Uncertainty of Requirements in Agile Requirement Engineering Process

Ansar Ahmed,
Department of computer Science,
University of Management and Technology,
Lahore, Pakistan.
ansar.ahmed@umt.edu.pk

Dr. Shahid Mehmood Awan
Department of computer Science,
University of Management and Technology,
Lahore, Pakistan.
shahid.awan@umt.edu.pk

***Abstract*—Uncertainty entangles early necessities and structural engineering choices furthermore, might uncover a product task to noteworthy danger. Yet programming modelers need support for assessing uncertainty, its effect, and the benefit of diminishing instability before making basic choices. Agile requirements engineering goes for applying agile contemplation to traditional requirement engineering. It is the enhancement and change of traditional requirement engineering, getting it fit to the persistent changes of requirements. This paper differentiates the agile requirement engineering and the conventional one, concentrates how agile requirement engineering practices could be connected to uncertainty (*Abstract*)**

Keywords— Uncertainty; agile;Requirement engineering ;

I. INTRODUCTION

Requirement Engineering is a branch of Software engineering that applies distinctive procedures and techniques for the prerequisite investigation, customer's requirements are guaranteed by prerequisite designers, engineers while examiners comprehend the product to be created through these necessities and they are likewise completely mindful of various components and impediments of the framework being created. Traditional Requirement Engineering takes after the documentation for the framework advancement that is built by social affair all framework necessities from the customers at right on time stage. In the Traditional waterfall model necessity social affair is finished at right on time stage and engineers need to accomplish all capacities of the framework in a cycle. This makes an exceptionally troublesome circumstance at the point when customers need a few changes in given necessities.

We propose to apply choice investigation and multi-objective streamlining procedures to give such backing. Similarly Uncertainty is lake of complete knowledge about the actual consequences of the iterative. Uncertainty cannot be avoided in early phases of software development. The first is when organizations have options which project to initiate and second, when software architect make decision about software system. The aim, of these decisions, is to maximize benefits and minimize cost and time. Uncertainty is about objectives of the project, impacts of available alternatives on the objectives, cost and benefit of available alternatives, future changes in the project objectives, business context and technological environment.

In the present day business society, as information technology creates, business sorts and environment changes, requirement is turning out to be an increasing amount immaterial—changing quickly and difficult to characterize, which conveys difficulties to traditional requirement engineering (TRE).Traditional waterfall model, engineers could fulfill every one of those elements of the framework inside of a cycle, while customers can't clear their Requirements at once. Accordingly, the developing process won't not be fit. To this point, the extensive documentation of necessities is no more material, which will bring about the gigantic monetary misfortune, as well as additionally the deferral of programming conveyance. Finding new technique and rehearse turns into the most squeezing need to change the present status of necessity building.

In this manner, agile requirements engineering (ARE) rises.

In this paper we focus on how Uncertainty of Requirements in Agile Requirement Engineering Process can affect the software and its architecture. Compared with TRE, what's the favorable circumstances and inconveniences of ARE? Instructions to apply ARE to various unobtrusive measured data frameworks advancement ventures by component? We have to calculate uncertainty and how additional information can reduce uncertainty and failure of software.

In the previous research they considered waterfall methodology and current software engineering industry has shifted towards agile software development which has full involvement of customer and stakeholders as well which were not considered in previous research.

Current approach of this paper is on the following formalizing decision problems in terms of domain specific measurable goals, elicit and represent uncertainties as probability distribution, value of information (from decision analysis) it helps in measuring uncertainties about alternatives. Also it determines which could be profitably reduced.

Contributions of the paper are systematic method for applying statistical decision analysis to requirements at every iteration; define a decision matrix for requirements using agile process

and information value- how additional information could reduce risks.
We have to take an alternative based on cost benefit model which will be applied in every iteration. It means to select one alternative among a set of alternatives based on their cost and benefits whenever we got some new information that can reduce uncertainty.

II. Cost Benefit Analysis and Value of Information

Cost-benefit model comprises of set A of alternatives, set Ω of model parameters, Two functions: Cost(a,ω) a = single alternative from A and Benefit(a,ω) ω = parameter values and Net benefit NB(a, ω)=benefit (a, ω) –cost (a, ω).
The normal estimation of data advises the leaders to focus on lessening instability about data with high expected quality. The normal estimation of data is characterized with respect to a result (net advantage/programming dependability/or something else) to be augmented. The normal estimation of data advises the leaders to focus on lessening instability about data with high expected quality. The normal estimation of data is characterized with respect to a result (net advantage/programming dependability/or something else) to be augmented.

III. Methmtical Modeling of Risk

EVTPI= expected value of total perfect information
EVTPI is the expected gain in the net benefit from using perfect information about all information. Mathematically
EVTPI=net benefit when perfect information-current uncertainty. EVTPI=E [max NB (a,ω)]-max E[NB(a,ω)]. By MC simulation (NB)ˆ is mean max (NB)ˆ[i,j] – max mean (NB) ˆ [i,j] Partial perfect information means perfect information about a subset of the model parameters. Ω= for all parameters Θ= for single parameter. Total/Partial imperfect information that reduces uncertainty but without completely eliminating it. EVPPI (Θ)=E [max f(a,θ)]-max E[NB(a,ω)]=E[maxENB(a,ω)]-max[NB(a,ω)]. f(a,θ= expected net benefit of alternative a conditioned on the parameter Θ fixed at θ .EΩ-Θ = expectation w.r.t all model parameters in Ω except Θ.

IV. Case Study

A contextual analysis for public hospital has been picked. This task of Hospital Management Information System (HMIS) has been created by programming industry advancement group. The specialists have connected deft prerequisite building for the improvement of this undertaking. They have as of now involvement with Traditional Requirement Engineering. We have assessed the outcomes on the premise of their master sentiments/reactions. HMIS comprises of taking after center modules: Patient Panel, Admin Panel, Lab test administration, User administration and so forth. Opinion of experts has been given in the form of 'Yes' and 'No' and has been tabulated in table II against success factors.
Graphical representation of this expert's response is shown in figure.

The product business has been wanting to add to this venture, encourage more they allocated the assignment to particular improvement colleagues which comprise of aggregate 12 specialists out of 2 Data base heads, 2 Managers, 3 Developers,3 Technical Writers and 2 QA specialists. They were moved to light-footed programming advancement from conventional programming improvement. We assessed the outcomes through meetings with the specialists that why they moved to light-footed advancement. We chose 10 basic achievement elements from. On the premise of these basic achievement elements we met these specialists and assessed the outcomes. It was watched that Agile RE was showing improvement over Traditional RE with the accompanying elements are given underneath:
1. Small Duration Project (SDP)
2. Project Team with Expertise (PTWE)
3. up front Risk Analysis (URA)
4. Good Customer Relationship (GCR)
5. Face-To-Face Communication (FTFC)
6. Right Amount of Documentation (RAOD)
7. Flexibility (FLXB)
8. Responsive To Change (RTC)
9. Correct Integration Testing (CIT)
10. Effective Delivery Management Process (EDMP)

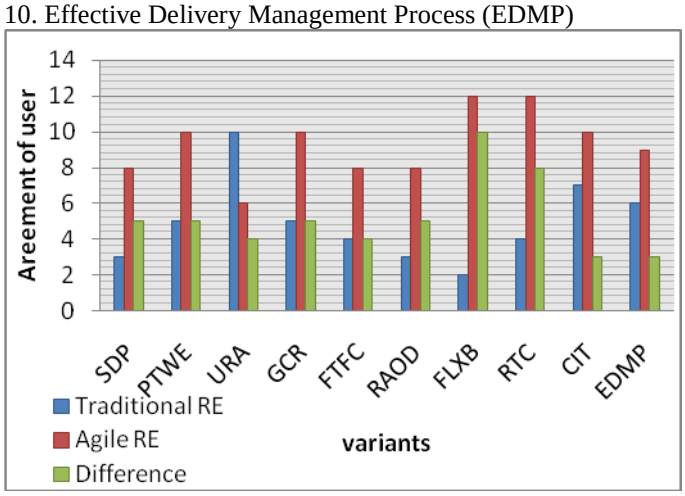

Figure 1: Risk assessment, Traditional vs Agile

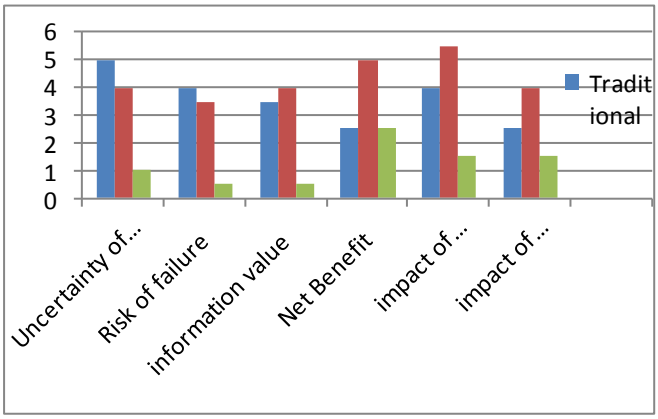

Figure 2: Comparison of Risk with Traditional and Agile methodology

Critical Successes factors	Y/N	SDP	PTWE	URA	FTFC	RAOD	FLXB	RTC	CIT	EDMP
Traditional RE	yes	3	5	10	4	3	2	4	7	6
	no	9	7	2	8	9	10	8	5	3
Agile RE	yes	8	10	6	12	8	1	1	1	9
	no	4	2	6	0	4	20	20	02	3
Difference		5	5	4	8	6	10	8	3	3

Figure 3: Comparison of Risk with Traditional and Agile Methodology

V. Defining the Architecture Decision Model

Multi objective architecture decision model (MOADM) consists of recognizing the choices to be brought together with their alternatives, characterizing the objectives against which to assess the choices, characterizing the objectives against which to access the choice.

VI. Calculating Cost and Benefit at Every Iteration

When the number of objectives and iterations increase the problems also increase. MOADM may have many a goal. It is better to convert MOADM to cost-benefit model. CBM allows architect to design a model which is financially good for something.

Software architect define cost function and benefit function. **Cost function includes** Development cost, Deployment cost and Operation and maintenance cost. Similarly **Benefit function includes** estimated financial value and Goals attained. CBM sometime exclude from their equation cost and benefit that are difficult to calculate and measure. But they are important for computing uncertainty. An advantage of defining utility as weighted sums of goal preferences is that it is extremely easy to apply. Its biggest inconvenience, however, is that the utility scores correspond to no physical characteristics in the application domain making them hard to interpret and impossible to validate empirically. In other words, the utility functions are not falsifiable.

Figure 4:Agile Methodology

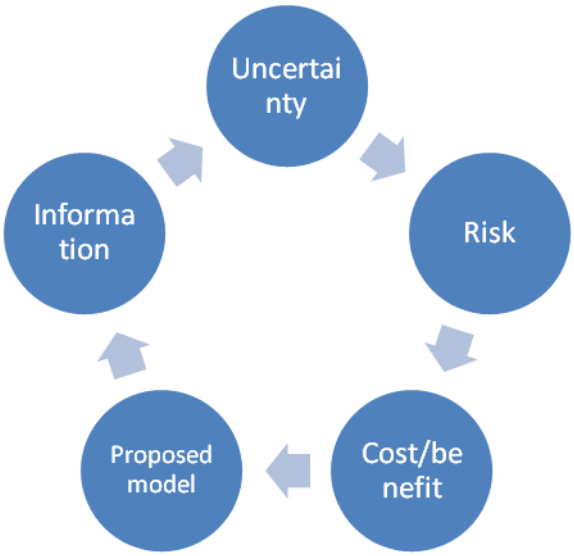

Figure 5: Uncertainty cycle

VII. Conclusion

Requirements and architecture decisions are essentially decisions under uncertainty. We have argued that modeling uncertainty and mathematically analyzing its consequences leads to better decisions than either hiding uncertainty behind point-based estimates or treating uncertainty qualitatively as an inherently uncontrollable aspect of software development. We believe that statistical decision analysis provide the right set of tools to manage uncertainty in complex requirements and architecture decisions. These tools may be useful to other areas of software engineering, e.g. testing, where critical decisions must be made by analyzing risks arising out of incomplete knowledge. In future work, we intend to validate and refine our method on a series of industrial case studies and address the problem of reasoning about model uncertainty.

References

[1] G. Adzic. Impact Mapping: Making a big impact with software products and projects. Provoking Thoughts, 2012.

[2] G. Baio. Bayesian Methods in Health Economics. CRC Press,2012.

[3] B. Flyvbjerg and A. Budzier. Why your IT project may be riskier than you think. Harvard Business Review, 89(9):23–25, 2011.

[4] G. E. Box and N. R. Draper. Empirical model-building and response surfaces. John Wiley & Sons, 1987.

[5] W. Heaven and E. Letier. Simulating and optimising designdecisions in quantitative goal models. In 19th IEEE International Requirements Engineering Conference (RE 2011), pages 79–88. IEEE, 2011.

[6] R. Howard. Information value theory. IEEE Transactions onSystems Science and Cybernetics, 2(1):22–26, 1966.

[7] R. Kazman, J. Asundi, and M. Klein. Quantifying the costs and benefits of architectural decisions. In 23rd International Conference on Software Engineering (ICSE 2001), pages 297–306. IEEE Computer Society, 2001.