DRISHTIKON
Symbiosis Centre for Management and HRD
Vol 3, No. 1

Analysis

Scope Creep, scrap & Churn are NOT SINS in Requirements Engineering

SP Rahmesh[1], B Madhavan[2]

Abstract

This article covers the entire spectrum of requirement gathering process and how it should function to handle creep, scrap or churn of the scope of the project requirements. Requirements engineering is a discipline that builds the foundation for any project to get implemented successfully. Organizations with very good processes, people and tools along with good experience, too fail when scope creep, scrap or churn takes place in the project. What's the impact on cost, schedule, resources and to business? How to efficiently handle such scenarios? Volatility of requirements is a factor should be measured and controlled using benchmark values.

Key words: Requirements Engineering, Churn, Creep, Scrap, tools, Software Requirements Specifications (SRS)

Introduction

Business Requirements: This means defining the boundaries of a project consisting of elements what's inside the scope and what's outside. Also define the subsystems which are interfacing with the new functional requirement, which will facilitate to gain a overall view of the entire system. Requirement Engineering according to Weinberg [1]

[1] Senior Project Manager, HCL Technologies Ltd.
[2] Professor, Department of Management Studies, AMET University, Chennai

means the rationale for requirements “Requirements are made for a common purpose; to convert vague desires into explicit and unambiguous statements of what customers want”

In reality, before requirements document is ready and approved by business users, development team begins with design phase and generates various questions. Such questions create new insight into business and updates previous decisions. This way the requirements get refined further.

These statements are then used to compare with what was built and what was desired. Due to some budget constraints and deadline from business users, more requirements are pressed into the same schedule / budget or part of the requirements are hewed away from initial scope. This requirement stretching, squeezing & hewing should be kept within certain boundaries. What’s that acceptable boundary? Some times requirement volatility is necessary and healthy for application. But perpetual change will drive project management crazy, developers mad and sponsors disappointed as their cost increases and project delayed. Jones emphasizes that although creeping requirements are troublesome, they are technically necessary. Volatility is fact of life but should be under control.

In Requirements Engineering the formula used to get compound monthly volatility rate (R) of requirements is

$$R = \left(\sqrt[t]{\frac{Sizeatend}{Sizeatstart}} - 1 \right) X 100$$

Scrap: Requirements do not always grow but may also get chopped off from the original requirements for various

reasons may be due to cut in budget or change in business rules or legal reasons.

Creep: When scope of project requirement increases with additional features or functionality is called creep.

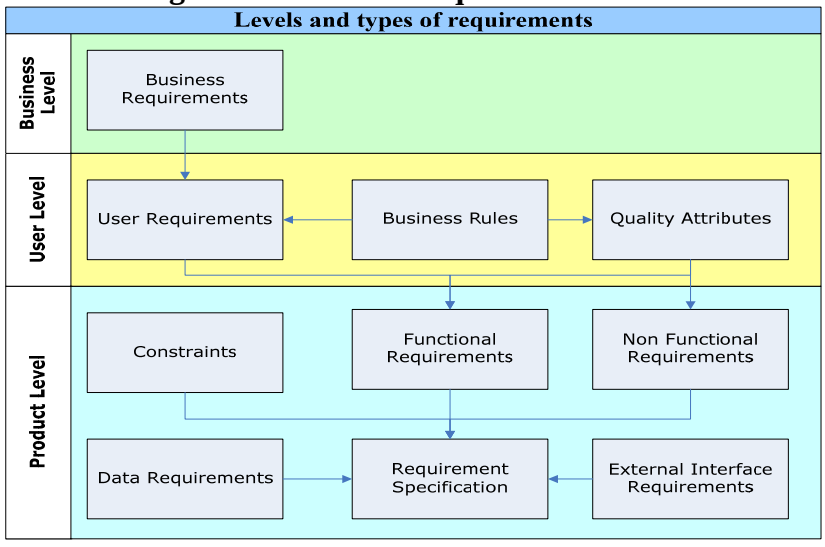
Churn: Requirement churn happens during development life cycle, it grows or reduces but not stable throughout the project. This phenomenon is call churn.

Churn Ratio – Number of base-lined Configuration Items (CIs) that have been modified and rechecked into the configuration management system over the last reporting period divided by total number of base-lined CIs in the same period.

Requirement stability will be measured by (1- (number of changes coming into the requirements/ total base lined requirements) X 100)

This ratio acts as an indicator of the requirements or of the system's soundness and stability. If the churn ratio is high it means that the system has undergone lot of changes. It could also reveal unmanageable requirements, which could indicate that the project is drifting towards disaster. Requirements change per month is calculated by dividing the number of new, changed or deleted requirements specified in the current reporting period by the total number of requirements at the end of the same period. If this percentage is high, it indicates that the customer is not sure of what he wants or the original requirements provided is too fickle or sketchy.

Figure 1: Profile of Requirements



Five W's of Software Requirements Engineering (Westfall, 2006)

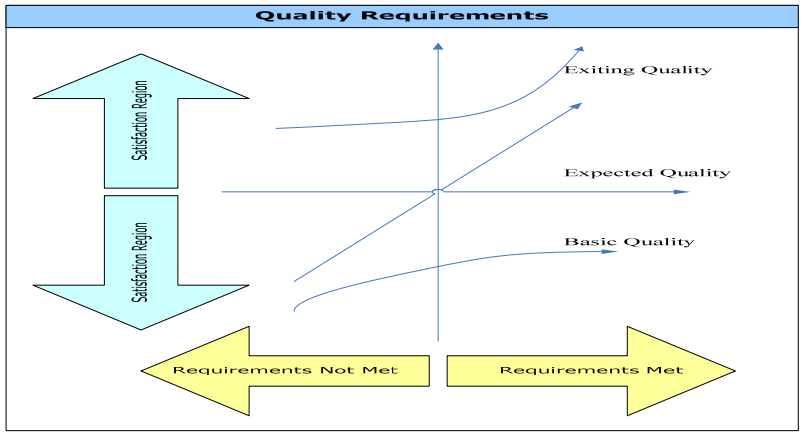
What

Requirements must be determined and agreed to by the customers, end users and suppliers of the software product before the software can be built. The requirements define the "WHAT" of a software product. What the software must do to add value for its stakeholders. The practitioners should gain better understanding of what information they need to elicit, analyze, specify, and validate when they define their software requirements.

Business level: In general the business requirements define why the software product is being developed.
User Level: Define what the software has to do so that the users must accomplish their objectives. Multiple user level requirements may be needed in order to fulfill a single business requirement.
Product Level: The software functionality must be built into the product to enable users to accomplish their tasks, thereby satisfying the business requirements. The software may be part of a much larger system that includes other components. In this case, the business and user level requirements feed into the product requirements at the system level. The system architecture then allocates requirement from the set of system requirements downward into the software, hardware, and manual operations components.

Why
The hardest part of building the software system is deciding precisely what to build followed by what for it is built and why it is built. Eliciting, analyzing and writing correct requirements are a critical part of software engineering. There are many issues that can have a negative impact on software development projects and products if practitioners don't do a good job of defining their software requirements. These issues can be incomplete requirements, lack of user involvement, requirements churn, wasted man power, gold plating and incorrect estimates.

The following chart shows the relationship between customer satisfaction and quality requirements.



The expected quality line represents those quality requirements that customer explicitly states. The exiting quality is the innovative requirements level and represents unexpected items. The basic quality levels are the requirements that a customer expects the product to have. The requirements define the scope of the products that are being developed. With out a clear picture of that scope, estimates of the project schedule cost and quality will be misleading.

Who

Stakeholders and end users are affected by the software product and therefore have a great level of influence over the requirements for that software product. There are three main categories of stakeholders as the acquirers of the software product, the suppliers of the software product and other stakeholders.

The Acquirers: This can be divided into two major groups. First there are the customers who request, purchase and pay for the software product in order to meet their business. The second is the users also called end users, who actually use the

product directly or use the product indirectly by receiving reports, outputs or other information generated by the product.
The Suppliers: Individuals and teams that are part of the organization that develops the software product or part of the organizations that distribute the software product or involved in other product delivery methods. The requirements analyst, the designers, the developers, the testers, the documentation writers, project manager and the technical support plays their respective roles for performing their duties.
The Others: The other stakeholders include legal or contract management, manufacturing or product release management, sales and marketing, top management, government or regulatory agencies, auditors and society at large.
Identifying the stakeholders and getting them involved in the requirements engineering process brings different perspectives to the table that can aid in a more complete set of requirements early in the software development life cycle.
First Step: The first step in identifying the stakeholders is to make sure that one considers all of the potential stakeholders. It is almost impossible for the development of a software product to take into consideration the needs of all the potential stakeholders, without conflicts.
Second Step: This is for the practitioners to decide how they are going to deal with these conflicts. They accomplish this by determining which stakeholders have higher priorities based on their contribution to the success of the software product.
Third Step: This involves deciding who will represent each stakeholder group. The main choices are Representative, Sample and Exhaustive.

When

Requirements Development encompasses all the activities involved in identifying, capturing and agreeing upon the requirement. The requirements engineering is iterative process where the business and user level requirements are fed into

the definition of the product level requirements. Once the software design and development team goes thru the requirements, it may uncover implicit requirements or the need for the further refinement of the business, user, and product level requirements. **Until the requirements are signed-off by business managers, it's better not to begin other phases of the project.**

How

Software requirements engineering is a disciplined, process oriented approach to the definition, documentation and maintenance of the software requirements throughout the software development life cycle.

Requirement Elicitation: This stage includes all of the activities involved in identifying the requirement's stakeholders, selecting representatives from each stakeholder class, and determining the needs of each class of stakeholders.

Requirement Analysis: Here the stake holder's needs, assumptions and other information identified are molded together and refined into further levels of details. The information gained in the analysis stage may necessitate iteration with the elicitation step so that clarifications are raised, conflicts are exploded and missing requirements are identified.

Requirement Specification: The requirements are formally documented during the specification step so they can be communicated to the product stakeholders. We should have pre defined specification template. These will help to focus on content instead of the format. A requirement tool can be used here.

Requirement Validation: This is the last step. This is to ensure that requirements are well written, complete and will satisfy the customer needs. Validation may lead one to iterate

the other steps in the requirements development process because of identified defects, gaps, additional information or analysis needs, needed clarification, or other issues. The peer reviews are very important. It is to ensure the completeness, consistency and modifiability. The peer review process should also look at each individual requirement in order to ensure that it is Unambiguous, Concise, Finite, Measurable, Feasible, Testable and Traceable. The important step in validation is to write the test cases from black box testing. This will uncover the defects in the requirement.

The Requirement Management starts with getting stakeholder buy-in to the baseline requirements. This includes the activities used for ensuring that products and project plans are kept consistent and for tracking the status of the requirements as one progresses through the software development process.

Contents of a good SRS Document (Damian and Zoughi, 2003)

1. Define the project's vision and scope.
2. Identify user categories
3. Identify appropriate representatives from the user categories
4. Identify the requirements of decision-makers and their decision-making process
5. Select the elicitation techniques that you will use
6. Apply the elicitation techniques to develop and prioritize the use cases for a portion of the system
7. Gather information about quality characteristics and nonfunctional requirements from users and define all the attributes
8. Elaborate the use cases into the necessary functional requirements and document them

9. Review the use case descriptions and the functional requirements
10. Develop prototype models if needed to clarify the elicitation participants' understanding of portions of the requirements
11. Develop and evaluate user interface prototypes for portions of the requirements that are not clearly understood
12. Develop conceptual test cases from the use cases
13. Use the test cases to verify the quality of the use cases, functional requirements, analysis models, and prototypes
14. Repeat steps 6 through 14 for the other portions of the system until the team concludes that requirements elicitation is as complete as it needs to be
15. Baseline the requirements.
16. The requirements should be finite and without any open ended questions

How to Control and Manage Requirements Engineering process?

The following key rules would help to control Creep, Scrap and Churn

1. Design Quality Software and measure Quality of Software Design
2. Design simple, flexible, traceable, testable & reliable requirements
3. Efficiently utilize Business Analysts for understanding and developing SRS documents
4. Validate all the critical parameters and characteristics
5. Conduct feasibility study or prototype model to verify the requirements incorporated risks and issues
6. Avoid confusion over requirements. Initiate queries with business to attack any confusion

7. Inform the users about vague and ambiguous requirements to get proper clarity before doing any design
8. Prioritize the requirements, prepare the dependencies and define a path to follow
9. Do not get paralyzed with vague requirements but escalate proactively to overcome
10. Develop functionalities that are really required for core business and avoid spending time and effort on no-use functions
11. Follow change management process for any addition or deletion to that will influence the schedule and effort
12. Use Version Control tools to track changes to requirements
13. Gain approval & Sign-off for any changes before incorporating the change requirements as that will influence the schedule and effort
14. Establish a User & Developer forum to work on requirements which will kill issues and clarify any confusions without any delay
15. Maintain traceability matrix for all the requirements till product development
16. Use proper templates to develop use case, SRS and scope definition documents
17. Prepare Test Cases once the requirements are frozen
18. Use Requirements Management tool to monitor changes and track till delivery

Define the Product's Business Requirements
Start documenting the requirements in an SRS template or in the form of a Use Case document. Only by documenting the requirements with correct document version history, it is possible to know whether scope creep or scrap is happening.

Get Extensive User Involvement

Multiple studies indicate insufficient user involvement as a common reason why software projects struggle and fail (Standish Group Report). Every project should identify its distinct user classes and their characteristics. Users might differ in their frequency of product use, features used, privilege levels, or skill levels. Determine which of your user classes will carry the greatest weight in priority discussions, when resolving conflicting feature requests, and in driving design choices. Find the suitable representatives who can serve as the voice of the customer for each important user requirement.

The key tasks of users and designers are

- Developing usage scenarios and use cases
- Resolving conflicts between proposed requirements
- Defining implementation priorities
- Specifying quality attributes
- Inspecting requirements documents
- Evaluating and prioritizing enhancement and change requests.

Focus on User Tasks

A use case describes a task which the user must be able to perform with a software product. The discussion on Use cases would help to transform the project from the initial feature-based focus towards the actual requirements of the user. The use case approach helps to avoid missing essential functional requirements or implementing non functionality that no one uses.

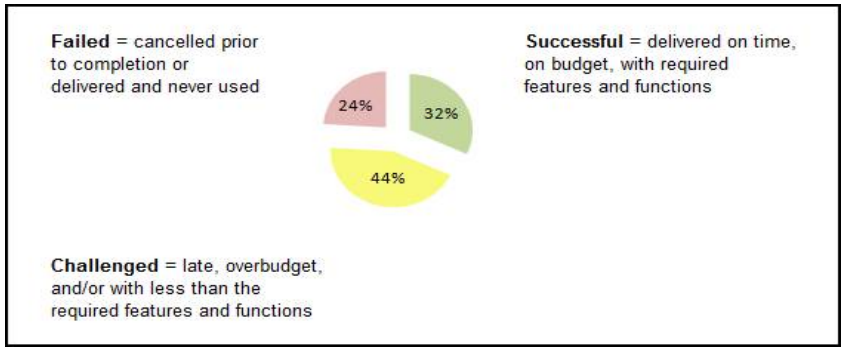
Highlights of the CHAOS Report

Note: Standish Group is an International Consultant for IT Industry, focuses on project failures and helps to succeed.

The Standish Group collects information on project failures in the IT industry and environments with the objective of making the industry more successful and to show ways to improve its success rates and increase the value of the IT investments. The latest results have been compiled into the CHAOS Report 2009 published by the organization in the month of April.

Problem: The CHAOS report measures success by only looking at whether the projects were completed on time, on budget, and with required features and functions. These are the triple parameters that were considered for measurement and it left out the other collateral parameters like quality, risk, and customer satisfaction. This shows how critical and important the Requirements engineering is, for the successful and satisfactory completion of IT projects. So keep these in mind while handling the project.

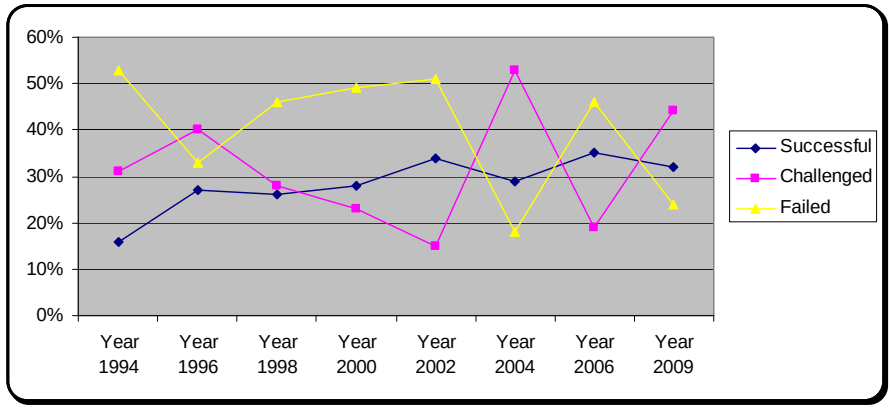
Figure 3: CHAOS Report - Pie Chart



The report shows that software projects now have a 32% success rate compared to 35% from the previous study in 2006 and 16% in 1994. On the other hand, 44% of projects were challenged (delayed delivery / cost overrun / schedule overrun and / or with less than the required features and functions) while 24% failed (cancelled prior to completion or delivered and never used).

	Year 2009	Year 2006	Year 2004	Year 2002	Year 2000	Year 1998	Year 1996	Year 1994
Successful	32%	35%	29%	34%	28%	26%	27%	16%
Challenged	44%	19%	53%	15%	23%	28%	40%	31%
Failed	24%	46%	18%	51%	49%	46%	33%	53%

Figure 4: CHAOS Report – Trend Analysis



So, must we conclude that project success is a little worse in 2009 than in 2006 (32% vs. 35%) but definitely better than in 1994 (16%)? For sure, there is better project management expertise (more certified project managers), better training, and better tools and techniques. On the other hand, project complexity and environments have increased while the time to deliver has been reduced. But still, project success in IT has improved when looking at all the many angles that are not being considered by the CHAOS Report. Nevertheless, the figures are still low and need to improve much more.

Conclusion

In summary Requirements Engineering is the foundation for the success of any software project. Business Analysts, Design Engineers and Practitioners must know the various approaches and levels of requirements to do an appreciable job of Requirements Engineering. It requires an understanding of the benefits of having good requirements so that adequate

resources, effort and time can be dedicated to the Requirements Engineering process throughout the software development life cycle [SDLC]. Doing Requirements Engineering demands an interdisciplinary approach that considers the needs of multiple stakeholders and business groups. It also requires expertise in the various skills of requirements engineering including requirements elicitation, requirements analysis, requirements specification, requirements validation, technology skills, business domain skills and above all requirements management. The Challenges on requirements engineering opens up opportunities for further research. With these arguments a competent requirements engineering team can manage the scope Creep, Scrap and Churn successfully and ensure optimum cost and resources.

References

A.J. Albrecht, "*Measuring application development productivity*", Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1979, pp. 83—92

Chaos Report on Project Failure, published by http://blog.standishgroup.com

Daniela E. Damian and Didar Zowghi, "*Requirements engineering challenges in multi-site software development organizations*", Requirements Enginnering Journal, 2003, pages 149-160

G.P. Kulk, C. Verhoef, "*Quantifying requirements volatility effects*", VU University Amsterdam, Department of Computer Science, de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands, pages 136-150

Gerald M. Weinberg, "*Exploring Requirements: Quality before Design*" 2011, Dorset House Publishing

Linda Westfall, "*Software requirements engineering: What, Why, Who, When and How*", 2006, pages 8-14, www.westfallteam.com

Wiegers, K. E, "*In search of excellent requirements. Process Impact* Web site", 2004, http://www.processimpact.com

Wiegers, K. E. 2003. *Software requirements, 2nd Edition,* Redmond, Wash.: Microsoft Press.