Towards Release Planning Generic Model: *Market-driven software development perspective*

Bassey Isong
Computer Science and Information Systems, University of Venda,
Private Bag X5050, Thohoyandou 0950, South Africa

Abstract

Several techniques to achieve software release planning (RP) in market-driven software development (MDSD) exist. One of such approaches is prioritization due to large volume of requirements that often can't be implemented at once. However, the task of selecting optimal sets of features for a particular release is challenging due to complex and fuzzy dependencies that often impact RP decisions negatively. In addition, existing RP models are not generic and known to only address limited requirements selection factors, making it impossible for engineers to choose a model that suits a particular application. Therefore, a generic RP model that supports all selection factors and allows users to define their needed factors is indispensable. To this end, the objective of this paper is to bring into light the challenges of RP, and proposed an approach for representing dependencies among requirements. In addition, the study proposes a generic framework for RP based on the EVOLVE Model. The proposed model will also allows for re-planning, assist MDSD organization to improve the quality of the selection and deliver quality products with attractive sets of features to have a competitive edge.*

IJERT

1. Introduction

In today's e-society, software development has become extremely a complicated and a critical activity due to extensive advanced technology usage, the growing awareness for software products usage and market demand. However, such remarkable progresses have not yet been balanced by software engineering practices. Requirements engineering (RE) is one of the key activities that deals with the discovery, documentation, communication and implementation of software requirements [1]. Unfortunately, RE processes are not sufficiently understood and poses huge challenges to organizations. The situation is exacerbated in today's market-driven software development (MDSD) environment where the requirements are characterized by large volumes and continuous changes throughout the course of a project [2],[3]. This stems from the fact that MDSD have no specific identifiable customers and the requirements are often invented [4].

MDSD objective is to attain a competitive advantage by taking a reasonable market share, attract wide range of customers and amassing profits [2]. This is usually achieved by a good software release planning (RP) [5],[6],[7]. It constitutes a determinant factor of the success or failure of a company's product in the market. However, achieving such objectives is challenging and though is critical to software product development [5]. RP is one of the most recognized activities that challenged several organizations developing for the mass market [2],[8],[9]. The problems that stems from RP has been described as "wicked" [10], involving a complex decision-making activity. In particular, such decisions have become even more complex with large number of stakeholders since it often yields more requirements that cannot be implemented at once [8]. This requires that requirements should be prioritized in order to that the most important ones are met by the earliest product releases.

Prioritizing requirements has been recognized as a sub-problem of RP, a crucial activity involving the selection of requirements based on a set of criteria such as scheduling, stakeholders, resource planning and interdependencies [11]. Regrettably, selecting requirements based on priorities has been known to be difficult, if not impossible since most requirements cannot be treated independently due to complex dependences [3],[5]. Consequently, decisions made to one or many requirements may impact others in ways not expected during development [12]. Hence, dependencies among requirements have to be taken seriously in order to enhance the process of prioritization which in turn facilitates quality product release plan in MDSD. Planning a product release in MDSD inevitably involves dealing with all categories of dependencies. With the complexity of RP, requirements dependencies pose an important research area since little attention has so far been gained in existing literature. Research in this area has been focused mostly on specific problem or a development activity which does not specifically address RP problems [3].

In addition, several software RP models have been proposed and some are validated, while some are not yet validated [13]. These models have been designed to support RP either on strategic or operational levels in MDSD and as well, improve the quality of feature selection [14]. Unfortunately, there are not generic and not applicable in all situations. Each model has been known to addressing only little selection factors: soft and hard constraints. In addition, re-planning is not addressed by most of the models. As a consequence, it becomes difficult to choose a model that is suitable for a particular application. We therefore consider it important to have a generic RP model that combines all these factors and allow engineers to make their choice of factors in order to generate a good release plan for a product. Thus, the objective of this paper is to discuss the nature of prioritization and dependencies challenges during software RP, and propose an approach to identify dependencies among requirements. In addition, we propose generic framework for RP model based on EVOLVE* model with the goal of enhancing the RP complexity, taking into account all requirements selection factors as well as re-planning of release. With this proposed approach, software engineers can be assisted in achieving a fast and quality product plan that adds values to customer needs.

The rest of the paper is organized as follows: RP is discussed in Section 2, prioritization and dependences challenges in Section 3 and 4 respectively. In Section 5 we discuss dependences effects on priority, while in Section 6 presents existing RP solution models. Section 7 and 8 presents the proposed dependences identification approach and a framework for RP model respectively. Finally, Section 9 is the paper conclusion.

.

2. Release Planning

Software RP is an important and a continuous activity in MDSD that has attracted significant attention in recent years. It is concern with the process of deciding on what subsets of requirements to implement during the course of product development and in which releases of the product. In other words, RP is aimed at selecting subset of requirements that can yield optimal realization of products in a certain releases [8],[15]. However, in MDSD organizations RP activity poses a huge challenge since there are several factors (i.e. technical, resource, risk and budget constraints) that influence the decision of selecting requirements - hard constraints and soft factors [13]. For instance, during the course of a project, many different decisions regarding product release plan has to be made like feature value and urgency, available resources, milestones, stakeholder concerns, available market opportunity, risks, product strategies, features interdependencies, cost and so on [5],[7]. All these factors are critical to the success of the products in the market, albeit is a challenging task for release planners.

.In the same vein, the value and cost of individual features are affected by the existence of relationship of diverse aspects which are roughly tied to fixed delivery date and set of available resources, etc. The complexity of RP is dependent on the accurate estimation of the cost and value [8],[9]. The implication is that, if cost is underestimated, deadlines may be missed while over-estimating cost may lead to the exclusion of valuable requirements. Accordingly, if value is over or under estimated, the result is products that does not bring into line, the actual market needs and consequently, a failure of the product. Consequently, the absence of a good RP will affect customer's satisfactions, project time, budget, and perhaps, a market share loss [15]. Thus, good RP approach should be in place in order to enhance the quality and speed of product release plan. Approaches such as prioritization, resource demand estimation, and requirements selection can be used to achieve RP in MDSD.

3. Requirements Prioritization

In MDSD today, due to the existence of large potential customers and stakeholders, large volume of requirements are produced that often cannot be implemented at the same time. This requires that the correct subsets of features which can add value to customers and be implemented within budget for an organizational success in the market for next product releases have to be selected. Prioritization activity is an essential step leading to good decisions taking involving product planning for multiple releases. It is the activity during which the most important software requirements, their implementation and testing order throughout the development lifecycle are established for a system [1]. It allows software engineers to focus on a subset of all the requirements, and implement them in the earliest product releases [8]. One greatest

IJERT

challenge is how to prioritize large number of requirements.

This however affects RP negatively, and prioritization has been recognized as a very challenging requirements activity with no effective and systematic methods to perform it in most software organizations [16]. Accordingly, Karlsson et al [17] stated that to prioritize requirements, domain knowledge and skills of estimation is critical for success. But in practice, priorities determination is difficult and is not simple to identify the aspects on the basis of which its decisions should be based [11]. Priority itself is a complex mixture of different aspects ranging from importance, cost, time, stakeholders, penalty to risk and where each is an extremely multifaceted concept [18]. For example, requirements importance could be a composite of implementation urgency, product architecture, strategic importance of the organization, etc. [11],[18]. Moreover, decision makers ought to take into account these various aspects before deciding the implementation scheduling of the requirements [18]. Requirements themselves do not exist in isolation, and priorities are always complex in relationship. Consequently, their importance varies from release/customers to another.

Several approaches to perform prioritization of requirements exist such as the analytic hierarchy process (AHP), greedy-type algorithms, cumulative voting, the 100-dollar Test, numerical assignment (grouping), requirements triage, Wiegers' Method, top-Ten requirements, planning games, etc [8],[19]. These approaches are categorized as either methods based on giving values or negotiation approaches [17]. Their drawbacks largely depends on their level of scalability, none consideration of different stakeholder views, RP effort constraints, etc [19]. In addition, they have fixed model and do not allow requirements changes or full priorities and in some cases, priorities are always influenced by the people involved. Other approaches are based on decision support tools with respect to RP such as the EVOLVE family, etc[13],[15].

4. Requirements Dependences

Software requirements are not isolated entities rather they are related to and impact one another in ways that is considered complex emanating from cost/value bonding. Consequently, most developed individual requirements cannot be treated separately during software development [9]. The implication is that several other development activities like RP are affected in a way not expected. For instance, one or more requirements may affect others by either constraining their implementation order, cost, or value to the customer. In practice, not all requirements are related or affect each other. Their levels of relationship are categorized. For example, a study by [3] shows that only about 20% of the requirements accounts for about 75% of the dependencies, in a software system and MDSD is known to have more value-related dependencies that bespoke.

Though, less work has been done in the area of requirements interdependencies, few strategies for identifying and managing interdependencies exist. The study by [9] identified three types of requirements dependencies: structural, constraints, and cost-value interdependencies. Another study, [3] proposed a classification method for interdependencies such as functional related (AND, REQUIRES) and value related (ICOST, CVALUE) for bespoke and MDSD respectively. Others are OR and TEMPORAL dependencies. In addition, visualization method was applied for the ease of interdependencies identification to facilitate RP. Johan et al [20], also work on automated similarity analysis which uses language tools to analyze sets of requirements based on [3]. Results obtained shown that the technique only identified similarities between requirements with a correct classification of up to 16% of the actual dependencies.

IJERT

5. Dependences Impacts on Priority

Requirements dependencies itself is not considered problematic, but the manner they affects a number of other development activities and decisions, makes them problematic and complex [3],[9]. In MDSD, interdependencies among requirements are value-related which tends to impacts requirements priority negatively with respect to RP in an unanticipated way. For instance, the selection process during RP is not always easy as thought because the relationship among requirements is complex. Consequently, the choice of one requirement may warrant the selection of one or several other requirements as well. For example, selecting a highly prioritized requirement **Req1** may trigger the selection of a costly but lowly prioritized requirement **Req2**. This implies that **Req1** will not be implemented without firstly implementing **Req2.**

All this contributed to several companies having a serious challenge leading to bundling related requirements without considering the cost-value complexity relationships among them which in turn give rise to poor customer satisfaction, product failure in the market, etc. In the perspective of RP, requirements dependencies is crucial but hardly ever identified clearly and have been deemed complex and fuzzy in nature [3]. Therefore, the understanding and identification of these relationships is indispensable in order to avoid costly mistakes.

6. Release Planning Models

Several models for strategic RP exist in literature, with most of them providing decision support for RP while others deals with the issues of prioritization and selection techniques of requirements [9],[19],[21]. The models are used facilitate RP decisions taking in to account several selection factors that were not addressed by prioritization techniques. In a systematic literature review carried out by [13] on strategic RP models, analysis revealed that the existence of twenty four (24) models out of which 23 (96%) models are validated in both industry and academia while one (4%) not yet validated. List of all the models studied can ie found at [13]. In addition, about 83% of the models are common for bespoke and MDSD while 17% are suitable for MDSD only. Further analysis by [13] shows that the dominant models are the one belonging to the EVOLVE-family and ReleasePlanner tool [13]. For instance, the EVOLVE-family constitutes the largest group of strategic RP models, which is about 16 models. In this group, EVOLVE+ and EVOLVE* are the direct derivative of EVOLVE. Others extensions are EVOLVE^{ext}, Evolutionary EVOLVE+, S-EVOLVE and F-EVOLVE*. EVOLVE* is one of the best models in this group which is based on genetic algorithm, taking account the all the technical constraints during potential release plans [10],[14],[15]. Its architecture is made up of the Modeling, Exploration and Consolidation phases [15].

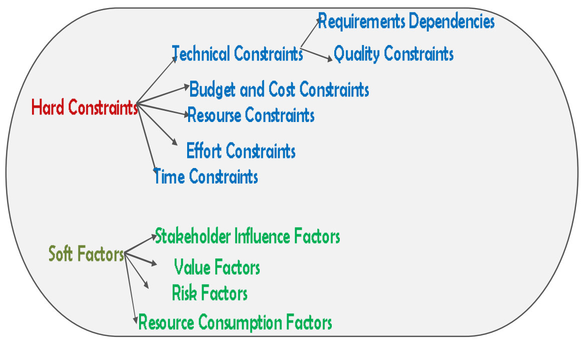

Fig. 1: Requirements selection factors [13]

However, one of the greatest issues with the existing RP models is the fact that most of the approaches focus on a limited set of requirements selection factors: soft factors and hard constraints (see Fig. 1). Furthermore, majority of models placed emphasis on hard constraints while about 58% include soft factors [13]. These requirements selection are critical to the quality of a given product release plan. Consequently, the models are not generic and cannot be applied to all situations or take re-planning into account. The bottom line is that, it is difficult to make a choice existing model which is suitable for a particular product. We therefore deem it imperative to have a RP model that supports all requirements selection factors and allow users to define which requirements selection factors are needed in order to generate a fast and quality release plan for a product.

7. Approach for Dependences Identification

The priority of requirements is critical to the success of RP, but is often crippled by requirements interdependencies. This stems from lack of explicitness which makes them complex to identify and managed coupled with their fuzziness nature [3]. Thus, to identify explicitly the nature of dependencies and support human decisions during the course of RP, an intermediate representation is indispensable. In this section, we propose an approach based on [3], utilizing dependency graph theory which we called requirements dependencies graph (RDG). The representation is simplified by the computation of both in-degrees and out-degrees for each requirement, ***R***. The goal of the representation is to support or facilitate possible and good way of scheduling requirements set in a release plan. Our intuition is that, the representation can go a long way to offer clear and fast identification of all forms of dependences (such as singular, clusters or highly dependent requirements [3]) at a quick glance. These are discussed as follows:

Definition 1: [**Dependency Type (DT)**] Based on [3], we first of all classify these dependences into six types: AND, REQUIRES, TEMPORAL, IVALUE, ICOST and OR. And ***R*** is defined in the following way:

Definition 2: **[RDG]** Given set of requirements to be selected for next product release, ***R*** and let **G < V, D, DT** > represent the RDG, where V is a finite set of nodes representing the requirements ***R*** and D = V × V × DT represents the set of various edges with dependency types: DT ={AND, REQUIRES, TEMPORAL, IVALUE, ICOST, OR}. The computation for the values of DT is described as follows:

IJERT

Definition 3: **[Out-degree]** The out-degree of ***R*** *ϵ v* is the number of DT emanating from that node. The out-degree of v is computed by |***A(v)***|.

Definition 4: **[In-degree]**The in-degree of ***R*** *ϵ v* is the number of DT incident on that node. The in-degree of v is computed by |***I(v)***|.

A typical example is illustrated in Fig. 2 and the corresponding in-degrees and out-degrees for each ***R*** are presented in Table 1. The representation is simple and easy to comprehend how requirements relate with one another. The nodes are the requirements while the edges are the dependencies types. Important conclusions about require drawn by merely looking at the graph. With Table 1, singular requirements (e.g. R8), clustered (e.g. R6) and heavily depended requirements (e.g. R9) can be easily identified.

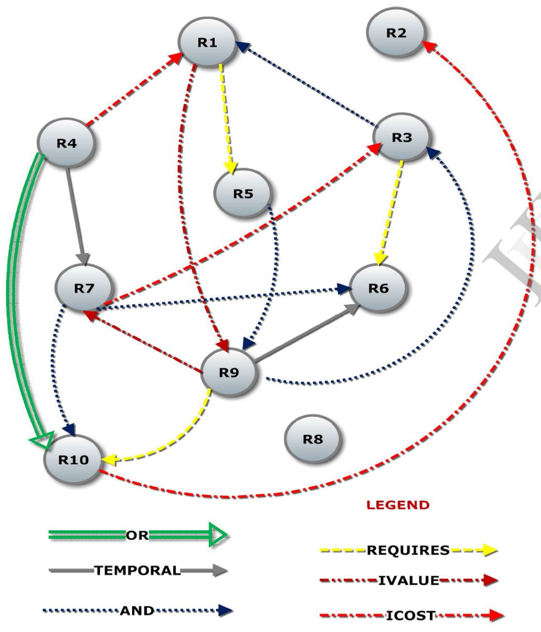

Fig. 2: Requirements dependencies graph

By following the recommendations in [9], with this representation we can see easily requirements with no relationship (i.e. singular requirements) and can be scheduled for any release as "top-off" depending on the amount of available development resources from an interdependencies perspective. Accordingly, risk can be minimized by scheduling requirements that are highly couple to several others for early release, while the clustered ones can be planned for any release as long as all involved requirements are planned for the same release.

Table 1: RDG In-degrees and Out-degrees

R ε v	*A(v)*	*I(v)*
R1	2	2
R2	-	1
R3	2	2
R4	3	-
R5	1	1
R6	-	3
R7	3	2
R8	-	-
R9	4	2
R10	1	3

IJERT

One important limitation of this proposed approach is when it comes to representing large volume of requirements. MDSD often have large requirements and it will be challenging representing them this way, except by automation of the dependencies. We however recommend more research in this area in order to explore more possibilities of identifying dependencies in requirements. We consider it important because knowing how requirements relates with one another will significantly help in speeding up more accurate cost and schedule analysis during product RP.

8. A Framework for RP Model

In this section, we describe the structure of our proposed model which is based on the EVOLVE* architecture: modelling phase, exploration phase and the consolidation phase (see Fig. 3). We consider the framework generic because it is designed to take into account all the hard constraints and soft factors, giving users the opportunity to define which selection factor is required during the course of RP. It serve as a guideline for developers of RP models to follow in order to develop models that can provide solutions for all situations (i.e. planning and re-planning) of software release. It is iterative is iterative in nature and we present here a high level description of the framework.

8.1. Modelling Phase

At this phase depending on the goal (i.e. planning or re-planning), the repository can be accessed for necessary requirements selection candidates, etc. Based on the design of the model in use, the problem is then formulated considering the soft factors and hard constraints. Though not all selection factors are useful in all situations, the basic factors that can influence the decisions on requirements selection for their particular case should be chosen. For re-planning problem, different types of information such as recent plan, change requests and other release information such as time, etc are important.

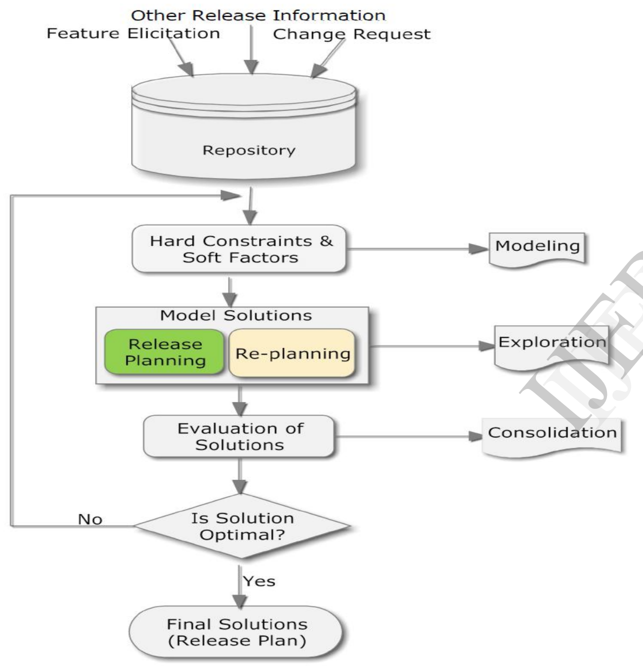

Fig. 3: RP generic framework

8.2. Exploration phase

With the formulated problems and based on the goal, the actual solution plan generation is done at this phase. Here model designer can use any algorithm of interest to achieve the desire objective of a quality and a fast plan. With the techniques in use, diverse solutions can be achieved such as the generated plan, re-planning information like when to re-plan, how to re-plan and what to re-plan. Consequently, information obtained will assist in generating attractive sets of features that is capable of entering the market at the right moment, and that can generate sufficient revenues to cover development costs and bring profits to the organization.

8.3. Consolidation phase

This is the last phase which is the decision making phase. Here, the solution or simply the plans generated at the exploration phase are presented to the decision maker for onward evaluation. The decision maker should study the solutions at hand and analyze them based on past experience and the context surrounding the problem. The analysis can contributes to the problem comprehension, modification of the parts of the underlying model if need arises and reduce the size and complexity of next iteration [8]. Several iterations are necessary until the desire solutions are achieved which can be used for either planning or re-planning.

9. Conclusion

Release planning is one of the most serious challenges that confront several software development organizations that produce software for mass markets. It has been known as a complex activity and constitutes a determinant factor of the success or failure of a company’s product in the market. In this paper, we have explored some of the challenges faced by RP, taking prioritization and dependencies into account as well as strategic RP models. These two activities play key roles in RP, but the relationship among requirements makes them challenging activities during the course of selecting optimal set of requirements for a particular product release. To this effect, we have proposed an intermediate representation of requirements using a directed graph to assist engineers to quickly and easily identify how requirements are related and also decide on which requirements to be scheduled in the next release that is capable of achieving higher business value. In addition, existing software RP models do not sufficiently addressed all the selection factors and do not consider re-planning, making it cumbersome to choose a good model that is suitable for a particular situation. To this end, we have also proposed a generic framework for release planning solution model based on EVOLVE* architecture to enhance the quality of software RP. We therefore conclude that RP model that supports all the hard constraints and soft factors and which allows for re-planning will go along way reducing the problems with RP to the lowest level and improve the quality of the selection. With good features selection in a release, MDSD organizations can deliver quality products to stay ahead of competition in the market.

Our future work will be based on implementing the approaches discussed in this paper on a real world system and evaluate their effectiveness.

IJERT

10. References

[1] Nuseibeh, B., Easterbrook, S.: "Requirements Engineering: a Roadmap". Proceedings of "The Future of Software Engineering", pp 35-46 May 2000.

[2] Karlsson, L., Dahlstedt, A.G., Natt, J., Regnell, B. and Persson, A.: Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study, Proceedings of Eighth International Workshop on Requirements Engineering: Foundation for Software Quality, 2003, pp. 101-112.

[3] Carlshamre, P. et al.: An industrial survey of requirements interdependencies in software product release planning. In: Proceedings of the 5th International Symposium on Requirements Engineering, Toronto, Canada, 2001, pp. 84-91

[4] Sawyer, P.: Packaged software: Challenges for RE. In Proceedings of the Fifth International Workshop on Requirements Engineering: Foundations for Software Quality(REFSQ 2000), Stockholm, Sweden, 137–142, 2000

[5] Carlshamre, P. "Release Planning in Market- Driven Software Product Development: Provoking an Understanding", Springer, pp. 139-151, 2002

[6] Ruhe, G. "Software Release Planning" University of Calgary 2500 University Drive NW Calgary, AB T2N 1N4, Canada, 2004.

[7]G. Ruhe, and D.Greer, "Quantitative studies in software release planning under risk and resource constraints", Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE 2003), IEEE Computing Society, Los Alamitos, CA, 2003, pp. 262–270.

[8] Pa¨r Carlshamre "Release Planning in Market- Driven Software Product Development: Provoking an Understanding", Springer, pp. 139-151, 2002

[9] Dahlstedt, Å.G., Persson, A.: Requirements Interdependencies: State of the Art and Future Challenges, Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality, 2003, pp. 71-80

[10] A. Ngo-The and G. Ruhe, "A systematic approach for solving the wicked problem of software release planning," Soft Computing - A Fusion of Foundations, Methodologies and Applications, vol. 12, pp. 95-108, 2008.

[11] Lehtola, L. Kauppinen, M. and Kujala, S. Requirements Prioritization Challenges in Practice, Springer, 2004, pp. 497–508

[12] Regnell, B. and Brinkkemper, S. "Market-Driven Requirements Engineering for Software Products," in Engineering and Managing Software Requirements, Berlin Heidelberg: Springer, 2005, pp. 287-308.

[13] Svahnberg, M. et al. "A systematic review on strategic release planning models". Journal of Information and Software Technology 52, 2010, pp. 237–248

[14] Du, G., McElroy, J., Ruhe, G.: Ad-hoc versus systematic planning of software releases: a three-staged experiment, in: Proceedings of the 7th International Conference on Product-Focused Software Process Improvement (PROFES2006), Lecture Notes in Computer Science LNCS 4034, Springer Verlag, Berlin, Germany, 2006, pp. 335–340.

[15] Greer, D. and Ruhe, G. Software release planning: an evolutionary and iterative approach, Information and Software Technology 46 (2004) 243–253

[16] Lehtola, L. and Kauppinen, M. Suitability of Requirements PrioritizationMethods for Market-driven Software Product Development, Softw. Process Improvement Practices, 2006

[17] Karlsson L, Berander P, Regnell B, Wohlin C. Requirements prioritisation: An experiment on exhaustive pair-wise comparisons versus planning game partitioning. In Proceedings of Empirical Assessment in Software Engineering (EASE2004), Edinburgh, Scotland, 2004

[18] Berander, P. and Andrews, A.: "Requirements Prioritization," in Engineering and Managing Software Requirements, A. Aurumand C. Wohlin, Eds. Berlin: Springer, 2005, pp. 69-94..

[19] Karlsson, J., Wohlin, C and Regnell, B.,"An Evaluation of Methods for Prioritising Software Requirements", Information and Software Technology 39 (1998), pp. 939-947

[20] Natt och Dag, J. Regnell, B., Carlshamre, P.; Andersson, M. and Karlsson, J., A feasibility study of Automated Natural Language Requirements Analysis in Market-driven Development, Requirements Engineering, 2002, p 20-33

[21] Lehtola, L. and Kauppinen, M."Suitability of Requirements Prioritization Methods for Market-driven Software Product Development," Software Process Improvement and Practice, vol. 11, pp. 7-19, 2006

IJERT