International Journal of Information Technology and Knowledge Management
July-December 2009, Volume 2, No. 2, pp. 249-256

RAPID DECISION MAKING FOR POST ARCHITECTURAL CHANGES IN AGILE DEVELOPMENT – A GUIDE TO REDUCE UNCERTAINTY

G. I. U. S. Perera[*] & M. S. D. Fernando[**]

The interest on improving software paradigms, to meet the rapid changing environments becomes a recent and prime research area in Software Engineering. Agile software development was emerged as a result of these studies. Agile practice is a customer oriented, light-weight software development paradigm, suited best for small size development teams in projects under vague and changing requirements. Especially the Agile practice is famous among the open-source communities as it goes easily along with the communities' conventions. Having more flexibility is a better attribute for a process, if it used by a competent experts who can take productive decisions at right moments. However, depending too much on expert knowledge to process and product adjustments is a questionable concern to a growing project with rapid changes to its code base and releases. This research was identified to prevail over this difficulty and provide a necessary guidance to formulate reasonable decisions in the Agile process practice.

Keywords: Software Process Improvement, Agile Process, Decision Uncertainty, Post Architectural Changes.

1. INTRODUCTION

The role of Information Technology (IT) and associated IT workforce is becoming more significant in organizations as information becomes one of the key resources for any activity. The development of the software and computing technologies has changed the world's operations to a large scale. However, software applications are complex and intangible products which are difficult to manage. Hence Software Lifecycle management becomes one of the key research areas in software engineering. Due to the nature of the software, software researchers and practitioners are focused on improving the software processes which are used to develop software. The underline assumption is that there is a direct correlation between the quality of the process and the quality of the developed software [1].

A software process can be considered as a set of tools, methods, and practices we use to produce a software product [2]. These are the prime parameters, also known as Key Process Areas (KPA), that differentiate the process based software development from ad-hoc programming. Identifying KPAs is one of main consideration when a certain process model to be improved [3]. In this research we identified some of the KPAs in agile practice and analyzed them. Specially, the enjoyment of late changes, reliance on key people, and immense flexibility were the driving KPAs of agile practice for this study. Then we examined possible approaches to provide formal systematic decision making guide to improve the agile practice of software development which can be easily utilized without putting much effort to normal agile practice in sustainable manner.

The organization of this paper is as follows. In section 2 we will discuss the background literature in related with this study. In section 3 we discuss the identified problem in brief. Thereafter the section 4 is included with the proposed model and the experiment methodology. The results section 5 describes the importance of the selected result parameters with some sample data items. Section 6, the analysis section explains the analysis done based on the collected results from this study which strongly facilitates the conclusion of this study. Future works; Section 7, and conclusion are thereafter along with the acknowledgements. The references will compile the paper.

2. BACKGROUND

As described early, this study was to evaluate the successfulness of the enhanced agile software development with the introduced new reference model for rapid decision making. To strengthen the study we referred large number of literature on agile process, decision sciences and related areas. Following is a comprehensive synopsis of the literature we used for this study.

A. Agile Software Development

"Agile Methods are a reaction to traditional ways of developing software and acknowledge the need for an alternative to documentation driven, heavyweight software development processes" [4]. In most of the traditional software processes, there are heaps of documents when the project finishes. Most of those are useless at later stages

[*] Department of Computer Science and Engineering, University of Moratuwa, SRI LANKA. *E-mail: indikaperera@uom.lk*

[**] Department of Computer Science and Engineering, University of Moratuwa, SRI LANKA. *E-mail: shantha@uom.lk*

since the requirements were changed several times throughout the project. This was the critical factor for inventing the agile paradigm. By the nature of this paradigm it also provides some other benefits like, flexible project management, cost effective adaptability, increase communication and ultimately increased customer satisfaction.

There are many principles behind the agile practice. Some of them are based on behavioral and managerial improvements to the software development [5]. Despite from those most obvious differences between plan-driven life-cycle models and agile development is that agile models are less document oriented and place more emphasis on code development [6]. The development process is flexible and agile practitioners believe for different projects, different approaches and process models have to be used. Agile process welcomes frequent requirement changes even at late stages of the project. With frequent deliverables, agile process measures its progress through the norm of working software [7].

B. Decision Making for Software Alterations

Decision making models for software process activities mainly facilitate the decision making process in the projects. Herbert Simon can be considered as the father of modern decision sciences. He mentioned that, "Decisions are programmed to the extent that they are repetitive and routine, to the extent that a definite procedure has been worked out for handling them so that they don't have to be treated from scratch each time they occur" [8]. Precisely the main objective of developing this tool was to facilitate the reusability of the knowledge related to decision making in software development. If there isn't any proper tool or framework, then always the practitioners have to their intuition in a situational manner. Situational approaches need lot of processing capabilities. For organizations the most critical task is not to search for or to generate still more information but to filter it so that it does not exceed their processing capacities [9]. Also it needs the expert judgment with a sound background of experiences. Hiring such individuals may not be affordable to every organization. If a Decision Support System was used in place of expert software professional there may be outcomes that is not the best suited to that situation. DSS design still faces an uphill struggle in relation to the design of possible alternatives in decision making as most DSSs treat alternatives as given and unchangeable [10].

3. Problem

Having discussed about the background knowledge we posses, we are about to formulate the research problem for this study. The problem is based on issues of agile practice when it comes to change rapidly at the post architectural stages.

A. Agile Practice Issues

There are many criticisms for agile software process even it address many problems with classical software processes. Heavy reliance on project team members with expert knowledge is the major issue with agile practice [11]. Another major concern with agile development is project management. This involves decision making, resource management, achieving progress, etc. For agile practices there isn't any defined systematic project management approach. As the name suggests all the activities are agile and flexible hence have high vulnerability to become chaotic if not properly handled. In terms of scalability agile practices are usually applied to projects with smaller teams of ten or fewer people [12]. Another fact is that even though the agile process focuses and highlights the out come of the project to the best form, it does not provide necessary well defined activity framework to get that achieved. It enjoys a situational approach for the project progress, which in some point of view, becomes the main bottleneck for the optimization of the process practice [13].

B. Research Problem Scenario

The selected project environment for study was an ideal system with several key characteristics relevant to the study. The University of Moratuwa e-Learning framework; the LearnOrg-MOODLE system is an ongoing project with in house development of the Learning Management System (LMS); the LearnOrg and customization of the open source system MOODLE as the Content Management System (CMS) [14]. MOODLE stands for Modular Object Oriented Distributed Learning Environment. In fact the LMS is the one with many developments as it continually evolve with frequent releases making it a typical open source agile project solution. Actually the academia and students frequently request changes to the system in different capacities. Not only the routine changes but also some government regulation changes needed to be accommodated in the university environment. To make the scenario worse, most of the users do not really understand the operational policies and interpret differently from time to time, diluting the strength of gathered requirements.

With a preliminary research study what we identify is that most of the time developers had to think much about their approach for the new requirement change. Some of the changes were routine, yet took larger portion of time to decide what to do and where to start. That was the main step stone for this study to formulate a sustainable model to guide developers to take rapid decisions under uncertain and ambiguous situations with requirements changes.

4. Methodology

To overcome the above said problems, a simple yet comprehensive reference model, which described below was

proposed. Then it was tested with the research data collection which was based on actual monitoring of the selected agile project. In order to preserve the fairness between experiment samples as much as possible, we used the e-Learning system framework of the University of Moratuwa as mentioned above. The development project includes set of developers who follows the agile process for the system development. In fact the significant factor with the experiment sample was the frequent requirement changes with ambiguous scopes to the changes.

A. The Model: Basis

The Model we proposed in this research which is called Requirement to Cost Matrix (R-C Matrix) is a set of possible decision types based on different combinations that one could find when a application requirement changes at the later stages of development. The underline rational of the proposed model is the relative importance of a given requirement and the total cost (developer time and impact to the system architecture). In fact what we propose here is already been practiced in the industry but merely with expert knowledge dependant. With the formation of a formal reference developer with any skill level could easily identify what the next step is without much uncertainty.

Value Engineering is sometimes taught within the Project Management or industrial engineering body of knowledge as a technique in which the value of a system's outputs is optimized by crafting a mix of performance (Function) and costs. Value Engineering is above all a rationale decision-making/thinking methodology, with a very specific uniqueness [15]. In this case a cost benefit analysis will be done regard to the possible alternatives and their respective costs. This was the main basis of theory for this matrix development.

B. The Model: Considerations

First when considering the requirement critical level; it can be divided in to 3 sections as Low, Medium and High. At the initial point another two extreme categories were included as Very Low and Very High, but later no significant difference was observed between these two and the Low and High categories respectively. On the other hand one could even further classify many requirement critical levels as he wish. Therefore, having 3 general levels were sufficient enough for this model.

When the cost factor was considered, it was identified that having an accurate cost values do not required for the model. Additionally, the traditional costing methods had become invalid relative to how the products and services consumed costs. Therefore the unfavorable impacts of the costing errors were becoming much more intense than in the past [16]. In this respect it is understandable that putting much effort to calculate accurate cost value is worthless if that level of accuracy is not required. Since this tool is only to give a quick and abstract decision about the possible alternatives, such kind of an accurate value is not required for the costing. Furthermore, for the matrix, what required is, an understanding of the cost level; i.e. whether that is a high or a low value. Another plus point is, the model allows the practitioners to use their own comfortable costing mechanisms to get this cost level. However, since the model is best suited for post development alterations, we recommend the Post Architectural Costing Method in the COCOMO2 model [17].

Top down or Bottom up approaches for software system analysis, divide a system in to different levels according to the granularity of the selected component. In this model a class/method/procedure was taken as the smallest level and the entire system architecture as the largest level. Between them an intermediate level was considered to cover packages/modules/sub systems. For this model the requirement critical level was defined from the system perspective. *i.e.* the model considers only about the critical level of the requirement to the existing system's stability. This does not consider the clients perspective on requirement prioritization or the requirement urgency for the associated business. Because of that, a relationship was defined between the system components and the considered requirement important levels. Then those were related with the selected most suitable industrial system development activities to form the options in the matrix.

The other aspect considered was the suitable behavioral best techniques for the project management. When the cost becomes higher to implement the required changes, rather doing it as it is, it is better to try some other possible alternative practices. That is simply to minimize the cost, and cost can be any thing related to the required resources for the selected improvements.

With those considerations, the following model shown in Table 1, was developed to facilitate the decision making.

C. The Model: R-C Matrix

Table 1
R-C Matrix as the Proposed Decision Making Guide

Cost to Implement	*Requirement Importance (R)*		
(C)	*Low*	*Medium*	*High*
Low	Dirty Fix	Module / Sub System Level Change (Clean Fix)	Architectural Level Change
High	Negotiate	Outsource (Split to many small tasks)	Split & Delegate

Let's consider the low cost options first.

Dirty Fix ($R_L C_L$)–is a solution that only focuses on the final result but not the consequences or harm to the rest of the system. For an example, if it is a hard coded solution; though it is not recommended sometimes becomes the best solution, ex. fixing an urgent error in a live system.

Module/ Sub System Level Change (Clean Fix) ($R_M C_L$)–this considers the changes within the respective module of the system without affecting the outside. Since the Clean Fix, it thoroughly analyses the changes and the consequences to the system.

Architectural Level Change (Clean Fix) ($R_H C_L$)–high critical requirements mostly affect to the entire system by their nature. Thus, an architectural level Clean Fix is needed to implement the solution.

The following three options are suitable when the cost becomes higher.

Negotiate ($R_L C_H$)–for a low valued requirement you may need to spend a larger cost. It is better to negotiate with the client for a more affluent state and even if it is possible since it is not a critical requirement.

Outsource ($R_M C_H$)–within the project team, implementing the selected solution is expensive. It is better to get the work done from a third party if their cost of implementation is lower than yours. This is suitable since the requirements are not so critical; hence the risk of outsourcing is minimized.

Split & Delegate ($R_H C_H$)–if the cost is high and the requirement is critical, it needs extreme careful consideration for a successful solution. Therefore we suggest splitting the main requirement into small tasks without any changes, and then delegating them as independent tasks; hence allow selecting other options.

Those six options may not seem as new ones where many have used them in different occasions. However, the important thing of the model is the proper collection and appropriate placing of those options for a formal reference. Especially it drives the developers to view the new requirements alterations systematically and rationally.

D. Research Hypothesis

To examine the proposed model for its appropriateness for real time usage we decided to follow the hypotheses testing with experimental data analysis. Considered research hypothesis for this experiment is as follows.

Null Hypothesis (H_0): Agile software process's development cannot be improved at Post-Architectural stage using R-C Matrix.

Alternative Hypothesis (H_1): Agile software process's development can be improved at Post-Architectural stage using R-C Matrix.

In order to evaluate this hypothesis we selected the statistical analysis of colleted data with reasonable amount of sample size.

E. Experiment Method

As mentioned in the problem section this study aimed to examine the success of applying the R-C Matrix to improve the agile method activities. The LearnOrg-MOODLE learning framework development for requirement changes was chosen as the environment of this experiment. In this scenario the experiment and the controlled experiment were conducted within the same project development. The entire experiment was conducted for 16 weeks by monitoring the real system development workloads.

Without indicating the experiment details the required information were collected for 8 weeks at the beginning (The Controlled Experiment). At that time none of the developers were aware on this experiment.

After that, the R-C matrix was introduced to them and they were been asked to practice the new model for about 4 weeks. Once they were familiar with the model, the next phase of the experiment started, again without their knowledge about the tracking of their data.

The same information was collected for that 8 weeks but giving the developers to practice the R-C matrix for their decision making activities in the system alteration works (The Experiment). Because of the order of the experiment and the controlled experiment, the interference between the two, had a minimum value.

5. RESULTS

One important obstacle that faced when deciding what types of measures to be taken was that identifying most suitable performance measures to track the project enhancements and the progress in this context. Software development processes and paradigms have their own performance parameters and their matrixes. However, since this is a study of evaluating certain improved software process activities, we had to focus on performance parameters and matrixes which are defined specially for the time and effectiveness of the decisions taken.

A. Measurement Parameters

For the performance parameters following measures based on the experiment objectives were selected. Actually the prime objective of developing such matrix was to facilitate quick strategic decisions to cater the frequent requirement changes. In that way for an experiment which has the intention of evaluating the feasibility of such matrix, should

be based on the time measurements for its usage and without usage. Measured parameters are as follows.

N–Number of requirement alterations emerged during the time periods.

Number of requirement changes emerged in the time period is important since the statistical comparison need to be done.

T_0–Time taken to formulate the decision about the solution for a requirement change.

This time value was the most important measurement since it shows the time taken to take the decision about the change implementation.

T_1–Time taken to design the complete solution for the requirement change.

This time value was considered to get an idea about the total implementation time through an intermediate measure.

T_2–Time taken to implement the complete solution for the requirement change.

Since the final comparison between the two samples were based on the time ratio between the time to formulate the solution and the total implementation of the solution *i.e.* T_0 / T_2, this measure is important.

B. Sample Data

Table 2
Selected Functional Implementations without the R-C Matrix

Requirement Name	T_0 *(hrs)*	T_1 *(hrs)*	T_2 *(hrs)*
Multiple academic year support	28	20	65
Enhanced authorization (for login bypass)	20	12	45
Clash table time out	2.5	2.5	6
Auditing for enrollments	15	16	54
Auditing for offerings	15	15	40
Editing disabled - name with initials	1	1	2.5
Fixed clash table conditions	0.75	0.5	2
Fixed user privilege level problems	8	8	24
AJAX supportability	25	12	50
Extended administrators functions	12	10	40

Table 3
Selected Functional Implementations with the R-C Matrix

Requirement Name	*Category*	T_0 *(hrs)*	T_1 *(hrs)*	T_2 *(hrs)*
Direct users to New MOODLE	Module level	0.15	0.5	1.75
Personal notices	Architectural	0.75	1.5	12
Attendance sheet support	Module level	0.5	3.75	7
Alteration in student registration	Dirty Fix	0.35	2	3.25
Course summary	Split & Delegate	0.1	1	2
Course summary - filter by degree	Module level	0.5	3.5	10
Course summary - New fields added	Module level	0.5	1.5	8
Field selection facility	Out sourced	3.5	30	55
Enrolling a student by Admin	Negotiate/Revise	0.1	0.5	1.5
LDAP Authentication	Split & Delegate	1.5	2.5	4.5
FAQ page improvements	Dirty Fix	0.5	1.25	4

Table 2 and Table 3 above show some of the selected measurements from the experiments. For the first phase of the experiment *i.e.* the controlled experiment without R-C Matrix, we collected 21 data records. That is the sample 1; Sample size $N_1 = 21$.

For the second phase of the experiment, *i.e.* the experiment with R-C Matrix, we collected 24 data records. That is the sample 2; Sample size $N_2 = 24$.

6. ANALYSIS

Statistical techniques have been used to perform the hypothesis tests based on the collected data. To compare the samples and to test the hypothesis, Analysis of Variance (ANOVA) is a suitable statistical technique for this type of experiment. Therefore we used ANOVA method to analyze the collected data. As the statistical tool we used Minitab (Release13.20) to automate the tests.

A. Fairness of Data

A simple ANOVA test was conducted using the T_1 values to check the two samples variation. Here the T_1 values were selected because both the T_0 and T_2 were the prime parameters for the hypothesis testing in the experiment. To evaluate the fairness of the data T_1 is the most suitable parameter since the design phases activities are not affected by the R-C tool. And the T_1 is an unbiased parameter for both samples. The Minitab output is shown in the Figure 1 below.

One-way ANOVA: C1, C2

```
Analysis of Variance
Source     DF       SS       MS       F       P
Factor      1    112.4    112.4    3.06   0.087
Error      43   1578.1     36.7
Total      44   1690.5

                                  Individual 95% CIs For Mean
                                  Based on Pooled StDev
Level      N     Mean    StDev  --+---------+---------+---------+----
C1        21    7.857    5.263              (---------*----------)
C2        24    4.690    6.673   (---------*---------)
                                  --+---------+---------+---------+----
Pooled StDev =    6.058           2.5       5.0       7.5      10.0
```

Figure 1: MINITAB Output–ANOVA Results for the Design Phase Times

According to the above results, as with previous ANOVA tests, the *p*-value can be found as 0.087 which is $p > 0.05$. In that case, the null condition has to be accepted. Hence the two samples are similar with each other and contain reasonable fairness for the experiment.

B. Hypothesis Test

To conduct the hypothesis test we had to formulate a mathematical model/expression to represent the samples. Indeed it was a derivation from the collected data parameters. Since the analysis was based on time reduction level for the decision taking the following ratio was used to derive the required information for the hypothesis testing.

Time to formulate the decision T_0

Time to implement the complete solution T_2

Requirements emerged were not identical and therefore the total time to implement the solution (T_2) could not be considered as identical. On the other hand the decision times showed a relation with the total time. But if the ratio T_0/T_2 was considered that will normalize the differences between the values. The output of the hypothesis testing from Minitab is shown in Figure 2.

One-way ANOVA: C1, C2

```
Analysis of Variance
Source     DF       SS       MS       F       P
Factor      1  0.89709  0.89709  264.50   0.001
Error      43  0.14584  0.00339
Total      44  1.04293

                                  Individual 95% CIs For Mean
                                  Based on Pooled StDev
Level      N     Mean    StDev  ---+---------+---------+---------+---
C1        24  0.09758  0.06102  (--*-)
C2        21  0.38060  0.05486                              (--*--)
                                  ---+---------+---------+---------+---
Pooled StDev =  0.05824           0.10      0.20      0.30      0.40
```

Figure 2: Minitab Output–ANOVA Results for Hypothesis Testing using T_0 / T_2

According to the obtained results from the ANOVA test following information was available for the analysis.

For the sample without R-C matrix usage (Sample 1):

Sample size $N1 = 21$, the mean value $\mu 1 = 0.381$ and the standard deviation $\sigma 1 = 0.055$.

For the sample with R-C matrix usage (Sample 2):

Sample size $N2 = 24$, the mean value $\mu 2 = 0.098$ and the standard deviation $\sigma 2 = 0.061$.

According to the ratio, having a lesser value is better. With the obtained mean values, the second sample (sample that used the R-C matrix) is better than the other one, since the time taken to formulate the decision is nearly 10% time of the total time. In the other sample 38% of the time was

spent to formulate the decision. However, to analyze statistically, the *p*-value has to be considered as in previous cases.

With the ANOVA test obtained *p*-value was 0.001; *i.e.* $p < 0.05$ and therefore the Null hypothesis ($H0$) can be rejected with 95% confidence level based on the time portion of the decision making. This implies that the agile process software development decision making can be improved by using the proposed R-C matrix.

C. Experimental Limitations

In this experiment, there were some types of experimental limitations which are worth to mention. Since this study was involved with human activities, this had the experimental limitation of different skill levels between the participants. However this was far better to other scenarios, since the same developers were involved in the in the two samples. Therefore the skill variation was very minimal when the total times were considered for the two samples.

The second and the most crucial limitation with this experiment was the difference between the requirement changes. Some requirements were very simple and some were not. Though it was really hard to eliminate this, what has tried to reduce the difference as much as possible, was the time ratio consideration for the main analysis. By considering the ratios the data samples were between a reasonable data range for the analysis.

Another limitation with the study was the truncation errors of the collected data. Literally, what happened was, the developers were confident on expressing their values with integer figures of hours or in minutes over the decimal or fractional values. For an example they might have said their actual work amount as 0.15 hours, but the precise value may be 10 minutes. In these kinds of extreme cases we used additional parameters like compile time and code base log files to cross validate the figures and minimal truncation. However since this is common to both samples of the experiment this was nullified at the end. Furthermore, this type of truncation errors have normal distribution behavior where the standard error mean is 0; *i.e.* the impact at population level is insignificant.

7.0 Future Work

Despite from the outcome from this research, there are some possible future studies relevant to this research. The experiment mainly focused on the post architectural application development phase of the project life cycle. There are other important life cycle phases in the agile software paradigm like Requirement Engineering, System Design, System Implementation, etc. which have to be examined for similar kind of improvements through the different models and guide tools to improve relevant decision making.

Domain specific customization of the proposed model and improvements is an imperative future research area. That will open up the paths to improvements to address the wide range of agile practitioners across the software industry. This should be studied properly; yet another future research area relate to this study.

Due to the limited resources the study was conducted in a selected environment. Though the results prove impressive observations for the proposed improvements, we expect, thus encourage the other scholars to conduct further researches based on the model with the industrial constraints. Consequently, we welcome any form of valid customization to the model for its betterment, by making it an open-model.

CONCLUSION

In this research we tried to introduce a reference guide for agile practitioners to formulate rapid decisions on their system changes and improvements under uncertain requirements. In that respect we referred much literature to develop a rational basis for our model. Then we examined possible improvements through the introduced model; R-C Matrix, and analyzed the collected data. Statistically we have seen that our argument of following the R-C Matrix can be used to improve the agile paradigm, is correct. Moreover since the objective of this study was not to invent any new software development paradigm but to improve the agile method, it is reasonable to say that the introduced model can be used without any additional effort and cost to the normal agile practice based on our observations in this study. As mentioned the R-C matrix is not a complete new invention. It is a collection of best practices available in the software engineering which were arranged properly in a model to use them effectively. Especially, since the Agile process welcomes frequent requirement changes even at the later stages of the project, critical requirement changing decisions have to be taken rapidly may be even just before the deployment of the project. In such situations so far it mainly depends on the expert project members. Even the experts sometimes take decisions by analogy. But with the aid of R-C matrix all those problems can be solved. The tool was developed by considering many aspects in the software alteration decision making which are visible in the field. Importantly another vital outcome we observed after the research was that the developers who used to practice the model were competent enough to use the model appropriately within very short time periods to formulate a solution. Which is indeed helped them to have a paradigm shift in their thinking process from their own words. With all this regard we believe that, this study will create a significant paradigm shift to the agile software development process. In conclusion, it is evident that this research is one of the significant achievements in the software engineering field. Extensive usage of this study's recommendations will ensure the benefits to the agile practitioners, thus assist to

raise the quality of living of the software stakeholders as the ultimate outcome.

ACKNOWLEDGMENT

We would like to thank to the University of Moratuwa, E-Learning project development team for their support to this research by allowing their project work to consider as the test samples. Also the people who helped for this study at the Department of Computer Science and Engineering, University of Moratuwa, for their support to make this research a success.

REFERENCE

[1] A. Fuggetta, Software Process: A Roadmap, in Proc. of the Conference on The Future of Software Engineering, ICSE, *Limerick,* (2000) 25-34.

[2] W. S. Humphrey, Managing the Software Process, SEI, *Pearson Education,* India, (2006) 03.

[3] R. Fatina, Practical Software Process Improvement, *Artech House,* Boston, (2005) 6.

[4] D. Cohen, M. Lindvall, P. Costa, A State of the Art Report: Agile Software Development, Data and Analysis Center for Software 775 *Daedalian Dr. Rome,* New York 13441-4909, (2003) 1.

[5] G. I. U. S. Perera, M. S. D. Fernando, *Bridging the Gap*– Business and Information Systems: A Roadmap, in Proc. of 4th ICBM Conference, (2007) 334-343.

[6] A. Dagnino, *An Evolutionary Life-Cycle Model* with Agile Practices for Software Development at ABB. In Proc. on 8th IEEE Conference on Engineering of Complex Computer Systems (ICECCS'02), (2002) 215–223.

[7] K. Beck, *et al.,* Manifesto for Agile Software Development, 2001, Available at *http://agilemanifesto.org/,* accessed on (2006).

[8] H. A. Simon, The New Science of Management Decision (3rd Revised edition; First edition 1960) *Prentice-Hall*, *Englewood Cliffs*, NJ, (1977) 46.

[9] H. A. Simon, The New Science of Management Decision (3rd Revised edition; First edition 1960*) Prentice-Hall*, *Englewood Cliffs*, NJ, (1977) 108.

[10] Pomerol J. Ch. et Adam F., Multi-Level and Multi-Model DSSs, in DSS in the Uncertainty of the Internet age, T. Bui, H. Sroka, S. Stanek et J. Goluchowski (Eds.), Karol Adamiecki University of Economics Press, *Katowice*, Poland, (2003) 333-346.

[11] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, M. Zelkowitz, *Empirical Findings in Agile Methods,* in Proc. of Extreme Programming and Agile Methods, (2002) 197-207.

[12] F. P. Deek, J. A. M. McHugh, O. M. Eljabiri, *Strategic Software Engineering* an Interdisciplinary Approach, Auerbach Publications, FL, (2005) 94.

[13] G. I. U. S. Perera, M. S. D. Fernando, Enhanced Agile Software Development–Hybrid Paradigm with *LEAN Practice,* in Proc. of 2nd IEEE Conference, ICIIS, Peradeniya, (2007) 239-244.

[14] G. I. U. S. Perera, Key Success Factors for *e-Learning Acceptability:* A Case Based Analysis on Blended Learning End-User Experience, In Proc. of IEEE International Advanced Computing Conference IACC09, Panjab, (2009) 2379-2384.

[15] Chang, T. H., *Making Value Engineering* as An Effective Tool for Project Integration Using Constructability and Partnering Concepts, in Proc. of SAVE International Conference, (1997) 111-118.

[16] G. Cokins, *"Can Anyone Connect Value and Cost?"* in Proc. of SAVE International Conference, (2000) 17-24.

[17] B. Boehm, C. Abts, A. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, B. Steece, Software Cost Estimation with Cocomo II, *Pearson Education*, India, (2000).