3, 2002, pp. 119-128*.

[10] Katsutoshi Yada, et al, "The discovery of business chance from customer knowledge", *IECON2000, IEEE, vol. 3, 2000, pp. 1638-1643*.