

Lightweight Replanning of Software Product Releases

Thamer AlBourae
Software Engineering
Decision Support Labs,
University of Calgary,
albourae@cpsc.ucalgary.ca

Guenther Ruhe
Software Engineering
Decision Support Labs,
University of Calgary,
ruhe@ucalgary.ca

Mahmood Moussavi
Schulich School of
Engineering,
University of Calgary,
moussavm@enel.ucalgary.ca

Abstract

Well defined product features are the essence of good product management. High quality features lead to successful software products, both functionally and financially. One of the crucial processes in software product management is release planning where features are assigned to releases. Volatile features, resources and stakeholder preferences have been recognized as factors that decrease release quality. In this paper, we propose a lightweight replanning process model where old features are compared with newly added ones using the Analytical Hierarchy Process (AHP). Then, a greedy replan algorithm is applied to select the most promising features to accommodate changing market driven product demands.

1. Introduction and Motivation

As incremental development gains greater acceptance in the software product development society, the farther we move from the monolithic waterfall model, and the greater the importance of the release planning process becomes. Incremental software product development allows customers to receive deliverables early. This means the business value of the product is recognized before the product is fully delivered. Furthermore, it opens the door for early feedback from clients. This implies the incorporation of the feedback into future product releases to gain higher customer satisfaction.

Features are the basis of software release products. They provide a foundation to plan and re-plan software releases, estimate effort and resources constraints and reflect the customer expectations of the product. Release planning is a process where we make decisions on which features are assigned to which releases. The planning process assigns incremental

components of the software system to be delivered on a specific calendar date. These sub-systems are planned in sub-releases to be delivered incrementally. As change requests arrive, plans need to be adjusted (replanned) for newly added features.

The process of planning or re-planning releases considers several aspects, including effort estimation, resource constraints and stakeholder preferences to gain higher customer satisfaction [1].

As the software product development process begins, a continuous stream of new change requests arrive. This is common in market driven software products [2]. These changes imply the modifications of some features or the addition of new ones. Since not all features can be implemented due to effort and resources boundaries, we have to decide which feature should be delivered and which have to wait. Moreover, we cannot neglect these changes requests since a high level of stakeholder satisfaction is important.

In this paper, we present the foundation for a process model to handle change requests and re-plan software releases. The model pools all available features ready to be allocated to releases. Then, we apply the analytical hierarchy process (AHP) [3] to define a degree of importance for each feature.

The rest of the paper is organized as follows. In Section 2, we present literature models related to the problem. In Section 3, we describe the problem being addressed. In Section 4, we formalize the re-planning problem from a decision support perspective. In Section 5, we perform a case study to demonstrate the contribution of the new approach. In Section 6, we summarize the findings and give an outlook for future work.

2. Related Work

2.1 Release Planning

Release Planning has gained interest in market driven software product development [4], [5]. This selection process has to take into consideration multiple stakeholder preferences and available resources. In addition, features interdependencies [6] must be handled. Feature interdependencies can force a low-priority feature to be implemented before a higher-priority one. This may decrease the client satisfaction and could lead to losing market share. Thus, the decision making process of selecting features and the tradeoffs involved are complex.

Various approaches address the problem of Release Planning (RP) both in industry and academia. Anton emphasized that complex software projects are likely to fail without a plan [7]. Penny [8] proposed a high level approach to RP where the estimation of the total effort required for developing a project features should fall in a certain confidence level. The planning game (GM) in extreme programming tackled the same problem in [9]. Others looked at the problem from an optimization point of view like [10] and [11]. Ruhe *et al* proposed a model (EVOLVE*) where a synergy between the computational intelligence and human decision maker is combined [12].

The replanning further adds to the complexity of the problem. Considering new changes, maintaining the stakeholders' satisfaction and adjusting releases to fit within available resources all have to be accounted for.

The re-planning of a product release is used to improve the process of release planning and increase the quality of the releases [13]. Furthermore, it makes us learn from the previous mistakes we committed in case a retrospective analysis is conducted [14].

Davis's paper [15] addresses a special case of the release planning problem where changes take place and we have to select a new set of requirements for the next release satisfying a given time and resource capacity.

2.2 Change-Based Process Models

Many Change-Based process models have been proposed in the literature. These models include the PRISM process model [16] and the FEAST process model [17]. PRISM is concerned with change propagation from environmental perspective while the FEAST model is more focused on the software process as it changes over time. Other studies proposed a Change Maturity Model based on their work on the

Design Improvements in Requirements Evolution (DESIRE) project [18].

2.3 Change Categorization

Many studies addressed the importance of categorizing changes to define processes for each category. Harker and Eason [19] proposed a classification to distinguish between stable and volatile features (emergent, consequential, mutable, adaptive and migration). Other studies agreed that categorization is a wise step towards better change management [20]. Some literature focused on the problem from a management perspective and categorized changes based on a cause-effect diagram. Then, different scenarios were identified, and for each scenario the authors define: a goal to be achieved, strategy to be adopted, metric to be used and failure mode to be avoided [21].

Some industrial organizations studied the problem of change to define classification schemes. For example, Stark's *et al* studied the problem [22] on 44 releases. Then, depending on the data collected, the study proposed a formula to predict the time needed to implement these changes. Also, the study extracted some guidelines to be considered in the process of release planning. On the same path, Higgins *et al* [23] studied the features change problem to overcome the potential of bottlenecks that can occur due to lack of resources. Also, the study defined a process that handles changes requests as they arrive.

2.4 Other Related Research

Other research provided some guidelines for handling changes [24]. For instance, identifying change early in the product development cycle, providing a process to incorporate changes in the development cycle and reducing the amount of change by investigating its origins, are all considered change handling fundamentals.

Other studies address the concept of incremental evolutionary delivery [25] or tackle the problem of categorizing changes [26].

3. Problem Description

Figure 1 illustrates the process of newly arriving features competing against those already planned for, where some features are already implemented. In the beginning of the product development cycle, a set of features $F(i) = (f_1, f_2, \dots, f_n)$ are assigned to releases to be delivered in a certain sequence. As the development

process starts on t_0 (moment of time), change requests arrive on t_1 with new added features $\Delta F(i) = (f_{n+1}, f_{n+2}, \dots, f_{n+m})$ and we have to re-schedule the next release to be delivered on t_2 . The dashed line represents t_1 , when these changes requests arrive, t_2 when the delivery of the release being developed is shifted to release time t_2 , and the dotted line separates the old and new features.

Figure 1 shows the different categories of features at a given time t_1 . We have identified four main categories of features in this scenario, these include:

- A. Implemented features: features which are already implemented and they are not included in the process of re-planning.
- B. On going features: features which are in the process of being implemented.
- C. Planned for features: features which are planned to be implemented in the future, but have not been started yet.
- D. Added features: features proposed from stakeholders to be added to the next release.

Thus, we can formulate the problem of replanning of the release being developed by defining a function to find a new set of features F' out of the previous set of feature F plus the new added ones ΔF , where: $F' \subset F + \Delta F$.

The new set F' contains the best possible set of features that provide the highest stakeholders satisfaction while having a limited capacity of effort and time available for the next release.

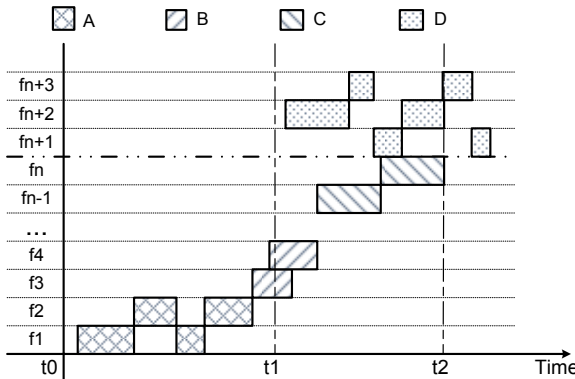


Figure 1: The Re-plan problem

4. Lightweight Replan

4.1 Overview

The main goal of the proposed Lightweight Replan model is to develop a new product plan that achieves higher stakeholder satisfaction given a limited capacity of time and resources. The Lightweight character reflects the fact that replanning consumes a

considerable amount of the Product Manager's time and effort [13]. Moreover, the computation complexity is reduced by handling fewer change requests instantly. This means incorporating changes earlier in the product life cycle where changes cost less. Applying the heavily weighted release planning process proposed in [1] is time consuming and of a high complexity. This would lead to delay incorporating the changes proposed to later phases where implementing these changes will cost more. In this section, we present the Lightweight Replan process model and describe the individual steps of it in details.

Figure 2 shows a generic process model describing main release replanning activities including their inputs and outputs. These activities work together to produce a framework for replanning product releases. It is noteworthy that the activities inside the shaded dashed box are part of the product release planning process [1], while other activities complete the proposed Lightweight Replan model.

In this model, three main roles are recognized – *Product manager*, who is responsible of 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. The supporting environment can be processes or tools that support the process. Major activities are represented in rectangles while outputs of any process are represented in ovals.

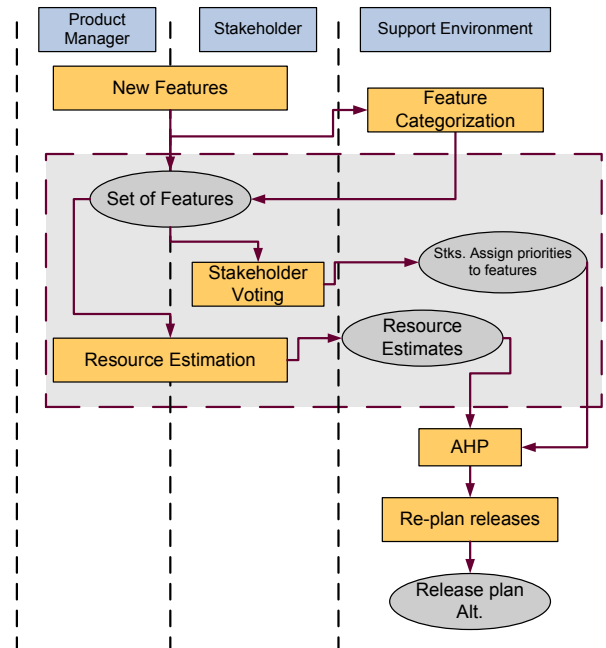


Figure 2: Generic process model for the Lightweight Re-planning

Every activity or output is engaged with the roles appearing at the top, within that activity or outcome's "swim lane". For example, the product manager and other stakeholders contribute to the process of "Resource Estimation" while the supporting environment facilitates the achievement of "Resources Estimates" through a tool. In the next section, we describe each activity or output in detail.

4.2 Main Steps

4.2.1 New Features. As the product development starts, the development team begins receiving new change requests for the features sets. The change requests received are added to the old sets of features and directed to be categorized by the feature categorization process.

4.2.2 Feature Categorization. Adopted from Higgins's et al work [23], change requests should be categorized to distinguish between duplicated features, on-going features or newly added ones. This process also helps assure that we have a complete and consistent set of features.

4.2.3 Stakeholder Voting. Stakeholders $Stk(p) = \{Stk(1), Stk(2), \dots, Stk(q)\}$ are a group of people who are concerned with the software product plan in a direct or indirect way. They may include team members from different organizational levels such as management, development or customer services. In the previous release planning process, an objective function is used to maximize stakeholder's preferences considering their relative weight of importance while observing the available resource constraints. To learn more about the process and how it is conducted, we refer the reader to [12]. Furthermore, stakeholders are required to assign to each new feature a level of attractiveness.

4.2.4 Resource Estimation. Resources capacities are one of the boundaries that should be considered when replanning product releases. The main aim is to determine the likely usage of effort $Effort(f_i)$, and time $Time(f_i)$ for each feature f_i for the next release. Resource estimation is a complex problem on itself and far beyond the scope of our research. Thus, we encourage the reader to refer to [27] for detailed approaches to effort estimation.

In our process model, we need to consider the rate of consumption for implemented and ongoing features when the changes were received at t_1 . Also, we estimate effort and time needed for new features. Other

feature categories estimations and releases capacities are assumed to remain unchanged. Figure 3 describes the effort estimation process for features and the release being developed.

The main goal is to maintain the effort and time available as we replan so that the new replanned release does not exceed the capacity available.

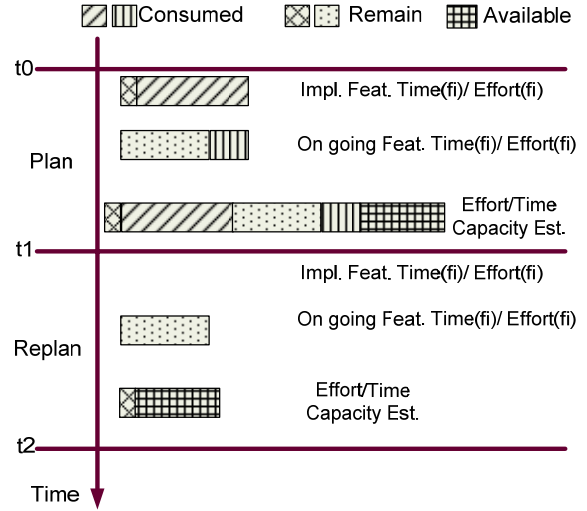


Figure 3: Effort and time estimation over Time (t_0, t_1)

2.5 The Analytical Hierarchy Process (AHP). The Analytical Hierarchy Process (AHP) is a multi criteria decision making method [3]. This process elicits experts' preferences in a formalized manner using a pair-wise comparison technique. Each expert evaluates a pair of features with respect to a defined criteria. Figure 4 shows the pair-wise comparison prioritization process using AHP.

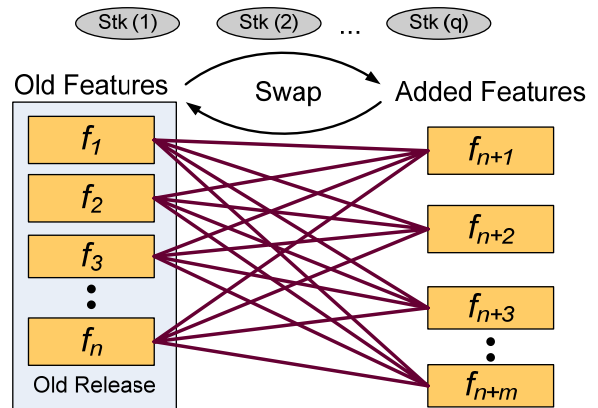


Figure 4: The Analytical Hierarchy Process (AHP) pair wise comparison

Each stakeholder compares between the old set of features F and the newly added features ΔF with

respect to the degree of attraction. The Weighted Average Satisfaction (WAS) was already computed for F during the previous planning process [1]. Assuming that the WAS values remain unchanged, each stakeholder compares a pair of features (an old feature from F with a newly added one from ΔF) to express the level of attraction. In this situation, some old low attractive features are substituted or swapped with newly added ones. Then, the most promising features are gathered in a new set to enter into the Greedy Replan algorithm in the next step.

Figure 5, 6 and 7 illustrates the Expert's Choice tool [28] where features are compared using the AHP technique based on urgency and time.

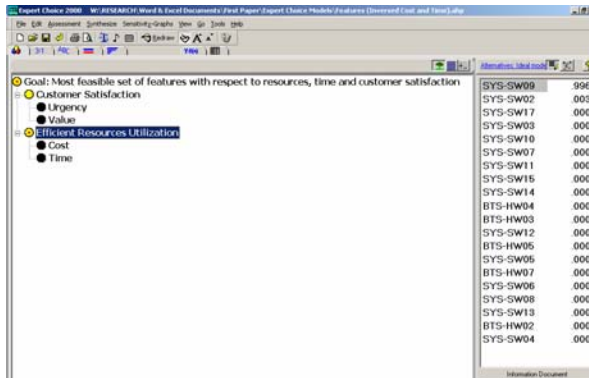


Figure 5: The Analytical Hierarchy Process in the tool

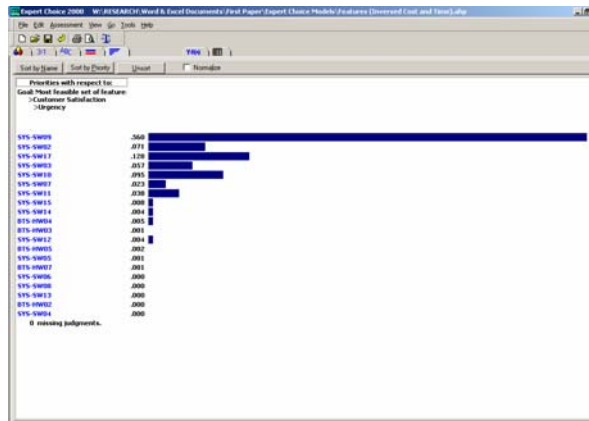


Figure 6: Pair wise comparison based on urgency (time needed to deliver a feature to the market)

4.2.6 Replanning Releases. The information gained from the AHP pair-wise comparison process is used to select a new set of features F' to be assigned to the new (replanned) release.

Selecting the most promising set of features F' to form a replanned release is a combinatorial

optimization problem. This problem is known as a type of Knapsack problem where given a set of features, we try to maximize the Weighted Average Satisfaction (WAS) while maintaining the available effort and time capacity. The algorithm to solve such problems are time consuming and of a high complexity [29].

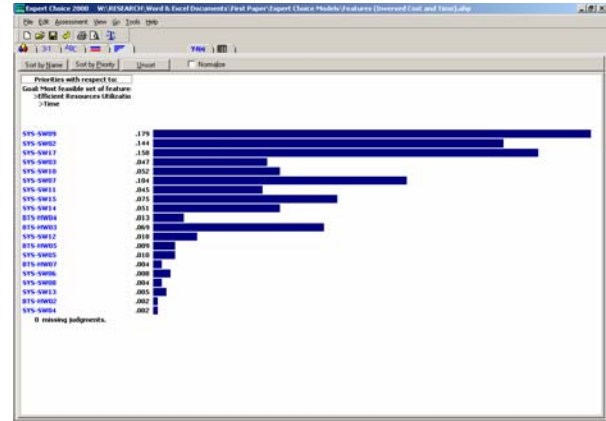


Figure 7: Pair wise comparison based on the time estimation Time (f_i) of each feature

In our process model, we adopt a lightweight greedy algorithm which does not give an optimum solution, yet it gives an approximation of it. The greedy approximation algorithm was proposed by Martello and Toth. For learning more about this algorithm, we refer the reader to [30].

A simple pseudo code of the Greedy Replan algorithm can be expressed as follows:

1. Define an objective Function $F(f_i) = 0$.
2. Define $F' = \emptyset$ and empty set of features.
3. Define Effort = 0, Time = 0 (Effort and Time Capacity).
4. Sort all features with the best attractiveness ratios gained from the AHP comparison in a descending order.
5. While Effort and Time are less than the release capacity C do steps 6 - 9.
6. Add feature f_i to F'
7. $F(f_i) = F(f_i) + WAS(f_i)$
8. Effort = Effort + Effort (f_i)
9. Time = Time + Time (f_i)
10. Return F' , $F(f_i)$, Effort, Time

5. Case Study: Re-planning product releases in a Telecommunication project

To demonstrate the contribution of the proposed process model we conduct a case study in on a project from the telecommunication using a web-based tool called ReleasePlanner[®] [28]. The tool is implemented

based on the hybrid intelligent approach proposed in [12]. The project has 7 stakeholders with different perspectives and degrees of importance.

On t_0 , the project starts with 15 features. At t_1 , the project receives change requests from different stakeholders while the team is still developing release 1. We categorize features as described in Section 4.2.2.

At the time changes arise (t_1), we have 3 implemented features (Feat.), 5 ongoing, 7 planned for, and 4 added. Table 1 shows this categorization with each feature ID.

According to each feature category, we re-estimate the capital and time resources required as described in section 4.2.4.

Table 1: Features categorization with each feature ID

N #	Belong to Set	Feat. Type	Feat.	Feature ID
1	F	Impl.	f_1	BTS-HW01
2			f_2	BTS-SW01
3			f_3	SYS-SW01
4	F	On going	f_4	SYS-SW09
5			f_5	SYS-SW07
6			f_6	SYS-SW06
7			f_7	SYS-SW08
8			f_8	SYS-SW04
9	F	Planned For	f_9	SYS-SW02
10			f_{10}	SYS-SW17
11			f_{11}	SYS-SW03
12			f_{12}	SYS-SW10
13			f_{13}	SYS-SW11
14			f_{14}	BTS-HW07
n=15			f_{15}	BTS-HW02
16	ΔF	Added	f_{n+1}	SYS-SW15
17			f_{n+2}	SYS-SW14
18			f_{n+3}	BTS-HW04
19			f_{n+4}	BTS-HW03

Then, we compare the old set of features (F) with newly added ones (ΔF) using AHP. During the AHP comparison process, we use the Weight Average Satisfaction (WAS) assigned to old features in the previous release planning process and compare it with the level of attraction assigned to the new features (ΔF).

As we apply the Greedy Replan algorithm (described in section 4.2.6) the new (replanned) release is formed. Comparing the new (replanned) release with the old release with respect to the WAS, capital and time usage, we have:

Table 2 shows the Old release compared to the replanned one with respect to the WAS, while Table 3 shows the Old release compared to the replanned one with respect to Capital and Time.

Table 2: The old and new (replanned) releases structures

Old Release		Replanned Release	
Feat.	WAS	Feat.	WAS
f_1	--	--	--
f_2	--	--	--
f_3	--	--	--
f_4	616	--	--
f_5	1632	f_5	1632
f_6	2004	f_6	2004
f_7	654	--	--
f_8	749	--	--
f_9	1485	f_9	1485
f_{10}	1325	f_{10}	1325
f_{11}	1296	f_{11}	1296
f_{12}	723	--	--
f_{13}	305	--	--
f_{14}	967	--	--
f_{15}	1114	f_{15}	1114
--	--	f_{n+1}	1962
--	--	f_{n+2}	1911
--	--	f_{n+4}	1593
Sum	12870	14322	

Table 3: The old and new (replanned) releases Capital and Time usage

Old Release			Replanned Release		
Feat.	Capital	Time	Feat.	Capital	Time
f_1	--	--	--	--	--
f_2	--	--	--	--	--
f_3	--	--	--	--	--
f_4	0	800	--	--	--
f_5	160	432	f_5	160	432
f_6	400	560	f_6	400	560
f_7	400	856	--	--	--
f_8	800	980	--	--	--
f_9	25	1150	f_9	25	1150
f_{10}	50	830	f_{10}	50	830
f_{11}	100	2200	f_{11}	100	2200
f_{12}	300	1600	--	--	--
f_{13}	500	1150	--	--	--
f_{14}	750	2045	--	--	--
f_{15}	1500	2130	f_{15}	1500	2130
--	--	--	f_{n+1}	0	550
--	--	--	f_{n+2}	50	650
--	--	--	f_{n+4}	200	300
Sum	4985	14733		2485	8802

It is obvious that the new replanned release gained a higher WAS compared to the old release. Moreover, the replanned release requires less capital and time in order to be delivered. However, the number of features included in the replanned release is less than the old release.

It is obvious that the old release had more features planned for implementation. However, our new (replanned) release has a higher Stakeholder satisfaction level with lower resource usage.

We refer the reader to our electronic version of the empirical case study for more details [31].

6. Conclusions and Discussion

The lightweight replanning process model provides a basis for incorporating changes instantly into the development lifecycle. The lightweight factor is beneficial since it will reduce the cost of introducing changes in the development cycle.

The light weight Replanning process model can be applied instantly to incorporate changes in the development cycle as soon as possible. This is recognized as one of the fundamentals used for handling changes [24].

Integrating the process of categorization of changes facilitates the re-planning process phases where we select features with higher value while not exceeding effort capacity available.

The model is of a low computation complexity compared with the release planning process proposed in [12]. This is considered good where the replanning must be achieved quickly especially in dynamic markets. Furthermore, light weight replanning enables performing the process at any time. This is considered to be common where the rate of change is often high [22] particularly in market driven products [2].

Some open issues to be explored include validating the lightweight replanning approach empirically. Also, deciding when we should replan and when we should not, possibly by defining a threshold, is an issue that needs to be explored. Another related issue is discerning how frequently the replanning process should take place. Another area of future work involves integrating an impact analysis process to quantify the change effects on the existing implemented features.

The Lightweight Replan model best fits products that are market driven where maintaining a higher stakeholder satisfaction is desired. However, the model and the presented results are limited to the data and the empirical study we conducted so far. Thus, there is a need for real world industrial experiments to validate the model and the results.

Acknowledgement

We thank the Ministry of Higher Education of Saudi Arabia and Alberta Informatics Circle of

Research Excellence (iCORE) for their financial support of this research. We also thank Jim McElroy and the anonymous reviewers for their detailed and valuable comments.

References

- [1] G. Ruhe and M. O. Saliu, "The art and science of software release planning," *IEEE Software*, vol. 22, pp. 47-53, 2005.
- [2] P. Carlshamre and B. Regnell, "Requirements lifecycle management and release planning in market-driven requirements engineering processes," In Proceedings IEEE International Workshop on the Requirements Engineering Process: Innovative Techniques, Models, and Tools to support the RE Process, pp. 961-5, Greenwich, UK, 2000.
- [3] T. L. Saaty, *The analytic hierarchy process*. New York: McGraw-Hill, 1980.
- [4] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *Software, IEEE*, vol. 14, pp. 67-74, 1997.
- [5] J. Karlsson, S. Olsson, and K. Ryan, "Improved practical support for large-scale requirements prioritising," *Requirements Engineering*, vol. 2, pp. 51-60, 1997.
- [6] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, and J. Natt och Dag, "An industrial survey of requirements interdependencies in software product release planning," In Proceedings 5th IEEE International Symposium on Requirements Engineering, pp. 84-91, Toronto, Ont, 2001.
- [7] A. I. Anton, "Successful software projects need requirements planning," *Software, IEEE*, vol. 20, pp. 44, 46, 2003.
- [8] D. A. Penny, "An estimation-based management framework for enhanceive maintenance in commercial software products," In Proceedings ICSM International Conference on Software Maintenance, pp. 122-130, 2002.
- [9] B. A. Nejme and I. Thomas, "Business-driven product planning using feature vectors and increments," *Software, IEEE*, vol. 19, pp. 34-42, 2002.
- [10] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittle, "The next release problem," *Information and Software Technology*, vol. 43, pp. 883-90, 2001.

- [11] J. Ho-Won, "Optimizing value and cost in requirements analysis," *Software, IEEE*, vol. 15, pp. 74-78, 1998.
- [12] G. Ruhe and A. Ngo-The, "Hybrid Intelligence in Software Release Planning," *International Journal of Hybrid Intelligent Systems*, vol. 1, pp. 99-110, 2004.
- [13] J. Momoh and G. Ruhe, "Release Planning Process Improvement - An Industrial Case Study," *Software Process: Improvement and Practice*, vol. 11, pp. 295-307, 2006
- [14] L. Karlsson, B. Regnell, and T. Thelin, "Case Studies in Process Improvement through Retrospective Analysis of Release Planning Decisions " *International Journal of Software Engineering and Knowledge Engineering*, June, 2006 (in press).
- [15] A. M. Davis, "The art of requirements triage," *Computer*, vol. 36, pp. 42-49, 2003.
- [16] N. H. Madhavji, "Environment evolution: the Prism model of changes," *Software Engineering, IEEE Transactions on*, vol. 18, pp. 380-392, 1992.
- [17] M. M. Lehman, "Feedback in the software evolution process," *Information and Software Technology*, vol. 38, pp. 681-6, 1996.
- [18] W. Lam and V. Shankararaman, "Managing change in software development using a process improvement approach," In Proceedings 24th EUROMICRO Conference, pp. 779-786, Vasteras, Sweden, 1998.
- [19] S. D. P. Harker, K. D. Eason, and J. E. Dobson, "The change and evolution of requirements as a challenge to the practice of software engineering," In Proceedings IEEE International Symposium on Requirements Engineering pp. 266-272, San Diego, CA, USA, 1992.
- [20] D. Rowe, J. Leaney, and D. Lowe, "Defining Systems Evolvability - a Taxonomy of change," In Proceedings International Conference and Workshop on Engineering Computer Based Systems, pp. 45-52, Jerusalem, Israel, 1998.
- [21] W. Lam and V. Shankararaman, "Requirements change: a dissection of management issues," In Proceedings 25th EUROMICRO Conference. Informatics: Theory and Practice for the New Millennium, pp. 244-251, Milan, Italy, 1999.
- [22] G. Stark, A. Skillicorn, and R. Ameele, "An Examination of the Effects of Requirements Changes on Software Maintenance Releases," *Journal of Software Maintenance: Research and Practice*, vol. 11, pp. 293-309, 1999.
- [23] S. A. Higgins, M. De Laat, P. M. C. Gieles, and E. M. Geurts, "Managing requirements for medical IT products," *IEEE Software*, vol. 20, pp. 26-33, 2003.
- [24] R. C. Sugden and M. R. Strens, "Strategies, tactics and methods for handling change," In Proceedings IEEE Symposium and Workshop on Engineering of Computer-Based Systems, pp. 457-463, Friedrichshafen, Germany, 1996.
- [25] T. Gilb, *Principles of Software Engineering Management* Addison-Wesley, England, 1988.
- [26] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*: John Wiley & Sons Ltd, 1998.
- [27] L. C. Briand and I. Wieczorek, "Resource Estimation in Software Engineering " in: *Marciniak JJ (ed) Encyclopedia of software engineering (2nd edition)*. New York: John Wiley, 2005.
- [28] <http://www.expertchoice.com/>.
- [29] A. Ngo-The and G. Ruhe, "Optimized Resource Allocation for Software Release Planning," *Software Engineering, IEEE Transactions on*, 2005 (Submitted).
- [30] M. R. Garey and D. S. Johnson, *Computers and intractability : a guide to the theory of NP-completeness*. San Francisco: W. H. Freeman, 1979.
- [31] T. AlBourae. Calgary, AB: <http://sern.ucalgary.ca/%7Ealbourae/>, 2006.