Comparative study on uncertainty of software requirement and RISK analysis

Muhammad Usman, Shehzad Ahmed, Shahid M. Awan

Abstract— **Software requirement Engineering has gauged much attention for over a past few decades. It is one of the most serious domains considered in software development life cycle. Different systems show different types of uncertainties depending on requirements. As per the academics says "Requirements are naturally Unknowable". Many researchers worked on minimization of uncertainty of requirements in different times. This article evaluates and compares among few of the well-known state of art methods used for requirement gathering to minimize the uncertainty and risk that were adapted by different authors in different years. Benchmark techniques (True Positive rate, False Positive rate, ROC curves etc) are used to analyse the sensitivity and specificity for the respective techniques. Paper is concluded with RMMM plan that satisfies the risk factor.**

***Index Terms*— Uncertainty, decision problems, prior probability distributions, candidate architecture.**

I. INTRODUCTION

Software project life cycle consists of many stages of requirement analysis, design, implementation, testing and evolution. Now-a-days agile software development methodology is being followed. This is a new evolving paradigm in software development processes. In this methodology, change in all cycles of software project development is being monitored and change is continuous. This method particularly deals with change and uncertainty occurs at any level of development cycle. They ignore the usual facts of traditional development mechanisms such as of heavy documentation, contracts, specific employee roles, forward planning and strict follow-up of pre-defined steps. In agile methodology, face-to-face communication is more preferable. As change occurs continuously, then there might be big risk factors or uncertainty in requirements. Risk can be defined as an uncertain event, if it occurs, might have a positive or negative effect on project's failure or success. In agile development, major reason of failure of project is improper handling of unstable and volatile requirements. For this, traceability matrix are also created for determination of when, how, where and why change was occurred.

According to Kotonya and Sommerville, "Requirements provide the description of the system, its behavior, application domain information, system constraints, specifications and attributes". They also stated that major of the failures caused because of inappropriate requirement. Requirements can be classified in several ways such as external interface requirements, functional requirements, non-functional requirements, database requirements and derived requirements. These requirements can be explained as:

1. External interface requirements identify and document the interfaces to other systems and external entities within the scope of a project. For example: UI, H/W Interfaces, S/W Interfaces, memory constraints, dependencies and assumptions.

2. Functional requirements are also known software product features. These are the exact services provided by the system and how system reacts in particular inputs.

3. Non-Functional requirements (NFRs) are actually the capabilities of a system. For Example: Time consumption, standards and efficiency.

4. Database requirements are related to the specifications of a database needed.

5. Derived requirements include those requirements which are implied from design requirements.

In agile methodology, these requirements changes or varies from time to time. In some cases, they cause uncertainty but in field of software engineering, uncertainty is considered as a second-order concept. Common misconception is that by focusing on normal behavior, uncertainty can be avoided. But in general, uncertainty can only be minimized but cannot be removed fully. It is neither practical nor desirable to collect all of the information about a system. One reason of uncertainty can be defined as a user's perception of system. It might be possible that user finds it difficult to express the actual requirements. On other hand, there might be possible that the person recording requirements is not able to understand the view point of the user. Hence uncertainty occurs when there is loose coupling between system's user, adaptation logic, and business logic. Uncertainty can be caused by external sources or internal sources. External uncertainty arises from environment or domain in which the software is deployed. Internal uncertainty occurs when there is any impact of change or impact of replacing software component.

There are other systems known as self-adaptive systems. They are the most dynamic system. Their requirements are highly dynamic and hence level of uncertainty of high. In such systems, they modify it to satisfy change. Benefits of these systems are plenty but their process of development is very challenging due to high uncertainty of requirements. In this paper, uncertainty of requirements according to many different systems will be discussed in upcoming section. Related research work till yet will be discussed in section of related work. Different techniques or models to minimize the uncertainty will also be discussed in section of technical framework. Comparison would be done in section of comparative analysis by using different graphs. Future work will also be considered and can be analyzed in section of future work.

II. Related work

Many researchers worked on minimization of uncertainty in software requirements as well as on risk analysis. Josh Dehlinger and Jeremy Dixon considered mobile applications process to analyze the uncertainty in requirements. They stated that uncertainty may occurs at any level of the process. They discussed many issues raised from design level to the final development level. Authors described that in mobile applications, non-functional requirements are more critical and causes more uncertainty. They also discussed self-adaptive systems for minimizing the uncertainty and suggested different tools for the analysis of different types of requirements. For re-using of requirements, they suggested approach of software product line engineering (SPLE) approach.For analysis of requirements of context aware applications, authors stated usage of process of agent oriented software engineering (AOSE). AOSE is mostly used for single agent application while it is also applicable on multi agents system.They used RELAX language for expressing requirement specifications for self-adaptive systems.

Tharwon Arnuphaptrairong listed top ten software project risks in his research paper. He studied papers from year 1981 to 2003 to describe dimensions of risks as well as how many software risks were found in each study. In his paper, he summarized the work of other researchers such as six dimensions of software risks by Wallice Et Al are stated as user, requirements, project complexity, planning a& controlling, team and organizational environment. By considering these dimensions, he explained all software risks of each category. Tharwon also listed risk factors stated by Boehm, Schmidt et al (USA, Hong Kong & Finland), Addison, Vallabh, Han & Huang and Pare et al. He concluded by calculating frequency of each dimension of software project risk and stated that the main factors of risks are lack of requirements, changes to requirements, failure to satisfy user expectations, lack of user involvement and lack of management and support. He also ranked these factors after conducting proper survey.

Veerapaneni Esther Jyothi andK. Nageswara Rao studied agile methods. They analyzed the requirements in agile methodologies such as: Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD) and Crystal Family. Authors listed main causes of project failures such as: incomplete requirements, less user involvement, lack of resources, changes in requirements and lack of planning. They also ranked these failure causing factors in percentage. Authors then proposed methodology of traceability of requirements in agile practices. They explained the importance of traceability of requirements as well as they described how to add levels of traceability when changes in requirements occur. They listed few tracing practices such as: tracing of stakeholders requirements, tracing of requirements of problem, tracing of requirements of product backlog, tracing of requirements of sprint backlog, tracing of requirements of code, tracing of requirements of testing and documentation. At the end, they concluded with eleven principles to reduce the risk factors as well as uncertainty in software requirements.
Rahul Thakurta proposed a framework for prioritization of requirements of a software project. He explained algorithm for non-functional requirements (NFR) prioritization. NFR algorithm consisted of six major steps. First step was to identify the NFRs. Second step was the creation of project level scenario. This scenario is then linked to the objectives in third level of the algorithm. Assessment was done in next step. Adjustment according to the score of NFRs will be done in fifth step and in last step; a heuristic was designed to decide the dropped NFRs from scenario. Then researchers took a case study and validate these steps. At the end they compared some prioritization techniques and stated their weaknesses and strengths.

Naeem Esfahani and Sam Malek considered self-adaptive systems for the explanation of uncertainty of requirements. They took an example of robotic software which a self-adaptive system and analyzed the uncertainty factors that were occurred. They stated important reasons of occurrence of uncertainty such as: simplifying assumptions, model drift, noise, parameters changes, decentralization and changing in context. They also discussed the impact of uncertainty in whole self-adaptive system. Authors also presented different mathematical theories for expressing uncertainty factors. They also used RELAX language for documenting the requirements.

Emmanuel Letier, David Stefan and Earl T. Barr presented uncertainty and risk in software requirements as well as in software architecture. They described cost-benefit analysis under uncertainty with impact of information on risk. They also discuss design decisions under uncertainty. They took experiment using cost-benefit model for taking decision on design issues and then also define design risks as well as architecture issues.

III. Proposed Methodology

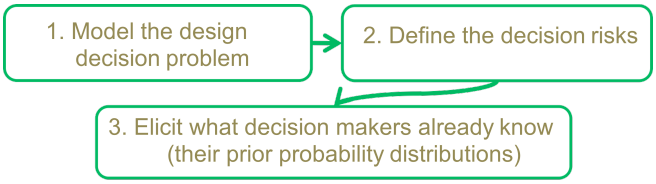
In this paper, the main focus is to reduce risk using proposed model and RMMM plan. There are few steps that should be followed before the execution of RMMM plan.

First step is to model the design decision problem. A designer should list down all the design problems. For instance, if there is a mobile application development, then a designer should consider problems related to user interface (UI) for all types of screens such as for I phones, window phones, android and blackberry. As there are large amount of difference and variations in the screen size, so designers faced difficulties in finding appropriate requirements for each

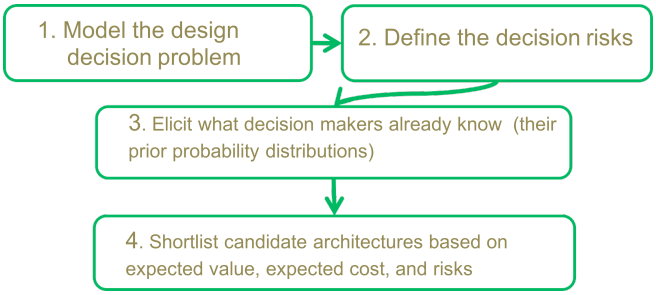
kind of resolution. For this, they selected requirements on group based sizes.

Secondly, define the process of reusability of software across different mobile phones. Each mobile phone is using different operating system platform, different hardware as well as different computing formats. Designers should consider all options during the process of mobile application software engineering because all have different influences on software requirements. If single platform will be targeted then developers have to build a single application that should be enriched in all platforms requirements with high risk of functional as well as non functional inconsistencies.

Third level is of designing of context aware mobile applications. As now-a-days mobile applications are going through rapid change so there should be such application which is dynamic. Mobile phones are using for each process such as for time awareness, location awareness, weather awareness and device awareness etc. Hence it should be context-aware mobile application. For such aware nesses, requirements should be keenly analyzed by the developers' team to get better product at the end. Second step of proposed methodology is to define the decision risk. After going through all requirements for all levels of process, there comes a major point to balance the change and uncertainty in requirements. Because of contextual nature of mobile application, it might be possible that application does not fulfils all functional as well as non functional requirements, hence this initiates the need of self-adaptive application. In self-adaptive applications, rich features are provided with less stringent requirements. In self-adaptive application, it runs in normal conditions with normal requirements but when needed, it will modify its behavior automatically which somehow reduces functionality but kept on providing it rather than providing nothing. For instance: if there is a location-based application, features needed for such application are GPS and it requires high consumption of battery. For such application, it is better to use data of old used location rather than giving no data. For this self-adaptive process, mobile application requirements must be integrated fully with agile development methodology. Hence developers should be more aware of the requirements integrations with the application for better understanding of application. Third step of proposed methodology is to elicit what decision makers already know (their prior probability distributions). If decision makers know it early that mobile application should only be self-adaptive application then there is no need of gathering information of any other application type and just start analyzing the requirements for self-adaptive application.



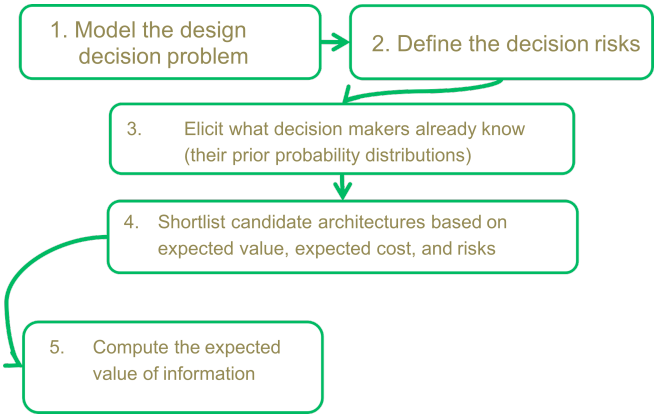
Fourth step is to shortlist candidate architectures based on expected value, expected cost, and risks. While considering the mobile application development process, main architecture decided by most of the authors as well as researchers was of self-adaptive system, Because of its less cost and risk. It has capability of at least fulfilling all functional requirements with many of other non-functional requirements as well.



Fifth step is to compute the expected value of information. In this step, collected information is computed whether it is enough or there is a need to analyze the requirements more. In mobile application development, many authors and researchers used many tools in every step of development process. Authors recommended Software Product Line Engineering (SPLE) which develops a suite of application which shares all common requirements as well as it manages all of them. It will be advantageous in the process of reusability and reduces cost of analysis and development processes. In SPLE approach, there are two main phases such as domain engineering and application engineering. In domain engineering phase, requirements that are common as well as variable are defined while in application engineering phase, all defined requirements are used to develop any application. This will help developer to understand that which requirements are common for what type of design and platform. By using this, it is possible to develop different products by using single team to developer as they already know common set as well as variable set of requirements and its easy for them to develop application in less time. Hence SPLE helps in duplication of early software engineering process work such of analyzing requirements.

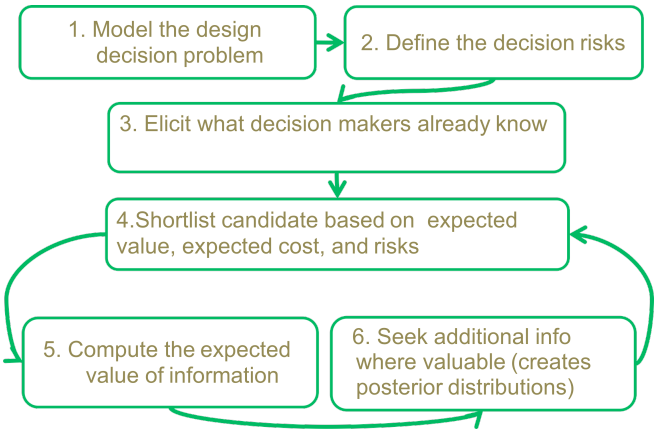
To provide, self-adaptive as well as context-awareness in application, approach named RELAX is most widely used. This is requirement specification language which provides medium to express behavioral and environmental uncertainty considering the dynamically adaptive system. RELAX shows two main types of requirements such as variant and invariant requirements. Variants requirements are those requirements that should always be satisfied while invariant are those which may or may not be satisfied. Both requirements are specified using structural natural language by using different operators, temporal, modal and fuzzy logic. In case of every variant requirement, RELAX process a document that what

are the effects caused by the environmental changes on the requirements and how they can be satisfied. It also documents uncertainty of requirements as well as how to adapt any change by the application while facing such uncertainty and also provide some functionality too. This will help developers a lot when it is integrated with agile methodology to provide better structures as well as better delivery of non-functional requirements in changing environment.



Sixth and last step is to seek additional information where valuable (creates posterior distributions). By using the above mentioned tools, there is a possibility of adding as well as deleting many of requirements according to the change in any level. This step will be helpful to track change whenever there is a change in requirements. It will keep the record of changes and helps in reducing risk.

Now after proceeding all above steps, RMMM plan will be executed. This plan might be executed as a part of software development process or might be a separate document. In most of the projects, organizations prefer to make a separate document of RMMM. Once it is documented, steps of risk mitigation and monitoring will be started. In risk mitigation, there is an activity to avoid problem by using many possible solutions while in risk monitoring process, there will be a project tracking activity.



It has three main objectives such as: it will assess whether predicted risks occurs or not, it will ensure that risk aversion steps defined for the risk are being properly applied or not and it will collect information that can be used for future risk analysis.

The findings from risk monitoring may allow the project manager to ascertain what risks caused which problems throughout the project and hence it is helpful for tracking of a project.

IV. Some Common Mistakes

Some researchers had done good work on minimization of risk as well as they reduced uncertainty up to some level. But most of them does not achieved the highest level of minimization. Mobile application development was very well considered by Josh Dehlinger and Jeremy Dixon. But they have not analyzed the requirements from the start of the process. At the end they just concluded that self adaptive application is better. Such application can only gives benefit up to some level and functional requirements are highly considered in such applications. Non functional requirements are somehow ignored and they were actually the most important thing when mobile application is under considerable situation. Similarly, in other paper by Tharwon Arnuphaptrairong, top ten risks were listed. He states many kinds of risks as well as their calculations while considering many dimensions of risks. The main problem was huge number of calculations. In this technique, most of the time was consumed in just analysis and with no output. Though he also ranked the risk factors according to their priority but still huge calculations was needed. Requirements of agile methodologies were discussed by Veerapaneni Esther Jyothi and K. Nageswara Rao. They listed major failures reasons of the project. Also they ranked these failures according to their priorities. The major mistake they made that they does not have followed any proper process for each type of agile methodology. They just used traceability matrices for tracing of requirements at each level. They have also listed some principles of reducing the risk factors which were good but proper process was not defined. Non-functional requirements were well considered by Rahul Thakurta. He proposed a proper framework for prioritization of requirements of a software project. He also presented an algorithm for this purpose named as NFR algorithm. But this algorithm can only be applied to some specific type of requirements and not to all requirements. Impacts of uncertainty of requirements were discussed well by Naeem Esfahani and Sam Malek. But they took only self adaptive system for their experiments as well as mathematical theories presented by them was highly time consuming. Emmanuel Letier, David Stefan and Earl T. Barr also presented their work in this field of minimization of risk. They have used cost-benefit analysis model. This model is only good for design decisions and not enough good for further levels of development process.

V. Conclusion

As discussed earlier in this paper about the common mistakes of many researchers, our RMMM plan is considered better as compare to other discussed techniques, plans and processes. RMMM plan is easy to execute as well as it is not much time consuming. There are only few steps to execute. Some steps

are iterative in nature which helps the executer to refine the requirements in lesser time. Also RMMM plan is suitable for all kinds of requirements at any level of development process. Hence suggested methodology is best as compared to other methodologies of other researchers.

VI. Acknowledgment

First Author – Muhammad Usman
Mphill Computer Sciences [University of Management and Technology - 2016]
Project Lead – Ufone Pakistan

Second Author - Shehzad Ahmed
Mphill Computer Sciences [University of Management and Technology - 2016]

Shahid Awan

VII. References

- Josh Dehlinger and Jeremy Dixon. (2012), Mobile Application Software Engineering: Challenges and Research Directions.
- Tharwon Arnuphaptrairong. (2011), Top Ten Lists of Software Project Risks: Evidence from the Literature Survey.
- Veerapaneni Esther Jyothi andK. Nageswara Rao. (2011), Effective Implementation of Agile Practices.
- Michalis Famelis, Rick Salay, and Marsha Chechik. (2013), Partial Models: Towards Modeling and Reasoning with Uncertainty.
- Rogerio de Lemos, Holger Giese, Hausi A. Müller And Mary Shaw. (2011), Software Engineering for Self-Adaptive Systems: A Second Research Roadmap
- Diane E. Strode* , Sid L. Huff, Beverley Hope & Sebastian Link. (2012), Coordination in co-located agile software development projects.
- Rahul Thakurta. (2012), A framework for prioritization of quality requirements for inclusion in a software project.
- Naeem Esfahani and Sam Malek. (2012),Uncertainty in Self-Adaptive Software Systems.
- Gabriel Tamura, Norha Villegas, Hausi Muller, Joao P. Sousa, Basil Becker, Mauro Pezze, Gabor Karsai, Serge Mankovskii, Wilhelm Schafer & Ladan Tahvildari. (2012), Towards Practical Runtime Veri_cation and Validation of Self-Adaptive Software Systems.
- Emmanuel Letier, David Stefan & Earl T. Barr. (2014), Uncertainty, Risk, and Information Value in Software Requirements and Architecture.