

# Manage your Requirements, Manage your Project

Siva Moorthy

*Siemens Communication Software, Bangalore, India*

*Member, IEEE*

*sivamoorthy@ieee.org*

## Abstract

*Project Management and Requirements Engineering are often seen as two different and often non-intersecting disciplines to meet the common objective of satisfying the customer's needs. The paper describes how requirements resulting from a well established and sound Requirements Engineering process can be used by Project Managers as an effective tool to manage their projects. This approach highlights how requirements form the thread that weaves together the different aspects of project management.*

## 1. Introduction

In today's technology driven rapid paced business environment there is a constant pressure to deliver complex customer needs in ever shortening time-to-market time frames.

Two disciplines that address these often conflicting demands are Requirements Engineering and Project Management.

One of the most critical tasks the project manager performs is taking right decisions and steering the project. For this, correct information seen in the light of stakeholders perspective, is extremely crucial.

Apart from this, the various aspects of a project that Project Managers need to tackle are Size and Complexity, Effort Estimation, Quality, Project scope change, Progress tracking, Fault & Test Case Tracking and Project Risk Management.

Traditionally, project management tackles these issues in ways that relegates a minor role to requirements management. The aspects of project management are handled in ways that rarely links them together. On the other hand, requirements engineering is largely localized to the analysis phase and traceability resulting in lending very little support to the actual act of project management.

This limited use of the power of requirements by projects and project managers has prompted this work. This work aims to move from multi-perspective project management to requirements driven project management.

The paper describes the requirements driven project management paradigm and the practices followed in the company. This approach has been used in a broad spectrum of projects. It has met with substantial success and support from stakeholders.

The first part of the paper discusses the details of the requirement engineering process followed. In the second section, the requirements management approach to the project management aspects is discussed. The paper concludes with scope for further investigation.

## 2. Requirements Engineering – the foundation

The foundation of requirements driven project management is well formulated and well documented requirements.

This foundation includes requirements definition (elicitation, collation and classification of the needs of the different stakeholders of the project) and requirements management (traceability and change management).

### 2.1. Requirements Definition

The steps followed in the requirements engineering process:

Step 1: Have a requirements repository with the critical information of:

- a. Requirement (as stated by the stakeholder)
- b. The source of the requirement or the stakeholder who asked for the requirement – this

information is crucial when there is a need to clarify requirements or resolve conflicting requirements

c. Priority as stated by the stakeholder

Step 2: Group requirements based on high level use cases. Additionally, the requirements are broadly classified as functional (requirements that directly relate to the features of the system under consideration), non-functional (requirements that describe the operational characteristics of the system, for example performance) and non-technical (requirements that deals with the activities and deliverables of the project, for example schedule, effort)

Step 3: Prioritize the grouped requirements using Kano analysis (or any of the standard prioritization techniques)

Step 4: Prioritize individual requirements in the group using any standard prioritization techniques

Step 5: Resolve conflicting requirements. Requirements from different stakeholders could conflict with other requirements, with requirements from other stakeholders or in the priority (as stated by the stakeholders). It is important to resolve the conflicting requirements either by discussion or by the relative weight of the stakeholders involved in the elicitation process.

Step 6: Establish that every requirement satisfies the basic quality of a 'good requirement' as stated in the IEEE Standard 830-1998. A requirements checklist is used to ensure compliance of requirements to the standard.

Step 7: Clarify the unclear requirements that result from the requirements quality check mentioned in Step 6.

Step 8: Populate the project risk database with the requirements that remain open at the end of Step 7. Associate the risk to requirement(s) identifiers.

Step 9: The result of the grouping and prioritization should be made available to all the stakeholders. The final version of the requirements set should be base-lined. Any change to the base-lined requirement set is to be treated as a change request.

Step 10: Arrive at the acceptance criteria for the project. This is essential to prevent any road blocks before the project is shipped to the customers

## 2.2. Requirements Management

Once the elicited requirements have been classified, prioritized and captured in a repository, the next step is to break down the requirements into manageable traceable entities and provide the metadata for the

requirements. The following activities need to be performed.

Step 1: Requirement Identifier:

The traceability matrix forms a crucial part of the paradigm of requirements driven project management.

While designing the traceability matrix it is essential to define a uniform requirements identifier format (along with its textual description). It is essential to define a project wide (or organization wide) format that would be followed to represent analyzed and broken down requirements. A 5-level requirements identifier is sufficient to enumerate requirements keeping it compact as well as descriptive. The format of A.B.C.D.E (where A – E are placeholders for the levels of requirement breakdown) is used to represent requirements. An example placeholder format could read <Project or System Requirement>.<Requirement Group>.<Feature>.<Sub-Feature>.<Component> where component is the artifact that physically implements the requirement. The number of characters for each placeholder should be restricted to 6 characters each so as to make the requirement identifier readable. The advantage of this format is that it supports requirement level abstraction. This requirement identifier is then used throughout the project scope to represent the requirement at its various levels. The code or the part of implementation would use the full requirement identifier format whereas a high level document may restrict itself to the first two or three levels of requirement breakdown. The system tester testing system level requirements may not be interested in the requirement that has been broken down to the last level E. For the tester an identifier of A.B.C could suffice.

Step 2: Requirements Size:

One aspect that plagues requirement breakdown is the granularity of requirement. It is the aim of the requirements breakdown activity to have equi-granular or comparable-granular requirements. The aspect of traceability matrix that addresses this issue is the size of the individual requirement. This forms a part of the requirement metadata.

According to the sizing technique recommended by the organization, it is essential to give a first cut size estimate of the individual requirement. This has two dimensions: The size measurement variants and the inter-size variant translation formula.

The first dimension: The requirement metadata should have a provision to represent the size of a requirement in the form of Lines of Code (LOC), number of test cases required to test the requirement, estimated unadjusted implementation effort. It could also have Use Case Points or Function Points as a measure of size. The philosophy is to have a

mechanism where the size of requirement can be captured. The reason why a standard size estimation technique cannot be used is that requirements may be of different types. Documentation requirement is one such requirement. This can be estimated only with the documentation effort or the number of pages. If the requirement is a testing requirement (for example performance requirement), the size of that requirement can be made only with the number of test cases that would ensure that the requirement is satisfied. In this case the number of test cases required to meet the requirement is an indication of the size.

The second dimension of the requirements size data is the translation. When a group of requirements have been sized with different parameters, it is very difficult to compare the two requirements. What is required is a project specific or an organization wide translation formula based on history data (depending on the type of project and the domain of the project). For example, 150 C++ LOC would be equal to 2 system test cases would be equal to 8 hours of effort would be equal to 2 Use Case Points. This kind of translation ensures that the requirement in the group or across the project is comparable. For example, the size of the two requirements “It shall be possible to back up across the network” (a LOC size estimate) and “The backup of data should not exceed 20 min” (a test case size estimate) can be compared with the above translation data. It should be ensured that the requirements breakdown is done in such a manner that individual requirements are comparable. This enforces that the requirements are not unevenly broken down and also ensures that the complex requirements are broken down into smaller requirements blocks.

Step 3: Requirements Tracing:

Analysis of requirements should also indicate the physical location where the result of requirements breakdown will be captured and stored (usually the documentation specification or code that will implement the requirement). This ensures traceability of the requirement in the analysis, design and implementation phases.

Step 4: Requirement Identifier to Test Case mapping:

To ensure that the requirements are tested and traced to the testing phase, there needs to be a mapping between the requirements and the test cases that would be written to validate the implementation of the requirement. A matrix should be maintained (for both module and system test cases) showing the mapping between the requirement and the test cases.

Step 5: Requirement Identifier and Change Requests:

Any changes to the low level requirements (once base-lined) should be treated as a change request. The change request repository should have the information of the stakeholder initiating the change request, the requirement that is impacted and the type of change – an addition of a requirement, a modification of a requirement or a deletion of a requirement

Step 6: Requirement Identifier to Use Case mapping:

In addition to the above, the correlation between the system use case (used during elicitation or during the finalization of the stakeholder requirements) and the low level requirement identifier should be mapped. This is essential to assess the impact of the change on the low level requirement(s). The change request from the stakeholder usually happens at the use case level and the impact of this change should be known to the implementation community at the requirement identifier level. This is achieved by maintaining a matrix of use cases and low level requirements. Any change in the use case would indicate the affected requirements. In addition, any change in the requirement would highlight the use cases that will be impacted. This information is critical to the system testers, stakeholders and the documentation team.

The above mentioned activities and outputs like requirements repository, requirements breakdown, traceability and requirements metadata ensures a rock solid foundation on which the full potential of requirements can be used to manage projects. The next section covers how these well formulated requirements can help project managers manage their projects.

### **3. Requirements and Project Management**

Project management has various aspects or dimensions. In this section it is shown how the different aspects like Size and Complexity, Effort Estimation, Pro-active Quality check, Project scope volatility, Progress Tracking, Fault & Test Case Tracking and Risk Management can be managed with requirements.

#### **3.1. Size and Complexity**

The first and foremost aspect of Project Management is the size of the project. This is essential to schedule the project, plan resources and to give an indication as to what is at hand.

The requirements metadata is the starting point to determine the size of the project. The rolled up figure available as a part of the requirements metadata gives an indication of the size of the project. The breakdown

of requirements into comparable-granular requirements ensures that the complexity factor of the requirements from the customer is taken into consideration. With the multi-parameter translation formula in place, it is also possible to indicate the size of the project in terms of effort or LOC.

### **3.2. Effort Estimation**

In requirements driven project management approach, the effort estimation is a 'size and cost driver' driven process. This takes the size of the project as the input and is transformed into effort by the prime cost driver – the number of requirements. The number of requirements has a direct impact on the number of test cases and the effort to execute the same. The constant effort (for example, documentation) also depends on the number of requirements. The systematic requirements breakdown and the requirements repository provide input to the effort estimation process. The breaking down of requirements into comparable manageable entities ensures that the complexity and/or improper breakdown of high level requirements do not distort the effort estimates.

### **3.3. Quality**

Requirements driven project management mandates test cases are identified and associated with the relevant requirement identifiers. This gives the project manager an assurance that the requirements are traced to the testing phase and also tested. There exists a m:n relationship between requirement identifiers and test cases. For example, one requirement can be tested by multiple test cases and one test case can cover multiple requirements. The two-way mapping between requirements identifiers and test cases is a valuable tool to the project manager.

As the requirements management process ensures that the requirements are of comparable granularity, the test cases for each of the requirements should also be comparable. The number of test cases per requirement identifier ensures that sufficient test case coverage exists. In case the number of test cases per requirement identifier is way above or below the average (as defined in the project or organization), the test case identification and test strategy can be appropriately modified.

The reverse mapping of number of requirement identifiers covered by a test case is an indication of the quality of the test case. The numerical limit for the number of requirements covered under a test case

would vary from the phase for which it is being used. The module test case to requirement identifier mapping would be low (1 or 2 requirements per test case). The system test case to requirement identifier would have a higher number. It is left to the project manager to define the numerical limit to the metric. Too many requirements covered by a test case would mean that the requirement is not being adequately tested by the test case. Too few would translate into too many test cases to adequately test the requirement.

This approach overcomes the limitations of traditional test case identification – coverage and requirements depth of test cases (Quality). The project manager gets an insight into the quality of testing and quality of test cases much before the testing phase and is able to take corrective action.

### **3.4. Project Scope Change**

As any project manager would agree, requirements creep is no longer an exception but an integral part of project management. The challenge is to react to the requirements creep and communicate the impact due to requirements creep as soon as possible.

The requirements management process provides a powerful tool in this regard. The mapping of system level use cases to the broken down requirements identifiers provides an instant view of impact due to a change request. This mapping and view is useful to both the development and testing community. The change request could be at two levels – either the system use case changes (inputs from stakeholders) or the broken down requirement could change in the course of software development. The mapping gives an indication as to which requirements are impacted due to the change in the system level use case. Similarly any change in the requirement gives an indication as to which system level use case is impacted. The impact can be communicated to the concerned stakeholders (as indicated and captured during the requirements definition stage).

The impact on the system level use case is also a valuable input to the system testing team which relies on the system level use cases to derive test cases. The impacted test cases can be retrieved as the requirement identifiers - system level use cases - test cases mapping has been defined and captured during requirements management.

This approach provides the project manager an instant view of the impact of the change request on the project and components. The impact matrix also ensures that the changes are communicated to all relevant stakeholders in a language that they understand.

With the established link between requirements and effort estimates, any change in requirements can be easily translated into effort. The requirements management process ensures that impact of the change is instantly assessed and the effort to accommodate the change is also calculated.

### **3.5. Progress Tracking**

Project management is incomplete without progress tracking. Traditional project tracking methods relies on a subsystem approach wherein the progress of the project is tracked and reported on the basis of the degree of completion of the implementation of the component (it could be on any part of the construction phase continuum). This approach, unfortunately, is extremely development focused and does not convey the right picture to the stakeholders. The stakeholder perspective is always software system based or is feature centric. Since the component architectural design of the software system is not usually exposed to the stakeholders, reporting progress on that basis would not add much value.

Requirements driven project management paradigm mandates that the reporting of the progress of the project be made on the requirement identifier level. The levels of decomposition of the requirement (as indicated above) offer different levels of abstraction and makes reporting at different levels possible. With this requirement breakdown, reporting can be made at a component level and can be scaled up to the project level. The stakeholders can be provided with a feature based progress status and the project team can be provided with feature-phase-component wise project status. A phase wise progress reporting would indicate how complete the phase is (for example, if 8 of the 10 requirements have been analyzed, the analysis phase is 80% complete). It is left to the project manager to assign weights based on priority of the requirement groups while reporting the progress of the project or component.

This flexibility in reporting progress of the project to different stakeholders is the key aspect of requirements driven project management. Feature based progress reporting is a very powerful tool as this gives the stakeholders an option of re-looking at the project deliverables based on the overall progress. Certain nice to have features that have been slow can be deferred to the next version. This information would not be available if the reporting was made on the basis of completion of component implementation. This reporting is also a very valuable method to indicate to the entire project community as to where the project stands with respect to the end user. This

insight is often missed when the project team gets its hands on to implementation.

### **3.6. Faults and Test Case Tracking**

Conventional progress reporting during the testing phase is also limited to reporting the test case progress. The requirements management process already provides a mapping of test cases to requirements. Reporting based on test completeness of features or use cases gives the power in the hands of the project management. Beta releases, trial releases, reprioritization of feature releases or schedule/effort revision decisions can be made easily if the progress is reported in this manner.

Also critical in this phase is the fault reports that arise out of testing. The requirements driven paradigm also mandates that every fault report be associated with test case(s) and the associated requirement(s).

The mapping of faults to test case(s) clearly indicates which of the test case(s) need to be executed again to ensure that the feature is error free. This simple mapping saves a lot of time in the identification of regression test cases for the verification of the correction.

Plotting of the faults against the requirements gives an indication of the quality of the feature. Based on this report, the project manager can take a decision on whether more testing needs to be performed for that feature to ensure its quality or if the feature can be released given the fact that it has been the source of so many errors.

### **3.7. Project Risk Management**

Risk management activities encompass identification, mitigation and risk tracking. Traditional risk reporting usually occurs at the project level.

The perspective that the requirements driven project management paradigm adds is that the risk information is classified and reported according the requirements breakdown. For example, OEM risks are associated with only those features that require the OEM. This way, risk reporting does not spread or associate the risk with the entire project. Since the requirements are broken down and represented in various degrees of abstraction, the risks with impact at different levels can be associated with different levels of the project. Risks could impact only a sub-feature or the entire group of requirements. With this association, the risk reporting to the stakeholders is more accurate. This view of risk along with the requirements based progress tracking,

gives a 'modular' view of the entire project. Decisions and actions are then more focused and also taken with the stakeholder perspective in mind.

#### **4. Conclusion**

The requirements driven project management paradigm addresses the primary needs of every project manager – gauge the project at the start, manage the project, have the correct information at hand, take quick informed decisions, keep customer focus and communicate decisions along with its impact. This method identifies well defined and formulated requirements as the single common binding force for all project management related activities.

This methodology has been successfully used in projects which range from 10 staff years to 300 staff years and that span multiple geographical locations. The project stakeholders have supported this approach and have highlighted this as one most beneficial improvement activities undertaken. This is evident from the improved results in the Project Management and Scope Definition areas in the Customer Satisfaction Survey and Internal Project Survey.

#### **5. Further Investigation**

Tools for requirements management and project management exist as independent entities (highlighting the difference that currently exists in the two domains). It has to be explored which tools can support requirements driven project management. It has to be explored if a new tool to suit this paradigm needs to be developed or if a customizable middleware can be developed that brings the best of the evolved tool sets from both the domains.

#### **6. References**

- [1] McGovern, Fergal "Managing Software Projects with Business-Based Requirements", *IT Pro*, September | October 2002.
- [2] Grâce, Benoit De, "Project and Requirements Management: A Hand-in-Hand Approach", [www.primavera.com/partners/pdf/PMConnexWhitePaper15.pdf](http://www.primavera.com/partners/pdf/PMConnexWhitePaper15.pdf)
- [3] IEEE Std 830-1998, IEEE recommended practice for software requirements specifications