Criteria-Based Framework for Software Product management

Samer I. Mohamed [(1),] Islam A. M. ElMaddah [(2),], and Ayman M. Wahba[(3)]

(1) Department of Computer and Systems.Ain Shams University (Egypt)
E-mail: samer.mohamed@eds.Com

(2) Department of Computer and Systems.Ain Shams University (Egypt)
E-mail: islam_elmaddah@yahoo.co.uk

(3) Department of Computer and Systems.Ain Shams University (Egypt)
E-mail: ayman.wahba@gmail.Com

ABSTRACT

Value-Based Software Engineering (VBSE) becomes one of the most promising approaches for software product management [10]. It focuses on the critical role by which stakeholders and business core values affect decision making which in turn influence the product success. This paper illustrates the Criteria-Based approach for software product management through a computer based software framework. The framework can select the best candidate requirements for each release based on the stakeholders' input values for multiple criteria associated with each requirement. These criteria reflect the priority of each requirement not only in terms of perceived importance to the stakeholder and anticipated implementation cost criteria [18] but also through technical risk, relative impact and market-related aspects criteria. The framework has the capability to balance between the different stakeholders' preferences based on the stakeholder's weight provided by the product manager [35]. By this means the introduced framework will enable the product manager to overcome many challenges throughout the product life cycle by providing him with the different features that make the decision making process much easier and finally yields to product success [16].

Keywords: Release planning, Value-Based software engineering, Value-Oriented prioritization, Hierarchical Cumulative Voting prioritization, Requirements management tool.

1- INTRODUCTION

A value-based approach for software product management becomes one of the major aspects that affect the software industry in the current market-driven environment. This happens because satisfying the customers' needs and expectations becomes one of the most important factors for software product success. Today's market-driven environment with rapidly changing customer requirements, needs faster adaptability by the product team to respond to these changes while maintaining a focus on the value gained from these changes.

This approach is more appropriate for today's challenging environment because it sets not only goals for improving both productivity and product quality based

on the stakeholders' considerations but it also keeps tracking for the core business values over the product life cycle. Adopting this approach will help avoid eventual shortfalls from the lack of user inputs, unrealistic expectations, unclear objectives and incomplete requirements.

The product quality in software organizations is shaped by the ability to satisfy their stakeholders' requirements and expectations. Therefore, one of the major aspects on which software organizations focus is how to maximize the satisfaction of their stakeholders [15]. This requires a mechanism to support the decisions regarding to the requirements contents of the different product releases as well as maintaining the strategic organizational goals.

The product manager has a crucial task throughout the product life cycle due to the ongoing challenges in the market-driven environment. Requirements prioritization is considered to be one of these challenges. First, because the term priority has different meanings and varies between different stakeholders. Second, due to involving different stakeholders and combining their opinions and preferences that may contradict with one another [4]. Thus, the decision making process about which requirements to include within each release, becomes a critical task for the product manager and makes an automated-support mechanism critical for the product success [34].

Our main objective from this paper is the design of a new prioritization algorithm that gets benefit from both Hierarchical Cumulative Voting (HCV) and Value-Oriented prioritization techniques to come with a more powerful technique used as the core engine for our product management framework. The proposed framework incorporate and integrate other features like effort estimation and risk assessment features that can be used to facilitate the product management over the product life cycle.

The organization of this paper is as follows. In the next section, we will refer to the related work for our research. The first part of this section will describe the requirements prioritization which is the main core engine of our framework. The second part will describe two of similar product management frameworks from the market. In section three, we will elaborate on the rationale for the criteria-based framework, and the research methodology we have applied to develop it. In section four, we will discuss the basic architecture of the criteria-based framework. In section five, the basic framework features will be illustrated. In section six, an industrial case study for a new software product will be represented to show how the framework works under different conditions. The final section will summarize our conclusions and suggest future directions for further work.

2- RELATED WOR

2-1 REQUIREMENTS PRIORITIZATION

Requirements prioritization becomes one of the critical parts of the product release

planning nowadays because it becomes necessary to distinguish the vital requirements from the less important ones in order to maximize the overall business value by satisfying different key interests, technical constraints, and preferences of critical stakeholders [4]. By identifying the requirements that are most important, least costly, least risky, it is possible to find a favorable mix of requirements that can be used to produce a system that implements only a subset of all requirements, while still satisfying customers. To find the requirements that add most value to business, it is possible to utilize some of the available prioritization techniques. Two of the prioritization techniques that focus on maximizing the business value are the Hierarchical Cumulative Voting (HCV) prioritization and Value-Oriented Prioritization (VOP). Our proposed prioritization technique Value-Oriented Hierarchical Cumulative Voting (VOHCV) integrates the strengths of both prioritization algorithms. VOHCV depends not only on requirement value/importance to the stakeholder as HCV but it takes different other aspects into account while prioritizing the requirements. Examples of these aspects are associated implementation cost, requirement technical risk, relative impact and market-related aspects. VOHCV also supports hierarchical requirements structure which differs from VOP that supports only flat structure. This enables the proposed technique to prioritize requirements on different abstraction levels. Supporting requirements prioritization on different levels of abstraction is vital because requirements exist naturally on different levels of abstraction and can form hierarchical structures. In large-scale software development in general, and in market-driven software development in particular, requirements commonly arrive in different shapes and form, at multiple levels of abstraction, and are described at various level of refinement. Lehtola and Kauppinen concluded that requirements on different abstraction levels caused problems when performing prioritizations since lower level requirements were considered as less important than higher level requirements [20]. Thus, it is important to be aware of different abstraction levels when prioritizing requirements, and requirements should only be compared with requirements on the same abstraction level [31].

2-1-1 Hierarchical Cumulative Voting prioritization

HCV technique is one of the prioritization techniques that addresses the weakness of the Analytical Hierarchy Process (AHP) technique and gets benefit from the advantages of the Cumulative Voting (CV) technique. The main idea behind the HCV technique is to quantify the importance of the requirements and prioritize the requirements based on the weight given to each requirement. The main difference between CV and HCV is that not all requirements are prioritized at the same time in the second technique, but prioritization is performed at different levels of the hierarchy. The importance of this technique increases as the number of requirements grows, which makes the need for a structural approach to prioritize the requirements vital. HCV supports that structural approach by using the relationships between the requirements while performing the prioritization. The prioritization process is done in a series of steps to minimize the number of requirements prioritized at the same time [3].

2-1-1 Value-oriented prioritization

VOP is one of the prioritization techniques that are proved to align product demands with company goals and stakeholders expectations. This is done through providing a visible and defined process for prioritizing and managing requirements over the product life cycle [30]. It helps the stakeholder view the whole picture for the sake of the organization targets and vision, rather than arguing over which product requirements to implement. The main idea behind VOP is to focus on the core business values that lead to stakeholders' satisfaction while prioritizing the product requirements; as indicated by Karl Wiegers [31]. Each business value is given a weight based on the organization's objectives. Each stakeholder puts his estimate against each business value for each requirement. All these input values are then consolidated together to produce the requirements' ranks.

2-2 REQUIREMENTS MANAGEMENT TOOLS

Requirements management tools are designed to help improve the software development process through providing many features to ensure higher product quality. There are a number of commercial Requirements Management (RM) tools currently existing in the market [12, 14] and others identified in the research literature [21]. We will refer to Borland CaliberRM and ReleasePlanner as two examples of these tools that have been used in our research. Our proposed criteria-based framework manages not only requirements over the product life cycle as CaliberRM but it also provides the product manager with a complete release plan. The generated release plan identify the contents for each release based on the stakeholders viewpoints for the different aspects affect each requirement like importance, implementation cost, requirement technical risk, relative impact and market-related aspects. The criteria-based framework handling different requirements structures like flat structure which is only supported by ReleasePlanner and also hierarchical structure. This feature makes our proposed framework capable of dealing with requirements in different abstraction levels as described in the previous sections.

2-2-1 CaliberRM

CaliberRM provides a central and secure repository for all the product requirements. One of the major features of CaliberRM is that it has an open architecture that permits requirements to be linked and traced across the product lifecycle. It also supports requirements prioritization based on value/cost criteria for each requirement [6]. It permits web-based requirements management and provides means for communication among the project team.

2-2-2 ReleasePlanner

ReleasePlanner facilitates planning and prioritization of projects taken into consideration the resource capacities in addition to the stakeholders opinions. This will help decision makers to optimize their release planning through resource availability and stakeholders' satisfaction. It also enables the decision makers to

develop their own scenarios and perform what-if analysis as means of developing strategies to overcome organizational limitations [28].

3- RATIONALE AND RESEARCH TECHNIQUES

3-1 RATIONALE

The rationale behind developing a new software product management framework arises from the need for having an automated mechanism that enables a product manager to handle the different challenges he faces throughout the product life cycle [22]. These challenges can affect the product success [16], as indicated by the Standish Group Chaos Report, which is an annual survey of the successes and failures on IT projects [30]. Examples of these challenges are:

- How to identify the criteria on which requirements are prioritized within the product releases.
- How to best allocate the resources over the product release to be able to satisfy the stakeholders requirements.
- How to balance between the different stakeholders' opinions throughout the prioritization process.
- How to handle requirements volatility and the need for re-planning to satisfy the stakeholders.
- How to trace the requirements specifications through the different release phases.
- How to identify the requirements interdependency and inter-relationships.
- How to provide accurate estimates throughout the product life-cycle.

These challenges provide a starting point for introducing a group of new innovative ideas for tackling them throughout the product life cycle and making the decision making process more easier. Examples of these innovative ideas are:

- Use not only the perceived importance to the stakeholder and anticipated implementation cost criteria in identifying the priority of the requirement, but also through using technical risk, relative impact and market-related aspects criteria. This mechanism makes this framework differ from other corresponding frameworks and requirements management tools that depend on using only value/cost criteria in the process of requirements prioritization. This way our framework gets the effect of the different factors and criteria affect requirements while identifying the best release requirements.

- Support the PM with different resource allocation models to identify the best candidate based on the release case will enable him to best allocate the resources over the product releases.
- Provide a mechanism for the different stakeholders to be able to enter their viewpoints for each requirement without any conflicts. The mechanism will also balance between the different stakeholders' viewpoints based on the stakeholder weight.
- Provide the PM and stakeholders with an easy way to enter all of the necessary and required information for each requirement; making the requirement details clear and narrow down the volatility risks.
- Use a traceability matrix to link between the requirements specifications and source code; facilitating detection of any faults as early as possible in the release life cycle.
- Have a new mechanism to detect requirements' interdependency and relationships among them through post and pre condition features; this helps much in distributing the requirements over the product releases in a way that minimize the rework cost and enhance the product throughput.
- Provide a new technique to estimate the required development effort for each requirement through using a development template and a risk assessment. This enables the PM to accurately estimate the required effort and efficiently manage the product resources.

Our main contribution in this paper is the design of a new prioritization algorithm that gets benefit from both Hierarchical Cumulative Voting (HCV) and Value-Oriented prioritization techniques. Integrating the strengths of both prioritization algorithms makes the new proposed algorithm more powerful in handling the prioritization process efficiently because it incorporates multiple dimensions while selecting the best candidate requirements. This new algorithm acts as the main core for the Criteria-Based framework. This framework integrates different other release planning features and methodologies that handle the different challenges faced by the product manager throughout the product life cycle.

3-2 RESEARCH TECHNIQUES

The research methodology we followed for the conception of this framework is described as follows:

- Literature review for the current and practical challenges for the software product management from both business and strategic perspectives [4, 8, 11, 17, 25, and 26].
- Field interviews and brainstorming sessions with highly qualified product managers for the product management point of view, sales representa-

tives for the vendor point of view and general managers for the strategic view.

- A prototype implementation for the framework based on the knowledge gained from the previous points. The prototype features are mainly for tackling different challenges faced by the product manager [2, 24, 33].
- Validation session with a group of the product managers using an existing product case study to identify the different points of strength and weakness on the implemented framework.

Gathering feedbacks from the different parties based on the prototype practice to be the initiative for the future work.

4- FRAMEWORK ARCHITECTURE

4-1 PRODUCT MANAGEMENT HIERARCHY

The basic architecture for the proposed framework is based on the artifacts of the product management hierarchy: each product has a release cycle that consists of past, current and future releases. Each release consists of a group of selected requirements identified based on the stakeholders' input values for each requirement (these values reflect the importance of each requirement in terms of associated anticipated cost, market-related aspects, technical risk, relative impact and perceived importance to the stakeholder). Finally the basic building block for each release is the requirement which adds a technical or functional value to the whole product [1].

4-2 ARCHITECTURE DETAILS

As indicated from the previous sub-section, there are three main layers for the product management hierarchy. We built our framework based on this artifact as (shown in figure 1). The details for each layer will be described as follows.

4-2-1 Product road mapping layer

This layer focuses on the product high level processes. The product manager initially edits the stakeholders profiles and assigns a weight for each one on scale from 1 to 9 based on the importance of this stakeholder to the project (stakeholders who have a critical impact on the product success will be given 9 while those who have lowest impact will be given 1). Once each stakeholder has his profile enabled by the product manager, he can start entering his view point for each product requirement like ReleasePlanner tool. These viewpoints will be taken into account during "Requirements identification" and "Requirements prioritization" processes. The product manager will use the stakeholder requirements as an input while identifying the product scope. Based on the product scope, the product manager will be able to identify the product resources required to implement the product. Having the product scope and resources defined, the product manager will be able to identify the different product releases required for product completion [5].

4-2-2 Product road mapping layer

This layer focuses on those processes that manipulate the requirements over the lifecycle. The product manager will be the source for gathering the input product requirements from different stakeholders (to reflect client objectives), and Research and Development (R&D) (to reflect technology-driven objectives). Once the product manager consolidates the product requirements from different parties, he and other stakeholders will be able to enter their input values for each requirement. These values will reflect the stakeholder viewpoint in terms of the different aspects affect each requirement. The stakeholder will enter his opinion/viewpoint for each aspect (Cost, Value, Risk, Impact and market-related aspects) through assigned weight from 1 to 9 such as 9 indicates highest importance and 1 indicates lowest importance. This will be done using the different requirement attributes through the "Requirements definition" process. The consistency of the defined requirements then will be examined through the "Requirement consistency check" process. Any consistency violations (e.g., start and end implementation dates check, unique identifier, allocated resources) will be prompt to the user to fix. A valid list of requirements taken from the consistency gateway will be examined for any correlation between the different requirements through the "Requirements dependency check" process. The correlated list of requirements will be taken as an input for the "Build tractability matrix" process. This process will be responsible to build the different links between the requirement and implementation modules/components. The requirements list at this phase will reflect the actual product objectives. Thus the product manager will be able to estimate the effort required for each requirement through the "Requirements effort estimation" process. This will help the product manager to efficiently allocate the product resources through the release planning.

4-2-3 Release planning layer

This layer focuses on the processes that manage product releases. The product manager will identify the release scope based on the final list of requirements taken from the "Requirement management" layer and release list taken from the "Product road mapping" layer. From the release scope which indicates the objectives from each release, the product manager will then edit the release profile by adding more details like; release planned total cost, number of resources, planned start and end dates. These planned values will be monitored and tracked against the actual values through the product execution phase to detect any overrun in both time and cost as early as possible. Also both actual number of resources and allocation percentage per resource will be monitored and tracked through the product execution phase to give the product manager the ability to utilize best the release resources and provide the highest throughput [35]. According to the product releases definition and requirements definition, the prioritization process will control how the requirements will be allocated over the different product releases. Therefore the product manager will initially setup the prioritization settings that control the prioritization process outcomes through the "Calibrate prioritization" process. The ranks produced by the "Prioritization" process will be used to identify the requirements contents for each product release. The

process of release planning is iterative [9]. Thus for any mismatch between the output releases contents and stakeholder expectations, the prioritization parameters are recalibrated and the process reiterated till having a match between the results and expectations [7].

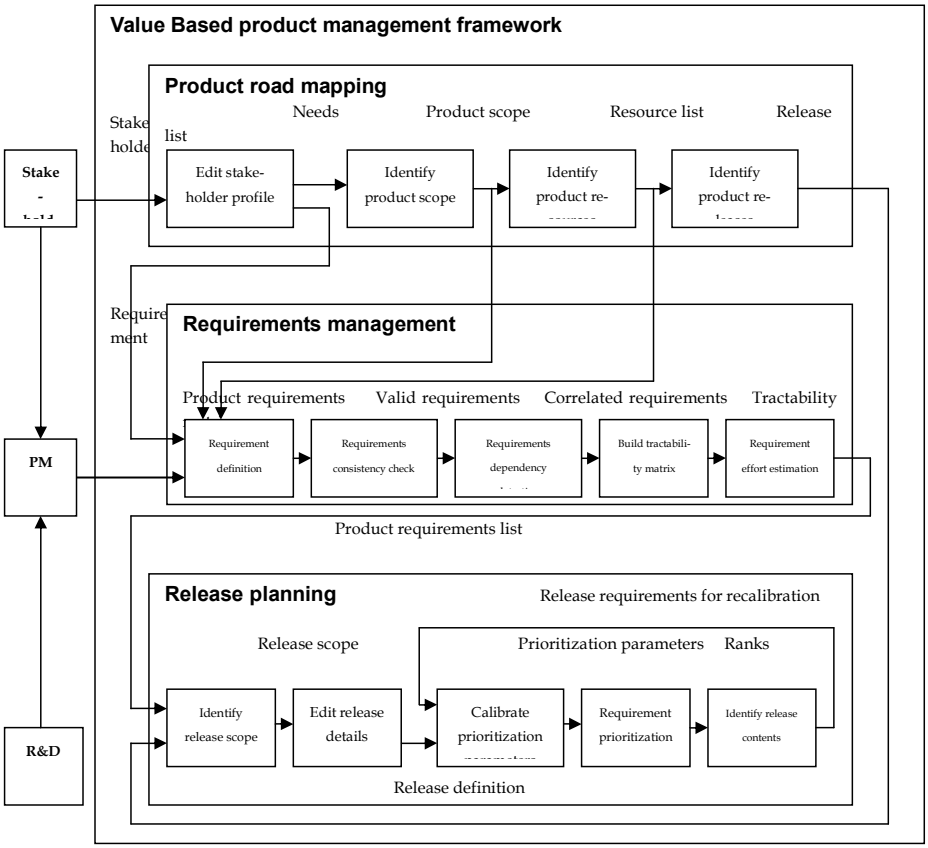

Figure 1 Value Based Requirements Management Framework architecture

5- FRAMEWORK BASIC FEATURES

The basic features for the framework architecture (shown in figure 1) can be summarized as follows:

1- Requirements identification feature through assigning a unique ID for each requirement. This can be used to unambiguously refer to the requirement during the product life cycle. Other requirement information like requirement name, description, owner name who requested the requirement, requirement status throughout the product life cycle, requirement type, start and end dates for actual implementation and importance weight (to show how it is important to the stakeholder) can be all defined through the requirement "General" attributes. These are all means to help stakeholders to get better management over the information complexity.

2- Requirements dependency detection feature is enabled through the requirement pre and post conditions. Product manager can make any pair of requirements dependant on each other by having the pre-condition for one of the requirement equal to the post-condition for the other one. The ability to detect the dependency and correlation between requirements enables the product manager to manage efficiently the release requirements because it's seldom possible to have a release with all singular requirements, as indicated by a survey of a group of organizations' requirements repositories prove that only 20% of the release requirements are independent [26] .

3- Requirements traceability feature between the source code modules and requirements specification is defined through "Tractability" attributes. This feature will enable the product manager to trace any missing development components and help him monitor the implementation progress through the requirement modules tab. A requirement is fully implemented if all the modules that implement the functionality of the requirement have a status of FINISHED. This will enable the product manager to check why a certain feature in product under development was changed or has not been fully satisfied [1].

4- Requirements estimation feature, through which the product manager will be able to estimate the effort associated with each requirement by knowing the different development modules that compose the requirement from the development team. This way the product manager will be able to efficiently plan the project schedule, review and negotiate the estimates from the product development team.

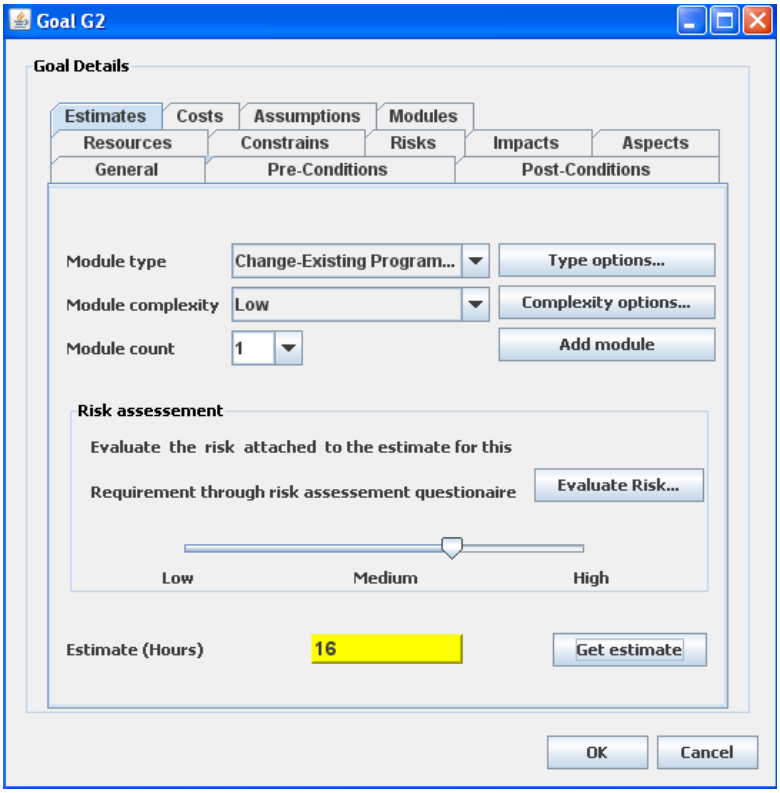

Figure 2 Requirement estimate feature

Along with this feature the framework also provides the product manager with a risk assessment for each requirement through a questionnaire maintained (Add/ Remove questions) by the product manager. Having requirement risk, value taken from the risk assessment (shown in figure 2) and the different modules types and complexities that compose the requirement as arguments to the estimate process will make the estimated requirement effort reflect the best development effort. This value will be taken as guidance through the product planning phase [21].

5- Requirements maintainability feature through different criteria like cost, importance, risk, impact and market-related aspects. This feature will enable the stakeholder to identify the different types of costs, risks, impacts and market-related aspects that influence each requirement and how much this type will impact the requirement implementation. In this manner the product manager along with other stakeholders will have the ability to control the different factors and aspects that may affect the requirement implementation from their point of view through assigning different weights for each criteria type.

6- Access control feature which controls the different access rights for the different stakeholders because product manager privileges differ from those of the

stakeholder's since the product manager is the product owner and must have full authority over the product. One of those privileges that only the product manager has is the ability to maintain (Add/Remove) the stakeholder's profile and assign a weight on scale from 1 to 9 for each one based on how this stakeholder is important to the project. Stakeholders will only have the privilege to enter their input values for each requirement to be taken into consideration through the requirements prioritization phase.

7- Resource pool feature, through which the product manager will enter the product resources who will be assigned tasks through the entire product life cycle since not all releases can have the same resources. By this means, the framework gives the product manager the flexibility to assign different resources to each product release based on the release needs.

8- Resource over release feature. This feature enables the product manager to quantify the resource allocation percentage over the whole product release based on the resource allocation type assigned to him. By this means, the framework will provide the product manager with the ability to best utilize the resources throughout the product life cycle.

9- Resource count per release feature. This feature supports the product manager with the ability to monitor the current number of FTE (Full Time Equivalent) resources over the whole product release and compare it against the planned value to be able to take the proper corrective action.

10- Cost per release feature. This feature supports the product manager with the ability to monitor the current release cost and compare it to the planned value to be able to take the proper corrective action.

11- Constrains and assumption feature through which the stakeholders will be able to assign different constrains and assumption to each requirement that will be taken into consideration while prioritizing the requirements.

12- Requirement hierarchical feature that arises from the requirement tree structure which provides the product manager with the look-and-feel of the requirement parent/child relationships as (shown in figure 5). This feature makes the ability to add, remove and modify the requirement a very simple task (just highlighting the requirement in the tree structure). By this means the process of maintaining (Add/Remove/Modify) the requirements tends to be very easy and not time consuming.

13- Requirement tip feature that gives the product manager the flexibility to know the requirement information like requirement name, client name, status, start and end dates by just highlighting the requirement in the requirement tree.

14- Requirement query feature. This feature supports the product manager with a utility to be able to use built-in queries to identify those requirements that satisfy the selected query parameters. By this utility the product manager will be able to retrieve easily the different information from the requirement repository

which will make the process of requirement management much easier.

15- Requirements consistency checking. This feature enables the product manager to detect any inconsistencies in the input requirement information as early as possible and avoid any misleading faults by guiding the stakeholder to the correct information or process that he should follow [23].

16-Requirement prioritization feature which acts as the core engine for the whole framework. It takes the inputs from the different stakeholders along with the product manager and provides the best candidate set of requirements for each product release [17]. The prioritization algorithm used in this process is the HCV "Hierarchical Cumulative Voting" [3]. The idea behind HCV is basically the same as behind CV "Cumulative Voting"; to use an open and straightforward technique to quantify the importance of requirements based on multiple criteria like Importance, cost, risk, impact and market-related aspects. As in CV, the prioritization is conducted by distributing points between requirements. However, when prioritizing with HCV, not all requirements are prioritized at the same time. Instead, prioritizations are performed at different levels of a hierarchy, and within different blocks of requirements in the hierarchy. The strengths of this technique will increase as the number of requirements grows in comparison to techniques like CV and techniques based on "flat" pair-wise comparisons. When the number of requirements grows, the need for a structured approach becomes more essential. HCV provides that structure by using natural relationships between requirements to perform the prioritization in a number of consecutive steps, and hence limits the number of requirements prioritized at a time [20].

17- Criteria selection feature. This feature provides the product manager with the flexibility to control which criteria (Importance, Cost, Risk, Impact, and Market-related aspects) will be taken into consideration while performing the requirements prioritization. The product manager by this feature will be able to understand how much each criterion affects the prioritization results and in turn the release candidate requirements [30].

18- Percentage per requirement feature. This feature provides the product manager with the ability to identify the effect in terms of percentage on how each requirement affects the selected feature (Importance, Cost, Risk, Impact and Market-related aspects).

19- XML save and restore feature. By this feature the product manager has the ability to save/restore the whole requirement tree structure, all the stakeholder profile information, and release details along with the resource pool data and all the other prioritization and feature settings. The framework will also have the ability to automatically load the last tree structure and settings before exiting the framework. This will make the save/restore process much faster.

20- Prioritization options feature. By this feature the product manager can control the different prioritization parameters. These parameters (shown in figure 3) are designed to control the following parameters:

Figure 3 Requirement prioritization options

- To control how the different criteria like importance, cost, risk, impact and market-related aspects will influence the prioritization process through assigning different weights to each criteria.
- To identify which requirement level will be used through the prioritization process since, as indicated in the previous paragraph, the HCV prioritization is done at a different level of the hierarchy. The default is to prioritize the leaves which are at level zero.
- To enable/disable the requirements dependency checking, because, once this option is selected, the prioritization process will take into account the dependency between requirements (e.g. any two dependant requirements will be given the same rank otherwise this will be neglected and all the requirements are treated as independent).
- To enable/disable constrains checking while prioritizing the requirements. If this option is selected, the requirement constraints will be dominant over the requirement assigned value and will be taken into

consideration while selecting the best candidate release requirements, otherwise the constrains will be neglected.

- To select the resource allocation model which will be used over the release and which is used also to identify if a certain resource is free or busy through the release. Free allocation model refers to that model on which each resource is assigned to only one task over the whole release. Therefore it will be always busy and allocation percentage for him will be always 100%. Full allocation model refers to that model on which the resource can be assigned to different tasks per release but only one task at a time, so, once the resource finishes the assigned task, the product manager can assign another task to him within the same release and hence the allocation percentage will depend on the percentage of his time to the total release time allocated through tasks.
- To control the resource number of working hours per day. This option can be used while calculating the resource allocation percentage over the whole release time.
- To control the resource number of days per month. This option can be used while calculating the resource allocation percentage over the whole release time.

To identify the weight for each criteria (Cost, Risk, Market-related aspect, Impact) and feature, such as a “Performance” impact feature. This option can be used to control the effect of each feature on the whole criteria.

6- CASE STUDY

6-1 CASE DESCRIPTION

In order to show the practical benefits from our proposed framework, an industrial case study has been analyzed. The case study is related to the project for evaluation and implementation of a new software system for low and medium energy physics experiments called GO4 in GSI systems. The project will cover data acquisition, data analysis, and setup control in on-line and off-line mode. The new object oriented system will be available on various platforms like UNIX and Windows NT, and it will also be based on software currently available like CERN software packages ROOT or LHC++ but it would require several extensions to handle specific user’s requirements of the low and medium energy physics experiments.

A stepwise process for the new system development based on the requirements from seven stakeholders will be illustrated in the following paragraphs. The stakeholders in this project have different importance to the organization and to the whole project. This relative importance will be controlled through

assigned weights described in each stakeholder profile. The product development is divided into three releases for six months each and with a total of fifteen resources per release. The steps will show the whole process starting from editing the release details and ending with the identification of the each product release best candidate requirements. The number of requirements for the case product is around one hundred requirements that entail all the product phases. The criteria values for each requirement have been requested from each stakeholder. These values reflect each stakeholder view point in terms of weighted criteria aspects. These aspects will control the final requirement priority from the stakeholder point of view

6-2 CASE ANALYSIS

In the following paragraphs, we will provide a detailed process of how the best release candidate requirements have been identified based on the stakeholders' inputs.

- *Step 1: edit the product releases details* as (shown in figure 4), by adding the product name, release name, release number, number of resources, release planned start and end dates, total release cost and number of resources. In our case we have three consecutive releases, started by the "Alpha/1.1" then "Beta/1.2" and ends with the "Release Candidate/1.3" which is equivalent to the final product release. The Alpha release had a planned start date for July 2006, Beta release had a planned start date for January 2007 and Release Candidate had a planned start date for July 2007. The fifteen resources who allocated for each release are distributed as follows: eight development engineers for the coding and implementations tasks and separated into two groups, two QA engineers for testing and verifications tasks such that one allocated for each group of developers, one business analyst for business case identification, one software architect for product architecture tasks, two team lead engineers for mentoring and supervision tasks so that one is allocated for each group of developers and a product manager.

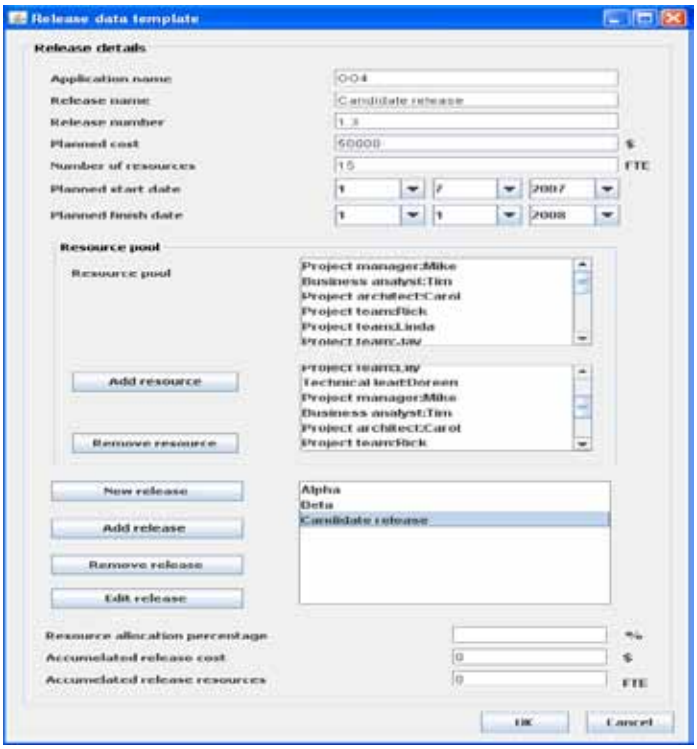

Figure 4 Step 1: Edit release details

- *Step 2: edit product requirements as (shown in figure 5), the product manager will add the requirements details like requirement name, type, description, client name, requirement number, planned start and end dates, risks, impact, aspects, costs, constraints, assumptions, modules, pre and post conditions and resources information. The product manager along with other stakeholders will enter their opinions through weights for each criteria (e.g., Importance, Cost, Risk, Impact, Market-related aspects) aspect for each requirement on scale from 1 to 9. Each stakeholder can enter different aspects for each criteria based on his point of view for each requirement. He will also need to enter the associated weight on scale from 1 to 9. These criteria aspects weights for each requirement will control the final requirement priority based on the weight assigned to each stakeholder on scale from 1 to 9 that reflects his relative importance to the organization and the project. The product manager will also have the ability to estimate the development effort required for each requirement as (shown in figure 2). All these requirements information will be used while prioritizing the best candidate for each release based on the total release cost, planned start and end dates and number of resources.*

Figure 5 Step 2: Edit requirements details

- *Step 3: prioritize input requirements. The results from this step as (shown in figure 6) will depend on each requirement' details, total release cost, planned start and end dates and number of resources. The product manager should initially set the prioritization options as (shown in figure 3) to setup some prioritization parameters like resource allocation model, dependency enabled option, constraint enabled option, resource number of hours per day, resource number of days per month, general criteria weights for the relative importance of each requirement criteria of risks, market-related aspects, impacts, cost and importance [32]. For this case study, the product manager set the resource allocation model to "Full allocation" to best utilize the resources over the product life cycle. Both dependency-enabled and constraint-enabled options are set to take the effect of both requirements interdependency and constraints while prioritizing the requirements. Resource number of hours per day is set to eight hours and number of days per month to 22 days. Table 1 will show the different criteria weights as agreed by the product stakeholders. These weights will control the share by which each criterion will contribute in calculating the requirement priority.*

Table 1 General criteria weights

Criteria	Importance	Risk	Cost	Impact	Aspects
Weight	9	2	7	3	6

- *Step 4: parameters calibration to adjust the prioritization parameters to get the best results that match with the stakeholder expectations. This can be handled by adjusting both the core business values weights for the different criteria (to reflect the relative importance of business value) and also through adjusting the general weights (shown in table 1) of the different criteria types.*

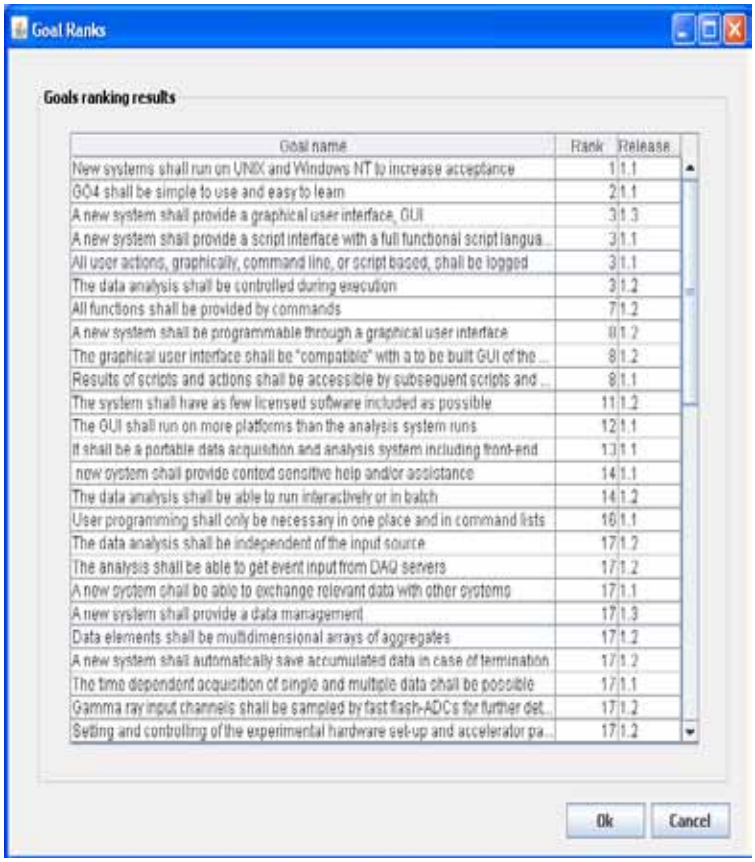

Figure 6 Step 3: Prioritize requirements

7- CONCLUSION

In this paper, we have presented work in progress in our research product in which we are developing a framework for managing software product development. This framework evaluates and selects the best candidate requirements not only based on the value/cost criteria as done in CaliberRM and ReleasePlanner but it also takes into account the effect of Impact, Risk and market-related aspect criteria. This provides the product manager with a utility to take all the aspects that affect the requirement into account while selecting the best candidate requirements for product release and it helps the product manager handle the different release planning challenges over the product life cycle. The proposed framework can balance between the different stakeholders priorities and incorporate the different viewpoints altogether while selecting the different releases requirements. It also facilitates to deal with complex systems with large number of requirements with higher complexity; this was too hard to be handled with the ad-hoc methodologies. The framework takes the different resources constraint as budget, time and resources into account while managing the product releases. It also helps the product manger not to spend more time in re-planning activity which is the nature of the evolving systems.

Our future work targets are twofold: first is to overcome the different drawbacks we have in our framework like, performance enhancement for those products have more than 500 requirements. Enhancing the calibration process for the criteria weights to get the best match is necessary because the current process requires a lot of iterations. Second, plan on adding more features to the framework to target the different challenges of managing software product development in current market-driven environments. One of these features we intend to add is to enable the stakeholder to develop their own scenarios and develop what-if analysis as a means of developing strategies. The other feature we would like to add is the generation of a set of alternative release plans that reflect the optimal solutions with respect to a weighted objective composed of value, cost, risk and profit. By this means the proposed Criteria-Based Software Product Management Framework, which integrates several software product management areas, takes the advantage of the value-based approach of software engineering to ensure stakeholder satisfaction and product success [10].

ACKNOWLEDGMENT

The authors wish to thank Hewlett Packard Corporation for supporting our research with a lot of resources that help us achieve our research objectives.

The authors would like also to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] L. Almefelt, “Requirements-Driven Product Innovation – Methods and Tools Reflecting Industrial Needs”, Chalmers University of Technology, Ph.D. Thesis Series No. 2400, Department of Product and Production Development, Chalmers University of Technology, Sweden, 2005.

[2] P. Berander, K. Khan, , and L. Lehtola, “Towards a Research Framework on Requirements Prioritization”, Proceedings of the Sixth Conference on Software Engineering Research and Practice in Sweden(SERPS’06), Umeå, Sweden, pp. 39-48, 2006.

[3] P. Berander, and P. Jönsson, “Hierarchical Cumulative Voting (HCV) - Prioritization of Requirements in Hierarchies”, accepted for publication in International Journal of Software Engineering and Knowledge Engineering (IJSEKE) - Special Issue on Requirements Engineering Decision Support, World Scientific Publishing Company, Singapore, pp. 76-84, 2006.

[4] P. Berander, and A. Andrews, “Requirements Prioritization”, in Engineering and Managing Software Requirements, ed. Aurum, A. and Wohlin, C., Springer Verlag, Berlin, Germany, pp. 69-94, 2005.

[5] P. Berander, “Prioritization of Stakeholder Needs in Software Engineering - Understanding and Evaluation”, Blekinge Institute of Technology Licentiate Series No. 2004:12, Department of Systems and Software Engineering, Blekinge Institute of Technology, 2004.

[6] Borland CaliberRM 2007 main features URL: http://www.borland.com/us/products/caliber/rm.html.

[7] P. Carlshamre, B. Regnell, “Requirements Lifecycle Management and Release Planning in Market-Driven Requirements Engineering Processes”, 11th International Workshop on Database and Expert Systems Applications (DEXA’00), p. 961, 2000.

[8] A. Dver, , “Software Product Management Essentials – A Practical Guide for Small and Mid-sized Companies”, Anclote Press, Tampa, FL, 2003.

[9] D. Greer, and G. Ruhe, “Software Release Planning: an Evolutionary and Iterative Approach”, Information and Software Technology, 46(4), pp. 243-253, 2004.

[10] R. Faulk, D. Harmon, and D. Raffo, “Value- Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering”, Proceedings, First International Conference on Software Product Line Engineering, pp. 205-223, 2000.

[11] R. Glass, I. Vessey, and V. Ramesh, “Research in Software Engineering: an Analysis of the Literature”, Information and Software Technology 44, pp. 491– 506, 2002.

[12] S. Heinonen, “Requirements management tool support for software engineering in collaboration”, University of Oulu, Department of Electrical and Information Engineering. Master’s Thesis, 2006.

[13] N. Hill, J. Brierly, and R. MacDougall, “How to Measure Customer Satisfaction”, Gower Publishing, Hampshire, England, 1999.

[14] M. Hoffmann, N. Kühn, M. Weber, M. Bittner, “Requirements for Requirements Management Tools”, In: 12th IEEE International Requirements Engineering Conference (RE´04) pp.301-308, 2004.

[15] A. Kappel, “Perspectives on Roadmaps: How Organizations Talk About the Future”, Journal of Product Innovation Management, 18(1), pp. 39-50, 2001.

[16] L. Karlsson, A. Dahlstedt, B. Regnell, J. Natt och Dag, and . Persson, “Requirements Engineering Challenges in Market- Driven Software Development - An Interview Study With Practitioners”, Accepted for publication in Information and Software Technology: Special Issue on Understanding the Social Side of Software Engineering, pp. 588-604, 2007.

[17] J. Karlsson, “Software Requirements Prioritizing”, Proceedings of the Second International Conference on Requirements Engineering (ICRE’96), Colorado Springs, CO, pp. 110-116, 1996.

[18] J. Karlsson K. Ryan “A Cost-Value Approach for Prioritizing Requirements”, IEEE Software, 1997, vol. 14, No. 5, pp 67 – 75, 1993.

[19] L. Karlsson, Björn Regnell, “Introducing tool support for retrospective analysis for release planning decisions”, Proceedings of the 7th International Conference on Product Focused Software Process Improvement (PROFES’06), Amsterdam, the Netherlands, June 2006, pp. 19-33, 2006.

[20] L. Lehtola, and M. Kauppinen, “Suitability of Requirements Prioritization

Methods for Market-Driven Software Product Development", Software Process Improvement and Practice (SPIP), 11(1), pp. 7-19, 2006.

[21] L. Lehtola, M. Kauppinen, and S. Kujala, "Linking the Business View to Requirements Engineering: Long-Term Product Planning by Roadmapping", Proceedings of the, 13th International Conference on Requirements Engineering, Paris, France, pp. 439-443, 2005.

[22] J. Nicholas, "Project Management for Business and Technology - Principles and Practice", 2nd Edition, Prentice Hall, Upper Saddle River, NJ, 2001.

[23] N. Nurmuliani, D. Zowghi, and S. Powell, "Analysis of Requirements Volatility during Software Development Lifecycle", Proceedings of the 2004 Australian Software Engineering Conference (ASWEC '04), Melbourne, Australia, pp. 28-37, 2004.

[24] F. Moisiadis, "The fundamentals of prioritizing requirements", Proceedings of the Systems Engineering, Test and Evaluation Conference, Sydney, NSW, Australia, pp 108–119, 2002.

[25] PMBOK, A Guide to the Project Management Body of Knowledge, first ed., Project Management Institute, Pennsylvania USA, 2001.

[26] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell and J. Natt och Dag, "An industrial survey of requirements interdependencies in software product release planning", Proceedings of the 5th IEE international Symposium on Requirements Engineerin, PP. 84-91, 27-31, August 2001.

[27] S. Robertson and J. Robertson, "Mastering the Requirements Process", Addison-Wesley, Harlow, England, 1999.

[28] G. Ruhe and A. Ngo-The, "Hybrid Intelligence in Software Release Planning," International Journal of Hybrid Intelligent Systems, Vol. 1(2004), pp. 99-110, 2004.

[29] R. Smith, J. Azar, and D. Cordes, "A Value-Oriented Approach to Requirements Prioritization", tech. report, Dept. of Computer Science, Univ. of Alabama., vol. 24, no. 1, pp. 32-37, 2007.

[30] Standish Group, CHAOS Report, (2007): www.standishgroup.com

[31] Wiegers K., "First Things First: Prioritizing Requirements", Software Development, vol. 7, no. 9, www.processimpact.com/pubs.shtml#requirements, 1999.

[32] C. Wohlin, and A. Aurum, “Criteria for Selecting Software Requirements to Create Product Value: An Industrial Empirical Study”, in Value-based Software Engineering, ed. S. Biffl, A. Aurum, B. Boehm, H. Erdogan, and P. Grünbacher, Springer Verlag, Berlin, Germany, pp. 179-200, 2006.

[33] C. Wohlin, and A. Aurum, “What is Important when Deciding to Include a Software Requirement in a Project or Release?”, Proceedings of the 2005 International Symposium on Empirical Software Engineering (ISESE 2005), Noosa Heads, Australia, pp. 246-255, 2005.

[34] I. van de Weerd, S. Brinkkemper, S. Nieuwenhius, R. Versendaal, J. and L. Bijlsma, “On the Creation of a Reference Framework for Software Product Management: Validation and Tool Support”, Proceedings of the First International Workshop on Software Product Management, Minneapolis/St. Paul, MN, pp. 3-12, 2006.

[35] V. der Hoek, “Software release management”, Proceedings of the Sixth European Software Engineering Conference together with the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering. Springer: Heidelberg, Germany, pp. 159–175, 1997.