International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:4, No:4, 2010

Meta-requirements that Model Change

Gouri Prakash

***Abstract*—**One of the common problems encountered in software engineering is addressing and responding to the changing nature of requirements. While several approaches have been devised to address this issue, ranging from instilling resistance to changing requirements in order to mitigate impact to project schedules, to developing an agile mindset towards requirements, the approach discussed in this paper is one of conceptualizing the delta in requirement and modeling it, in order to plan a response to it. To provide some context here, change is first formally identified and categorized as either formal change or informal change. While agile methodology facilitates informal change, the approach discussed in this paper seeks to develop the idea of facilitating formal change. To collect, document meta-requirements that represent the phenomena of change would be a pro-active measure towards building a realistic cognition of the requirements entity that can further be harnessed in the software engineering process.

***Keywords*—**Change Management, Agile methodology, Meta-requirements

I. Introduction

THE practice of requirements engineering [1,5], usually occurs in a silo, at the beginning of the software development lifecycle, with the objective of eliciting, representing and validating requirements for the software development endeavor. The emphasis, during the requirements engineering phase is on "what" needs to be coded into software as opposed to "how" and thus requirements engineering is undertaken to outline the context within which software engineering should be performed. The end of the requirements engineering phase is often interpreted as the "freeze" point for any significant changes to the existing body of requirements. Agile software development methodology[2] is an approach that has been identified to work around the "freeze" mindset and this methodology engenders reactivity to changes as opposed to resistance, which makes traditional, conservative, software development practitioners skeptic of its viability and more importantly reliability.

Requirements are increasingly recognized as having the characteristic of being mercurial in nature and as requirements change, there is a need for creating software development processes that recognize the changing nature of requirements and integrate change as a concept within the software process model. There are two types of change – formal change and informal change. Formal change requires support by a structured process in order to facilitate the change whereas informal change requires a mindset that accepts and adapts to the change, without having the need to spend significant amount of resources on justifying the nature of the change and documenting it in detail for implementation purposes. Agile methodologies like Scrum, XP [6,7] have been created such that these recognize informal changes as a best practice which also contributes to making the software development endeavor more customer-centric. However in non-technology firms, the emphasis is on implementing changes to requirements in a more formal, well-documented manner, in an effort to increase code portability and maintainability during the lifecycle of the software product. Software coding conducted in non-technology firms is performed for building systems that can be supported and maintained by a changing body of personnel, hence the need for supporting formal changes, in order to mitigate the risk of loss of knowledge and information. Employing agile practices to facilitate change is an adaptive approach towards software engineering and one way of addressing changing requirements. This paper aims at propose an alternative approach towards addressing changing requirements and that is to formulate meta-requirements during the requirements engineering phase, that model and communicate the phenomena of change and the impact it has on the software development lifecycle. Before entitizing change let us look at what we mean by meta-requirements.

Meta-requirements refer to data that further describes the attributes of a requirement, giving more information about the requirement which can be significant for the software development effort. Active formulation of meta-requirements when initiated during the requirements engineering phase and integrated into the software engineering process, leads to the provisioning of information about requirements which enmeshes the practice of requirements engineering with the other phases of the software development lifecycle as opposed to being isolated to a silo of the software engineering process. Meta-requirements creation entails gathering and provisioning of relevant information about requirements which can potentially be of use in all phases of the software development lifecycle. For the purpose of this paper, the focus is on meta-requirements that model the attributes of change as well as concepts pertinent to it in order to facilitate and develop a realistic cognition of the requirements entity.

Gouri Prakash is with the Business Risk Infrastructure Department at HSBC USA 93901 (Phone: 412-512-6743; fax: 831-754-4031; e-mail: gouriprakash@ live.com).

II. META-REQUIREMENTS

A. Empirical results validating status quo

Are meta-requirements collated in contemporary software development projects and if so what are these meta-requirements? The empirical validation was performed for a small sample of software projects conducted between the year 2006 and 2007 at an investment bank. Based on data collected from 8 different software development engineering projects, it was determined that meta-requirements were collected and even represented in the requirements engineering phase, however the data collected was not explicitly recognized as meta-requirements and was collected more in the line of following standard, repeatable practices. Furthermore, it was uncovered that the meta-requirements gathered for the projects were not actively referred to or used in other phases of the software development project.

The following lists the set of meta-requirements that were common across all 8 projects and were part of the requirements representation effort:

- **Requirement Owner:** The requestor of the requirement, representing the person who had a stake in requirement implementation.
- **Requirement Submission Date:** The date on which the requirement was created and submitted to or handed over to the software project manager.
- **Requirement Expected Implementation Date:** A date communicated to the requestor of the requirement as the date on which the requirement would be realized by the system. This attribute therefore indicated the expected implementation date of the requirement.
- **Requirement Version:** The version of the requirement, indicating the number of revisions the requirement has undergone to reach its current form of definition.
- **Requirement Impact:** Impact, specific to the domain of requirement implementation, and qualitatively described in terms of the impact on the stakeholders of the software development projects. These stakeholders were also the requestors of the requirement and were tasked with building a case for it.
- **Requirement Criticality:** Criticality ranked certain requirements as being of relatively higher importance than other requirements and was a culmination of relative and subjective evaluation of the requirement itself – an input that was actively sought by the software development team, from the requestors of the requirement.
- **Risk of not implementing the requirement:** The risk of not implementing the requirement was in some cases articulated in terms of financial losses and in others was expressed as a qualitative assessment of the adverse impact to the business if the requirement was not implemented.

The above indicates that the meta-requirements collected focused primarily on the time dimension and the significance dimension of their respective software engineering projects. The dates and version type of requirements indicate the progression of requirements over a period of time while the other meta-requirements emphasize the need for implementing the requirement. Based on empirical analysis of software projects, the meta-requirements collected are not explicitly recognized as meta-requirements and more importantly in their current representation restricted to usage in the requirements engineering phase and to some extent used in the software validation phase of the software development lifecycle.

The objective of creating meta-requirements, which models the phenomena of change, is to realistically represent requirements and more importantly set the expectations of the software development team about the requirements entity. Modeling the phenomena of change as meta-requirements isolates the representation of change as a by-product of requirement definition and keeps the focus on change separate from the actual requirement. Entitizing change, discovering and defining the attributes of change, thereby modeling it for the benefit of harnessing it for representation in the requirements engineering phase as meta-requirements, creates a degree of acceptance towards change.

B. Modeling change

What attributes of changing requirements can meta-requirements model and represent and how would these attributes be useful in the overall software development process? To understand in greater detail, let us look at the change in requirements from a conceptual standpoint. Let there be a requirement $\boldsymbol{r}$ that has been identified and defined for a software development project. In the requirements engineering phase, we are seeking to model a possible change to $\boldsymbol{r}$ – let us call this change as $\Delta\boldsymbol{r}$. We now know that at a given instant of time $\boldsymbol{t}$, there exists a probability that the requirement $\boldsymbol{r}$ will change states and assume the state of $\boldsymbol{r} + \Delta\boldsymbol{r}$. Let us denote this new state with $\boldsymbol{r'}$. Fig 1 below indicates the two states a requirement can go into, which is persist the same state as indicated by $\boldsymbol{r}$ or change to a new state indicated by $\boldsymbol{r'}$**.** Furthermore, the probability that a requirement will persist the state $\boldsymbol{r}$ is given by $\boldsymbol{p}$ and the probability that $\boldsymbol{r}$ will change to $\boldsymbol{r'}$ is given by $\boldsymbol{1-p}$.

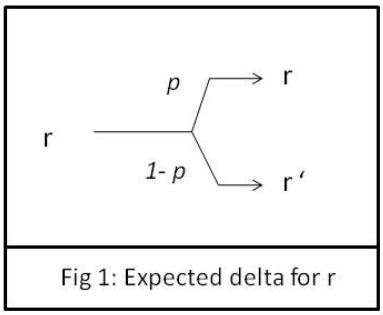

Fig 1: Expected delta for r

Supposing it is known that multiple changes to the requirement are possible, then the individual probabilities associated with each of the possible changes can be used to compute the combined probability that a change to the requirement can occur. The change to the requirement can be documented into the requirement change matrix as shown below. The four columns of the requirement change matrix, lists all the requirements of a sample project along with the probability that a change will occur. Furthermore the impact of change is identified and is assigned a positive or a negative connotation. Certain changes to requirements, such as removing the requirement altogether can actually be viewed as having a positive impact on the project. The final column documents whether the change occurred. Note that the assumption here is that the change in each requirment itself is atomic and not composite – implying that either all aspects of the change will occur or not occur and hence the expected change itself is well-defined.

TABLE I
REQUIREMENT CHANGE MATRIX

Requirement (R)	Probability of Delta(ΔR_N)	Impact of change	Change Occur?
R_1 - ΔR_1	0.7	+	Y
R_2 - ΔR_2	0.1	-	N
R_3- ΔR_3	0.6	+	N
R_4 - ΔR_4	0.2	+	N
R5-ΔR_5	0	-	N
R6-ΔR_6	0.5	-	Y
R7-ΔR_7	0.3	-	N

Instead of viewing change solely as a project risk with negative connotations, change needs to be acknowledged as an event that can occur after the requirements have been baselined, resulting in potentially multiple implications for the software engineering endeavor itself – especially in terms of schedule and budget, and these implications indicate whether the change is a risk or an opportunity, negative or positive.

III. ROLE OF CHANGE MODELER

The objective of collecting meta-requirements is to realistically address requirements and more importantly capture the realistic nature of requirements in a way that is useful and suitable for all members of the software development endeavor. SPL v5 framework [3], in defining the practice area of requirements engineering, recognizes the three roles that are needed in arriving at the requirements specification – the user, the developer and the requirements engineer [4]. The user is the requestor of the requirement, the developer is the software engineer who will create software that will implement the requirements and the requirements engineer will engage in activities related to requirements elicitation and specification. To arrive at the meta-requirements that model change in requirements is a task that can be undertaken by the requirements engineer or can be performed by a separate, independent role and this role would also be the explicit change agent.

That ***r'*** will occur has been proven empirically and is now a recognized phenomena in the software engineering industry. Estimating what the value of ***p*** will be requires addressing the uncertainty associated with defining the state of ***r'***. It is important to note that arriving at the definition of ***r'*** and associated probability should not be attempted by the software engineer as the focus for the software engineer is on requirement implementation. To arrive at ***r'***, it is necessary to engage the source of ***r'***, which is a stakeholder requesting the requirement implementation in order to determine ***p*** and ***1-p***. The objective is to anticipate changes in requirements and be able to represent it in a manner which is independent of documenting base lined requirements and implementing these.

IV. A CASE STUDY

A software project that was initiated to replace the existing file system structure of a small business with a centralized document repository was used for validating the approach and utility of collecting meta-requirements. The project started with collecting data from the file structures stored on the shared drive of the network supporting the documents used by employees of the small business. An evaluation was performed on the IT infrastructure options for the company and solutions were identified that could replace the fragmented documentation structure used by the company. When the technical architecture was created it was determined that the enterprise version of MS-Access could be deployed to meet the IT needs of the company while certain documents could continue to reside on the shared drive. A requirements document was base lined and this document contained 16 functional requirements and 5 non-functional requirements. The functional requirements were arrived at by gathering information present from the files recovered from the company's shared drive as well as by conducting interviews with stakeholders.

Given that the end users of the system were not tech-savvy, it was identified that the requirements arrived at with the help of existing documentation and stakeholders may be subject to change at various stages of the project including the user acceptance testing. Anticipating the mercurial nature of the requirements, a change modeler who had very good domain knowledge of the small business, was also assigned to the project and was tasked with modeling meta-requirements. The requirement change matrix was created that would be progressively filled and maintained by the change modeler.

The table below outlines the requirements change matrix documented and maintained by the requirements change modeler. The change in the requirement was not only anticipated but also documented wherever it occurred.

TABLE II
REQUIREMENT CHANGE MATRIX FOR FUNCTIONAL REQUIREMENTS

Requirement (R)	Probability of Delta(ΔR_N)	Impact of change	Change Occur?
R_1 - ΔR_1	0.0	-	N
R_2 - ΔR_2	0.1	-	N
R_3 - ΔR_3	0.0	-	N
R_4 - ΔR_4	0.2	+	N
R5-ΔR_5	0.0	-	N
R_6-ΔR_6	0.3	-	N
R_7-ΔR_7	0.4	-	Y
R_8-ΔR_8	0.3	-	N
R_9-ΔR_9	0.2	+	Y
R_{10}-ΔR_{10}	0.6	-	N
R_{11}-ΔR_{11}	0.7	-	N
R_{12}-ΔR_{12}	0.4	+	Y
R_{13}-ΔR_{13}	0.2	-	N
R_{14}-ΔR_{14}	0.5	-+	N
R_{15}-ΔR_{15}	0.3	-	N
R_{16}-ΔR_{16}	0	-	N

The findings were analyzed post-project implementation and it was uncovered that those requirements that were uncovered early on and determined to be core requirements were assigned low probabilities. However, requirements that were cosmetic by nature and that were not categorized as core requirements had significantly higher probability for changing. Essentially the latter were good to have requirements as opposed to the must-haves. Also the change modeler had to take into account that the change anticipated in the requirement could be positive from a business standpoint but negative from a technology standpoint and would have to perform a cost-benefit analysis in order to determine whether the potential change had an overall positive or a negative impact.

The limitation of the findings uncovered in the above study lies in the fact that it represents the findings uncovered from a single project but is useful for recording empirical observations. Also the requirement change matrix was not created for non-functional requirements for the project so the utility of such an endevour cannot be ascertained for non-functional requirements.

V. CONCLUSION

It is important to note that so far we have arrived at two possible attributes of interest for formulating meta-requirements – (1) possible modification to requirement which is represented by ***r'*** (2) probability of occurrence of a modification to the requirement represented by the probability ***(1-p).*** Further deductive and inductive modeling of the change entity can be pursued to uncover other relevant attributes of change that will create a model with attributes of change that should be uncovered while collecting meta-requirements and facilitate a climate within software engineering that recognizes change and accepts it.

The attributes uncovered thus far were used for recording observations on a single project to evaluate the approach. It was uncovered that addressing potential changes to requirements was a proactive approach that can be used for elucidating and mitigating project risk as well as identifying opportunities, where change would have positive implications.

The requirement change matrix has been introduced as a tool that can be used to further document and represent the potential change to requirements as well as the impact to the overall project. The meta-requirements discussed during empirical derivation of contemporary projects was also collected and documented during the study undertaken to validate the approach mentioned in this paper.

The overarching benefit of the approach discussed in this paper is to create a realistic cognitive approach towards changing requirements that seeks to formally integrate the phenomena of changing requirements into the software engineering process.

ACKNOWLEDGMENT

The author would like to thank the organizers of the conference for giving academic researchers and industry practitioners the opportunity to address issues and challenges encountered in the software industry in a dedicated forum.

REFERENCES

[1] Kotonya, G., and Sommerville, I., *Requirements Engineering Processes and Techniques.* John Wiley and Sons., 1998, NY
[2] Sidky, A. and Smith, G., *Becoming Agile in an imperfect World* , Manning Publications Co., Greenwich CT 2009
[3] Software Product Line v5 documentation, http://www.sei.cmu.edu/productlines/ , Software Engineering Institute, Carnegie Mellon, July 2009
[4] Barbara Paech, "What Is a Requirements Engineer?," *IEEE Software*, vol. 25, no. 4, pp. 16-17, July/Aug. 2008, doi:10.1109/MS.2008.106
[5] H. Elizabeth, J. Ken and D. Jeremy, "Requirements Engineering", Springer Publications, 2005
[6] K. Schwaber and M. Beedle, "Agile Software Development with Scrum", Prentice Hall, 2001
[7] S.W. Ambler, "Agile Architecture: Strategies for scaling Agile Development", 2001-2008, http://www.agilemodeling.com