International Journal of Computer Theory and Engineering, Vol. 3, No. 2, April 2011
ISSN: 1793-8201

Comparison and Investigation of Re-planning Methods for Software Releases

Amir Seyed Danesh

Abstract—**Release planning involves decision making on assigning features to sequence releases in incremental software development. Having a good plan for new release can improve the future of software. There are several release planning methods and approaches for release planning and some of them are based on modification and changing the delivered releases or re-planning of the product.**

It means based on old releases and features utilized in them, new releases are generated and companies start to re-plan for improving their old product.

In this paper, we are going to investigate release planning methods that are based on re-planning for a new release. We evaluated and investigated two release planning methods, PARSEQ and Lightweight, and compared the processing models that are based on re-planning.

Index Terms—**Re-planning; requirement engineering; software release**

I. Introduction

Planning and re-planning software projects involves selecting activities according to organisational policies, project goals and contexts, deciding how to affect the activities, and dealing with uncertainty in activity outputs[1].Release planning for incremental software development assigns features to releases in a way that most important technical, resource, risk and budget constraints are met[2].

Therefore, having a good plan for seeing all of the aspects is one of the most important worries in software companies and in software development process. Since environments are different the software conditions and the problems are different too. There are several approaches in solving the problem of release planning. We can improve our release by focusing on different aspects.

Penny [3] proposes is a high level approach to RP in which the estimation of effort required for developing a project features needs a certain confidence level. The planning game in extreme programming is about the same problem in [4]. Ruhe in [5] proposed a model called EVOLVE* in which a synergy between the computational intelligence and human decision maker is combined .

Anton in [6] emphasized that without a plan it is more likely that complex software projects fail.Although there are many methods and approaches in release planning but still there are complexities with the decision making process of selecting features in delivering new release. Therefore having a suitable plan and proper decision for software development is one of the challenges.

Re-planning involves generating a new plan to fix execution failures[7]. Past works are not always complete and sometimes there are some problems and new demands and they need improvement. Sometimes we see from re-investigation of the current plan that some changes are needed because of the new demands and we have to make a new plan.

Software development in large projects needs a series of different releases based on stakeholder's demands and it can be based on re-planning for releases.

So, understanding the concept of re-planning in release is one of the topics that can be discussed further.

Based on [8] two key ideas are of interest in the re-planning effort. The first idea points out that all process activities aim at achieving some desired goals in the project. The second idea is that of prediction uncertainty. One of the most problematic activities within the software engineering area is project planning and it always includes some uncertainty[9]. It means that undoubtedly after developing a new product we need to modify and change to improve it.

In this paper we are going to investigate and analyse the planning methods PARSEQ and Lightweight, for new release in software development and these are based on older release or re-planning how new release can be generated by these two methods.

II. Need for Changes in Software Development

In reactive handling of change requests, the simplest strategy is to freeze change requests [10]. In this approach companies do not accept new requirements until they start new implementation of a release. In fact they keep all of the requirements and once they want to have a new release they implement them. This approach is not acceptable in today's market because some of the demands are urgent and without these changes the product of the system may fail.

Software change is very important because with changes companies can meet many new requirements and this way satisfaction can be targeted. Also systems need to change in response to the new requirements in the market both to survive and be able to compete. A new requirement may include a modification in the system errors with an improvement in its performance or adding new features to the system. It is impossible to produce systems of any size which do not need to be changed. Once software is put into use, new requirements emerge and existing requirements change as the business running that software changes [11].

Therefore, it can be said changes are the main vital parts of

Manuscript received August 30, 2010

software and having a software without changes is impossible. Many studies have been done on the importance of categorizing changes to define processes for each category.

Harker and Eason [12] proposed a classification to distinguish between stable and volatile features (emergent, consequential, mutable, adaptive and migration).

Then, different scenarios were identified, and for each scenario the authors defined: a goal to be achieved, strategy to be adopted, metric to be used and failure mode to be avoided [13].there are a number of different strategies for software change that can be include Software maintenance, Architectural improvements .Changes for a software system not only are important but also essential to be able to compensate for the shortcomings in the older products.

All of this means that, after delivery, software systems always evolve in response to demands for change, and without investigation of new changes the system cannot be complete and to create optimal release the question is what should be changed and when ?

III. Methods for Re-planning Software Releases

Although there are several methods for release planning but here we are going to investigate and understand release methods that are based on re-planning, PARSEQ and Lightweight. This paper is an investigation of the process model of these two methods that are very useful

A. Parseq Method

The idea behind PRSEQ method was first introduced by Karlsson in [14]based on retrospective analysis.A retrospective analysis is a way to look back at events that have occurred and project managers use that to increase the efficiency of projects. The retrospective analysis is acknowledged as an important means for software process improvement [15]. In this method, requirements are analyzed again to select the best solutions for creating more efficient releases in future. This means the aim of the method is to find improvements for the release planning. This method is done by a systematic analysis of requirements in previous releases; that is why the investigation of the older releases is important.

The method uses the requirements database as input and assumes that all the requirements either implemented or not are available in the database. Each step in PARSEQ is divided into four steps [14] as shown in Fig 1.

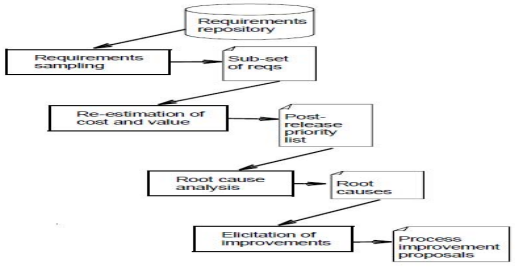

Figure 1: PARSEQ method Activities[14]

Step 1: *Requirements sampling*

Some requirements are selected from the database to be studied. Some of them are selected in the coming release, and the rest will be postponed to the following releases. The purpose of the sampling is to compose a reasonably small but representative sub-set of requirements, since the complete database may be too large to be investigated in the post-release analysis [15].

The requirements sampling can be performed in a number of ways, such as concentrating on a special market segment or on a different part of the product or on particularly difficult decisions [14].

The output of the requirements sampling is the number of requirements that need to be reinvestigated.

Step2: *Re-estimation of cost and value*

In this step, sample requirements taken from the first step are used as the input to estimate their cost and value or the other aspects. By using, for example, a cost-value prioritisation approach, it is possible to see the trade-off of the value to the users and the cost of development in a so called cost-value diagram [16]. To do so, as it was visible in the tool itself, prioritization is redone.

In this stage, there are three techniques used for prioritization (figure 1), AHP, planning game and 100 techniques. For example The planning game is a technique that used to prioritize the work based on three requirements categories (high, medium, and low).

Step3: *Root cause analysis*

The purpose of the root cause analyses is to understand on what grounds release-planning decisions were made [14].In this step, discussing the previous releases and decisions can show which decisions have been right or wrong. Actually, in this step the diagram that is made of selected aspects is analyzed. By discussing different areas of the diagram, we can decide which requirements can be implemented in the first release, which ones should be considered in the next releases, and which ones do not need to be implemented at all. Two questions that can be asked in this step are why these requirements are implemented too early or why they are not implemented yet.

Step4: *Elicitation of improvements*

The output of the root cause analysis step is used in this step for elicitation of improvement proposal. This step is more on understanding the strengths and weaknesses of a selection of requirements for each release.

A number of questions can assist to keep focus on improvement possibilities [14]. First, how we can improve our decision-making. Next question is what is needed to make a better decision. Changes which can be made to the current practices to improve requirements selection in the future is another question to be discussed.

B. Lightweight Method

Lightweight re-planning was first introduced by AlBourae in [17] and emphasized on re-planning by adding new features. In this method they assume they have implemented features and need to add new features to improve their software or product.

In fact, in the process model old features are compared with newly added ones by using Analytical hierarchy process.

In Incremental software development changes are very important and new change requests arrive during the process.

These changes imply the modification of some features or the addition of new ones.

The main goal of the proposed Lightweight Re-plan model is to develop a new product plan that achieves higher stakeholder satisfaction given a limited capacity of time and resources[17]. The Lightweight character reflects the fact that re-planning consumes a considerable amount of the Product Manager's time and effort [18].

Figure 2 shows a lightweight re-plan process model that is describing main release re-planning activities and their input and output.

In this model three main roles are considered, including: *Product manager*, who is responsible for the whole development process; *Stakeholders*, which include any team member who are concerned with the product development; and *the supporting environment*, which facilitates the achievement of the processes goals.

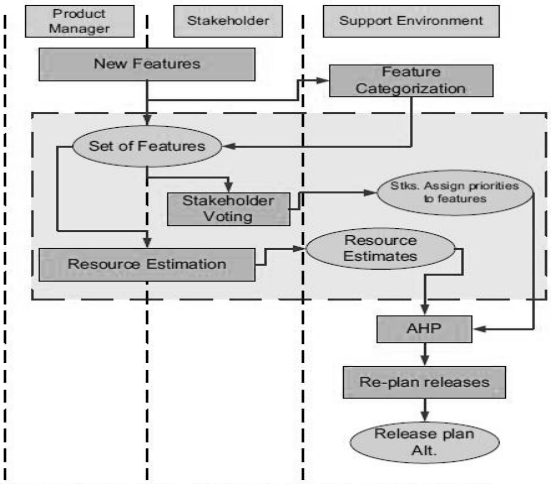

Figure 2: lightweight method Activities[17]

This method includes five steps that are designed as following[17]:

1) New features

When the developments are going to start, the new features must be selected for impemlation. The change requests received are added to the old sets of features and directed to be categorized by the feature categorization process.

A set of features that were assigned F(i)= ($f_1, f_2, .., f_n$)

New added features ΔF (i) = ($f_{n+1}, f_{n+2} ..., f_{n+m}$)

We have to re-schedule the next release to be delivered.

2) Feature categorization:

When a new feature is going to be investigated for next release based on Higgins's et al work [19], change requests should be categorized to avoid duplicated features. So, the needs for categorization of features in this step must be performed.

3) Stakeholder's voting

Stakeholders are people that are effective in development process. They include different levels such as managers, developers or end users. In the previous release planning process, the relative weight of importance of the stakeholders are considered in an objective function to maximize their preferences. Available resource constraints are also observed in this process. In [12] you can learn more about the process and how it is conducted.

4) Resource estimation:

The aim is to determine the likely usage of effort and Time for each feature for the next release and the main goal is to maintain the effort and time available as we re-plan so that the new re-planned release does not exceed the capacity available.

5) The Analytical Hierarchy Process (AHP)

Analytical Hierarchy Process (AHP) is a prioritization method based on [20]. In this step, after elicitation of experts' preferences in a formalized manner, we use a pair-wise comparison technique.

IV. Analysing The Methods

From the studies done on these methods and the analysis of the processing models, first we must know that changes are needed as part of process improvement and when a new release is created in order to consider the changes .

As you have seen before, the frist model of release planing that is based on re-planing was PARSEQ that is based on going back and re-invasgtaion and re-estimation of requimensts. It means this kind of re-planing is based on re-estimation of the database of requiements that are waiting for implementation.

As you see in the figure 3 there are three prioritization approaches was utilized in this method. After selecting the target requirements it needs to be prioritized. For each approach you can select your aspect and for sample value, cost and risk is shown

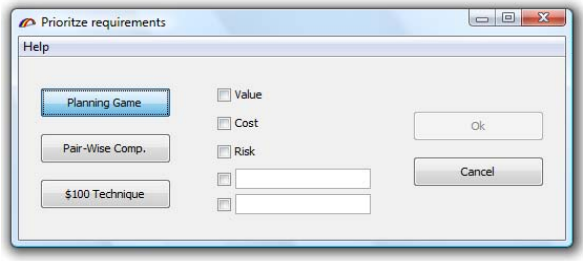

Figure 3: requirements priotization in PARSEQ

After applying the appropriate prioritization approach, the system should come out with a reasonable solution. For example, it would be recommended to implement feasture2 within release 3 as it is not necessary (in that specific time).

In the Lightweight method, the main goal is to study all the system's new features and estimate all these features in order to add them to the final product. However, in this method, we need first to have software with fully implemented features, and we improve our software by adding some new features based on new received requirements. The improvement of the release depends on the stakeholders' satisfaction; therefore we need to investigate all the new requirements based on the stakeholders voting in order to re-estimate the new features .AHP is the only prioritization technique that is used in this method. In this method, time and effort estimation is very important for the re-planning process. On the other hand, the PARSEQ recommended list of solutions that can help software developers to improve their undertaken release. These solutions come by using three different prioritization techniques as mentioned earlier.

V. COLCLUSTION

This paper intended to investigate the current re-planning methods .There are several release planning methods that have been defined to develop better software and improve the quality of the new releases. Some of these methods are based on older releases and decision making of previous releases. The paper aims to compare all these different kinds of methods. In this content, we have reviewed two re-planning methods in details.

REFERENCES

[1] Diana Kirk and S. MacDonell, "A Simulation Framework to Support Software Project (Re)Planning," presented at the 35th Euromicro Conference on Software Engineering and Advanced Applications, Patras 2009.
[2] G. Ruhe, "Software Release Planning," in HANDBOOK OF SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING. vol. 3 ed: Chang, S. K. (Ed.): World Scientific publishing, 2005.
[3] D. A. Penny, "An estimation-based management framework for enhanced maintenance in commercial software products," in In Proceedings ICSM International Conference on Software Maintenance, 2002, pp. 122-130.
[4] B. A. Nejmeh and I. Thomas, "Business-driven product planning using feature vectors and increments," Software, IEEE, vol. 19, pp. 34-42, 2002.
[5] Günther Ruhe and A. Ngo, "Hybrid Intelligence in Software Release Planning," International Journal of Hybrid Intelligent Systems vol. 1, pp. 99-110, 2004.
[6] A. I. Anton, "Successful software projects need requirements planning," Software, IEEE, vol. 20, pp. 44, 46, 2003.
[7] William Cushing and S. Kambhampati, "Replanning: a New Perspective," in In Proc. of ICAPS, 2005.
[8] D. Kirk and S. MacDonell, "A Simulation Framework to Support Software Project (Re)Planning," in Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on, 2009, pp. 285-292.
[9] M. C. Ohlsson and C. Wohlin, "An empirical study of effort estimation during project execution," in Software Metrics Symposium, 1999. Proceedings. Sixth International, 1999, pp. 91-98.
[10] K. E. Wiegers. (2nd ed. 2003). Software requirements : practical techniques for gathering and managing requirements throughout the product development cycle. Redmond, Wash.:Microsoft Press. xix, 516.
[11] I. Sommerville, "Software Change," in Software Engineering, ed: http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/ElectronicSupplements/SWChange.pdf, 2000.
[12] S. D. P. Harker, et al., "The change and evolution of requirements as a challenge to the practice of software engineering," in Requirements Engineering, 1993., Proceedings of IEEE International Symposium on, 1993, pp. 266-272.
[13] W. Lam and V. Shankararaman, "Requirements change: a dissection of management issues," in EUROMICRO Conference, 1999. Proceedings. 25th, 1999, pp. 244-251 vol.2.
[14] Lena Karlsson, et al., "Post-Release Analysis of Requirements Selection Quality - An Industrial Case Study," in 9th Int. Workshop on Requirements Engineering: Foundation for Software Quality,, Velden, Austria, 2003.
[15] L. K. and and B. Regnell, "Introducing tool support for retrospective analysis of release planning decisions," presented at the 7th International Conference on Product Focused Software Process. Improvement (PROFES'06), Netherlands, 2006.
[16] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," Software, IEEE, vol. 14, pp. 67-74, 1997.
[17] T. Albourae, et al., "Lightweight Replanning of Software Product Releases," in Software Product Management, 2006. IWSPM '06. International Workshop on, 2006, pp. 27-34.
[18] Joseph Momoh and G. Ruhe, "Release planning process improvement - an industrial case study," SOFTWARE PROCESS IMPROVEMENT AND PRACTICE, vol. 11, pp. 295-307, 2006.
[19] S. A. Higgins, et al., "Managing requirements for medical IT products," Software, IEEE, vol. 20, pp. 26-33, 2003.
[20] T. L. Saaty, The analytic hierarchy process. New York: McGraw-Hill, 1980.

Amir Seyed Danesh is a PhD student at University of Malaya, Kuala Lumpur, Malaysia since 2008 in the department of Software Engineering. He worked as software engineer for five years in Iran. He received the M.Sc degree from Shahid Beheshti University, Tehran, Iran in Software Engineering in 2006. He is interested in Requirements Engineering, Software Release Planning, methods and Requirements Prioritization. He also was part-time lectured at University of Guilan in 2009, Rasht, Iran.