JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. xx. No xx (200x), pp. xx-xx

Introduction to gathering, documenting and maintaining requirements

Konrad Swirski, Ph.D.
Warsaw University of Technology
Institute of Heat Engineering
00-665 Warsaw
Nowowiejska 25
e-mail: Swirski@itc.pw.edu.pl
Remigiusz Wasliewski, M. Sc. Eng.
Transition Technologies S.A.
Software Solution Center
01-030 Warsaw
Pawia 55
e-mail: R.Wasilewski@tt.com.pl

Abstract. *This article is going to provide information about a professional method of gathering, documenting and maintaining requirements. It will be an extension of an earlier written article, which was entitled TOWARDS A REPEATABLE SOFTWARE DEVELOPMENT PROCESS. It is going to be the second in a series devoted to describing recommended ways of software development.*

1. Introduction

We have 15 years of experience in designing and developing software for different customers. We understand the tendency of the "develop document later or never" practice. It seems to be so obvious but still many well known software companies struggle to specify, document and manage requirements. There are a lot of cases which cause that kind of problems. Some of them may be recognized as:

- indecision of the customer
- lack of understanding of customer needs
- inadequate help from the customer to specify changing requirements

In our work we have encountered situations in which the customers and software developers didn't understand the word "requirement" in the same way. There is also a tendency to say that requirement management is important for a large project but could be omitted in "our case". In our opinion there is no such a small commercial project that could be successfully finished without requirements management.

It is very important to acknowledge that when it comes to software development it involves at least as much communication and collaboration as coding, implementing and testing. One of the most common mistakes made during projects is immediate coding without ever really wondering what the REQUIREMENTS are. That kind of behavior of course causes a lot of havoc in projects makes them less enjoyable to do and can often result in failure. It is not advised to emphasize the construction and transition phases of the project. You would certainly ask why? The answer is simple 40-60%* of mistakes while doing the project are, on average, made during the inception and elaboration phase. Additionally those mistakes, as we believe you know are the most expensive ones.

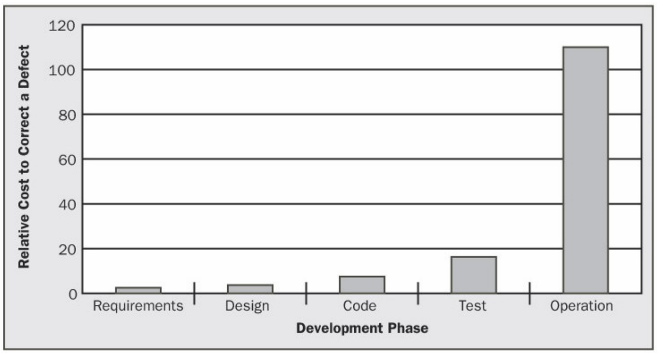


Fig. 1. Cost of correcting a defect

In this article we will try to discuss methods and tools which will enhance the communication among Project Managers, Software Analysts, Customers, Coders and Testers. We will present a mixture of approaches created by requirements analysts in the Software Solution Center and well known industry "best practices". The techniques we cover include a discussion on how to

* More about software requirements – Thorny issues and practical advice by Karl E. Wiegers

seamlessly incorporate them inside the software development process. Changing how people work is much harder than creating a methodology and we have to keep that in mind before implementing anything new. The main targets of this article are to make the software developer aware that continuous improvement of software requirement management will likely provide the greatest benefits to the project, thanks to achieving better customer satisfaction, reducing maintenance and support costs and improving the quality of the solution.

Information included in this article is based on several tens of projects in which we was involved as requirement analysts and a project managers, so both experience and creativeness are the things which have inspired this article.

2. Requirements

As we mentioned earlier there isn't always a clear understanding of what a requirement is. So in this paper we decided to try to define what we believe to be a requirement, before we are going to go further into details. Observers who have different roles in companies may describe the same type of requirement as a software requirement, functional requirement, a system requirement, or a business requirement. This can lead to mistakes. For example, a customer's definition of a requirement may be understood by a coder as a high level abstraction need – this is a very popular kind of miscommunication between interacting sides. Lack of communication regarding definitions may lead to many errors.

IEEE terminology defines a requirement as:

A condition or capability needed by a user to solve a problem to achieve an objective which has to be documented.

Already we can see that this definition encompasses both the stakeholder's and developer's view. There is most of the time an unclear usage of the word "user". So we can come to a conclusion that states that there is no universal definition of a requirement. So the first thing which should be done is to agreeing on a requirement definition with all the fractions working on the specification of any system. We should never assume on a project that the customer/developer knows what we understand by a requirement. In this chapter we will show how we could define requirements.

2.1 Structure of Requirements

In this section we provide a way of categorizing requirements which would be proposed to a client at the beginning of the specification of requirements phase. This categorization is based on the Unified Process (UP) and suggests that requirements fall into categories each with a different scope and purpose:

- Customer Needs
- Features
- Functional Requirements
- Non-Functional Requirements

The above tend to create an organizing schema which can help in the domain of requirements engineering. In our career we have seen many approaches, but none of them fitted as well as the one shown on the picture below.

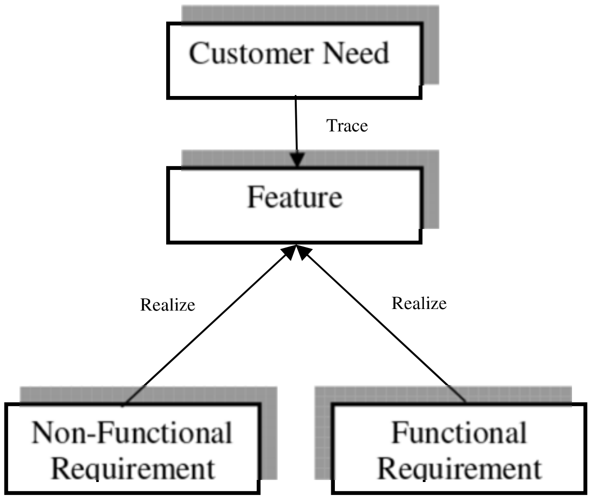


Fig. 2. Relations between certain types of requirements, based on UP

This type of requirements diversity was brought up by us in SSC (Software Solution Center) to properly identify and gather information from different kinds of "users". We wouldn't like to define universal definition of requirements because as far as we are concerned there isn't a one. There are types of requirements which we can define but it is important to acknowledge the division shown on Fig. 2. and to know how to explain certain specializations of requirements.

Underneath there are definitions which we tried to make as precise as it was possible. Understanding those is the fundamental thing on which requirement engineering is based:

Customer need – the Unified Process defines it as a business or functional problem that must be fulfilled in order to justify purchase or use of the system. To really make it easier we would say that this is the reason why the organization is undertaking the project. Customer needs state some benefits that the developing organization or its customers expect to receive from the product, whatever it is going to be.

Feature – in UP a feature is defined as an externally observable service provided by the system which directly fulfills a *customer need.* In other words feature describes what the user will be able to do within the product. It is a representation of goals and task which user can perform. From features we can derive the functional requirements.

Functional Requirement – These types of requirements are used for describing what the developer is obliged to create. Sometimes functional requirements may be called behavioral requirements, these are the traditional statements which define what is the functionality of the system. It is important to acknowledge that those requirements are captured by the use cases. We are going to talk about this subject further on. UP defines them as specific actions that a system must be able to perform, without taking physical constraints into consideration. These are often best described in a Use-Case Model and in use cases. Functional requirements thus specify the input and output behavior of a system.

Non-functional Requirement – Those are understood as any requirements which are architecturally significant, whether this significance is implicit or explicit. The idea of non-functional requirements, presented below, was created by Robert Grady and named FURPS+. Non-functional requirements are ones which address issues such as those described below:

Usability requirements may include such subcategories as:

- human factors
- aesthetics
- consistency in the user interface
- online and context-sensitive help
- wizards and agents
- user documentation
- training materials

***R**eliability requirements to be considered are:*

- frequency and severity of failure
- recoverability
- predictability

- accuracy
- mean time between failure (MTBF)

***P**erformance requirement imposes conditions on functional requirements. For example, for a given action, it may specify performance parameters for the following:*

- speed
- efficiency
- availability
- accuracy
- throughput
- response time
- recovery time

***S**upportability requirements may include:*

- testability
- extensibility
- adaptability
- maintainability
- compatibility
- configurability
- serviceability

3. Requirements management process according to Unified Processes as tailored by SSC

After reading about the basic concepts of requirements now we can easily go further on to get in common with the requirements management.

When talking about requirements management we have to remember about agreeing with the customer about the requirements. It should always be done in a way preferred by both sides. Customer acceptance isn't the one which is the only one – in case of making good projects we have to remember that developers also must agree up on the way of communicating. We, as experienced project managers, are going to point out few things which have to be included in case of requirements management:

- Every project even the smallest one has requirements which should be written down.

Story: *In my career I encountered a simple project in which a team was going to prepare an add-in for documentation. They didn't spend time to manage the requirements so they started to code from the beginning. The effect was that there was no button for confirmation like the OK button or close. This was a stupid mistake shown to the client because the programmers forgot such an obvious thing which never seemed to occur from the code level of abstraction.*

- We should always keep a versioning of requirements and a naming convention, which should be easy by the way
- Requirements analytics should always try to visualize (by Enterprise Architect) as much as they can the requirements to see more details about them
- There is also a rule which tells that Project Manager or a Project Leader should plan his projects according to current requirements
- There should be a traceability of all kinds of requirements to the elements which should trace or realize them
- Requirements Analytics and people responsible for the project should also keep track of the status of requirements

3.1. Role responsibilities

In this subchapter we wanted to mention the roles which are responsible for creating Software Requirements Specification (SRS)[*] and are managing the requirements. We will try to show that on the graph to make it clearer.

[*] A document in which all the requirements should be specified, It should create a way of communicating between the client and the developers.

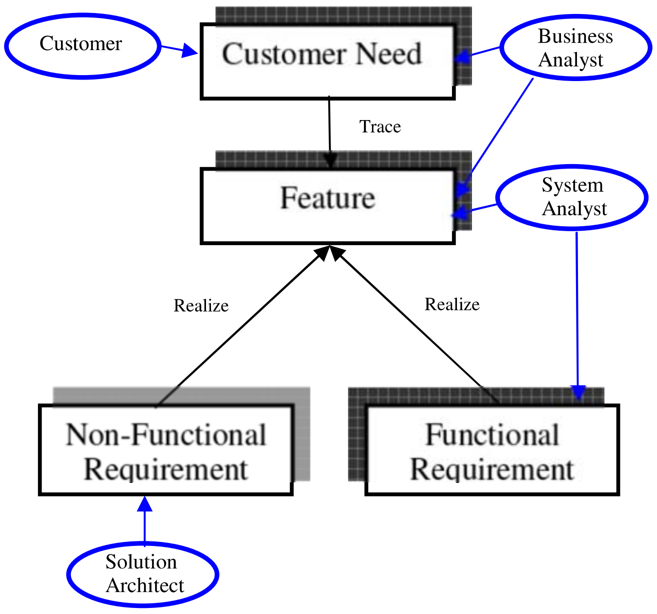


Fig. 3. Relations between certain types of requirements

The upper picture shows in blue circles what roles are assigned to what kind of activities in a very simple way. It is important to understand that those people have to have different skills. It is important to understand that this divergence is not an accidental one. We would like to specify precisely what kind of assignments are going to be affiliated to certain roles.

Table 1. Roles and their tasks in requirements structure model

#	Role	Tasks
1	Customer	• Represent an interest group whose needs must be satisfied by the project
2	Business analyst	• Establish customer needs by communicating with the customer • Creating features which have to be performed by the product. Additionally he will consult in this case System Analyst
3	System analyst	• Lead and create use-case modeling, by creating use cases with scenarios and defining actors, boundaries etc. • Helps to create the features of the product thanks to the knowledge about the functionality • Consults Solution Architect in case of the non-functional requirements
4	Solution Architect	• Specifying and maintaining non-functional requirements with the help of the system analyst. Main target of this is to help initial designing early in the project

3.2 The Case Studies

In this section we would like to present concrete best practices which refer to requirements management in Software Solution Center. But before that, we are going to demonstrate an example case study concerning the requirements management process. After a short introduction to best practices in case of requirement management we are going to show you a certain example of how to implement this using a specific tool called Enterprise Architect.

3.2.1 The Beginning

In this subchapter we are going to point out the main aspects which have to be fulfilled when trying to implement requirements management. Those statements are based on experience of many Software Solution Center software engineers who worked on many different software projects.

First of all we are going to mention all techniques used in SSC for acquainting requirements:

- Surveys and user questionnaires

- Reports and enhancement requests from the customer or consultants
- Interviews with customers
- SRS – Software Requirements Specification
- User observations while working

There is a need to gather requirements from multiple perspectives and sources to enhance communication in case of requirements engineering, especially management.

We are fans of software development methodologies, but thanks to few years of experience in creating software we prefer to use best practices if we want to be sure about achieving the aim. The best-practice approach gives you the opportunity to use generally accepted techniques you can apply to variety of problems. But who decides, what's the best practice? This is a good question to which an immediate answer is that there are experts who look for practices which were effective in successful projects and which aren't performed on projects which failed. That kind of behavior is called **best practices**. So in our case we can suggest those:

- Define change-control process
- Versioning of requirements
- Maintain change history
- Track requirements status
- Display requirements status
- Use a requirements management tool
- Create requirements traceability matrix

So in the next chapters we are going to try to present how we managed to do that in SSC.

3.2.2 Tools

We are not going to dwell on this subchapter because this is not the subject on which we wanted to concentrate in this article. Mainly in case of requirements management we are using in SSC Enterprise Architect, additionally we can mention SubVersion (SVN) for version Control.

Enterprise Architect - Enterprise Architect is a comprehensive UML-based analysis and design tool that aids in software development from the requirements gathering stages through to analysis, design models, testing and maintenance, so it cover the whole requirements engineering. It is a multi-user, Windows-based graphical tool designed to help you build robust and maintainable software. Features include the generation of high-quality customizable documentation, and extensive forward and reverse code engineering. Additionally it is important to notice that we are extending EA for our needs. That causes that we are able of getting the best from the program.

Subversion is an open source application for revision control. Also commonly referred to as **svn** or **SVN**, Subversion is designed specifically to be a modern replacement for CVS and shares a number of the same key developers. This tool is used for marinating the history in case of requirements management.

3.2.3 Implementation

We would like to use in this section an example which will be based on the RAS project lately made by Software Solution Center for our partner. Our purpose is not to show the whole project but just to show some parts which will act as good examples. In this case study we are going to map certain elements form the block diagram mentioned earlier to EA just like in the picture below:

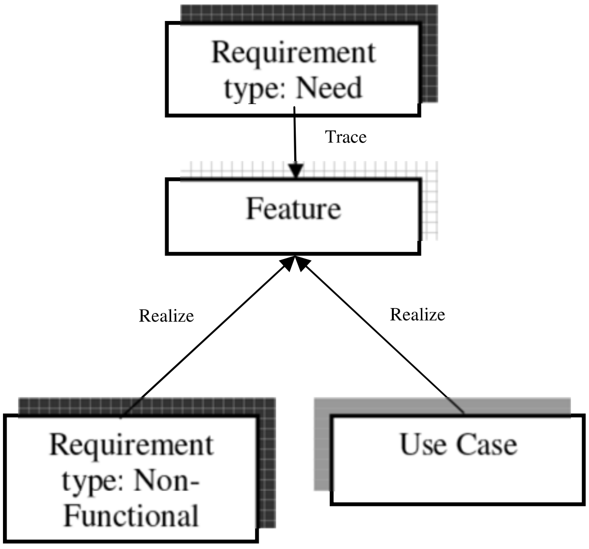


Fig. 4. Relations between certain types of requirements in EA

The problem which we are going to work on:

The process starts with analyzing the customer problem. This is done to recognize problems, customer needs. After proper analysis business analyst is able to define customer needs after performing multiple ways of gathering

information mentioned in subchapter 3.2.1. Requirements may come from many sources. People who will provide customer needs for us should be called *Customers*.
The result of that work is defining the:

1 Purpose
The purpose of this project is to build the system which will provide an easy and fast way of installing the Windchill environment. One of the main applications will be the Windchill configurator which will configure a standalone or VMware instance of Windchill. This configurator will choose the proper version of the software which can be deployed to create an installation package. The project also specifies maintenance of VMware images and other reusable artifacts as well as ways of searching and retrieving them with a suitable guidance and process description. Additionally verification of the configuration must be done with a possibility of restoring the system from a backup copy.

2 Scope
The proper configuration of the system can be cumbersome because of components version compatibility. Implementing a system which automates the process of choosing the proper system components is a great advantage. Moreover, providing reusable assets library can seriously enhance the process of customization of Windchill for an end-user.

3 Objectives
1) Windchill Instantiator which will be used for configuring a desired instance of Windchill.
2) VMware images of Windchill configurations.
3) LDAP structure modifier module.
4) Asset Manager which is a tool for managing the assets in context of placing them in and/or retrieving from the library.
5) Verifier which is a tool for verifying the created instance of VMware disk**.**
6) User manuals and other documentation for the project deliverables.

And creation of CUSTOMER NEEDS in EA:

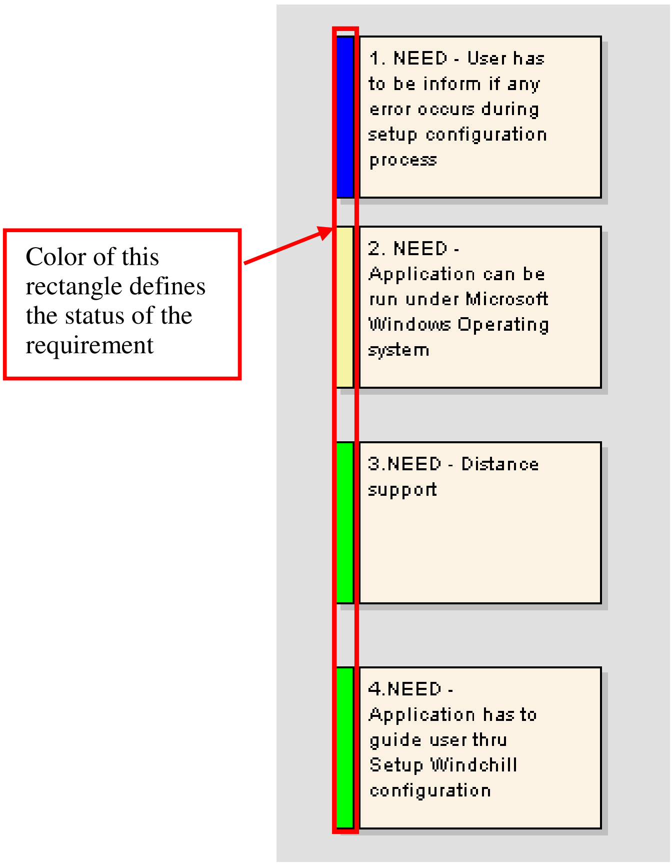

Color of this rectangle defines the status of the requirement
1. NEED - User has to be inform if any error occurs during setup configuration process
2. NEED - Application can be run under Microsoft Windows Operating system
3.NEED - Distance support
4.NEED - Application has to guide user thru Setup Windchill configuration

Fig. 5. Customer Needs

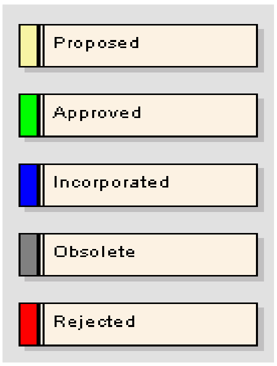


Fig. 6. Colors of the status type

The prefixes are generated automatically thanks to the options of Enterprise Architect, which gives us an opportunity to provide proper naming convention.

2) After defining the customer needs the System Analyst can derive form them FEAUTURES with the help of earlier involved Business Analyst.
Those are created precisely as in the procedures and look like that in Enterprise Architect:

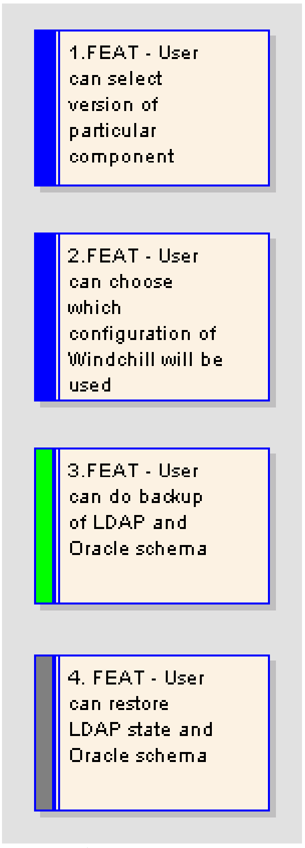

1.FEAT - User can select version of particular component
2.FEAT - User can choose which configuration of Windchill will be used
3.FEAT - User can do backup of LDAP and Oracle schema
4. FEAT - User can restore LDAP state and Oracle schema

Fig. 7. Features

3) In this step System Analyst and Solution Architect create the Use Cases (capturing FUNCTIONAL REQUIREMENTS) and NON-FUNCTIONAL REQUIREMENTS of the system, which can be displayed in EA as shown below:

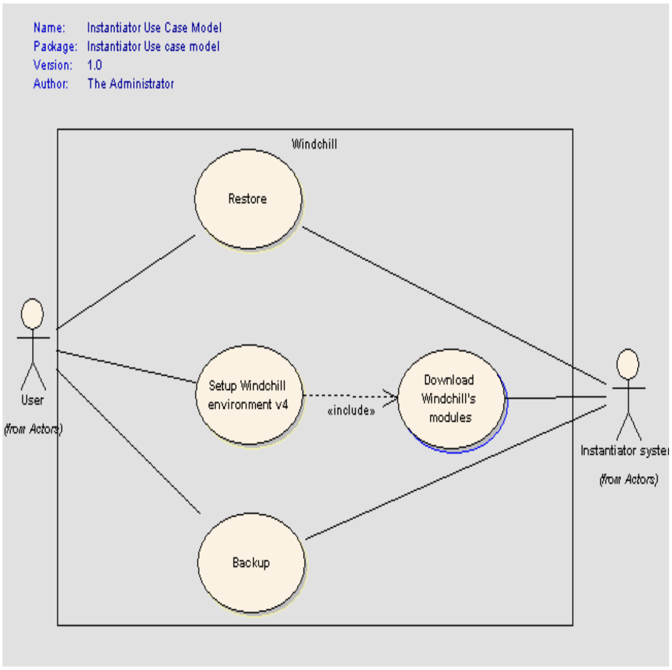

Name: Instantiator Use Case Model
Package: Instantiator Use case model
Version: 1.0
Author: The Administrator
Windchill
Restore
Setup Windchill environment v4
«include»
Download Windchill's modules
Backup
User
(from Actors)
Instantiator syste
(from Actors)

Fig. 8. Use case model

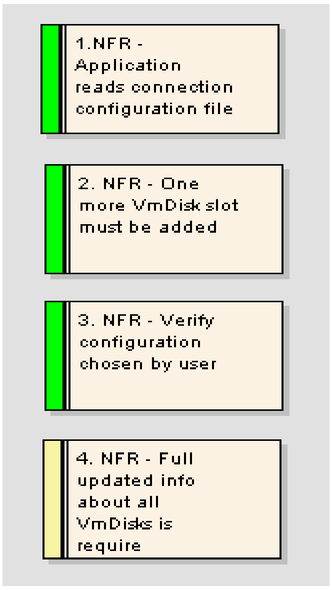
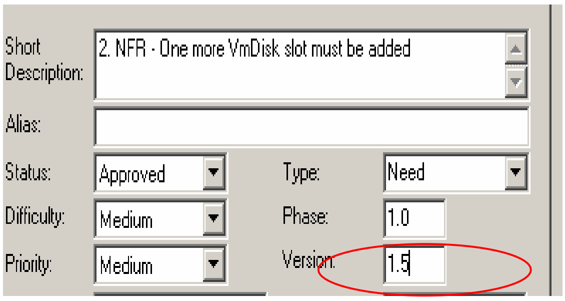

1.NFR - Application reads connection configuration file
2. NFR - One more VmDisk slot must be added
3. NFR - Verify configuration chosen by user
4. NFR - Full updated info about all VmDisks is require

Fig. 9. Non Functional Requirements

Additionally we would like to add what is guaranteed from the best practices mentioned earlier on in our case study:

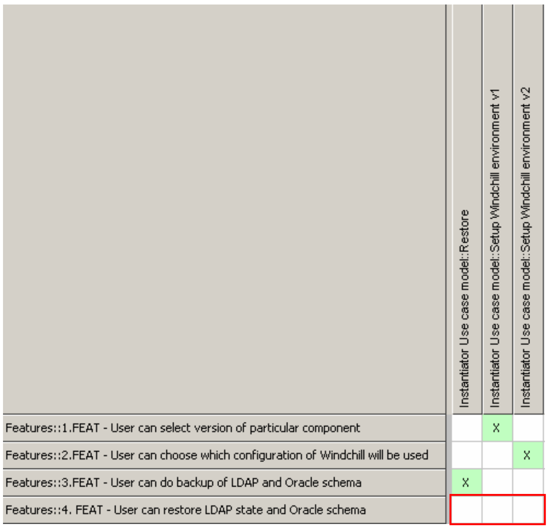
- **History tracking** is provided by the svn version control which can be attached to a project made in EA.
- **Change control and versioning** – this is assured thanks to the options affiliated in the properties (version) of elements which interest us and by the names of packages, which contains the view of the version:



Short Description:	2. NFR - One more VmDisk slot must be added		
Alias:			
Status:	Approved	Type:	Need
Difficulty:	Medium	Phase:	1.0
Priority:	Medium	Version:	1.5

Fig. 10. Version of the element

- **Traceability Matrix** – the purpose of using this functionality is to avoid not realizing certain requirements, by not associating them with proper elements. We should use the traceability matrix to make sure that we realized all the required elements.



	Instantiator Use case model::Restore	Instantiator Use case model::Setup Windchill environment v1	Instantiator Use case model::Setup Windchill environment v2
Features::1.FEAT - User can select version of particular component		x	
Features::2.FEAT - User can choose which configuration of Windchill will be used			x
Features::3.FEAT - User can do backup of LDAP and Oracle schema	x		
Features::4. FEAT - User can restore LDAP state and Oracle schema			

Fig. 11. Traceability matrix

4. Creating Documentation

The creation of documentation is provided thanks to the idea to customize EA in such a way that it would generate documentation that we would want. In case of the process of requirements management we must create a document called the Software Requirements Specification. It can be described as:

The Software Requirements Specification (SRS) is a document which focuses on the collection and organization of all requirements surrounding your project. It is useful for collecting your project software requirements in a formal, IEEE-830-1998 -style document. The software requirements specification is sometimes called a *functional specification,* a *product specification*, a *requirements document*, or a *system specification*, although organizations do not use these terms in the same way. The SRS precisely states the functions and abilities that a software system must perform and respect. SRS should describe as completely as necessary the system.

In SSC our Program Managers have created a SRS document, which contains the following:

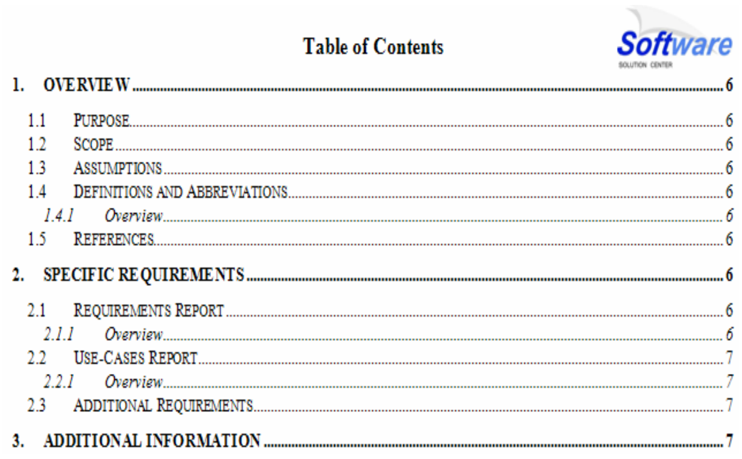


Table of Contents

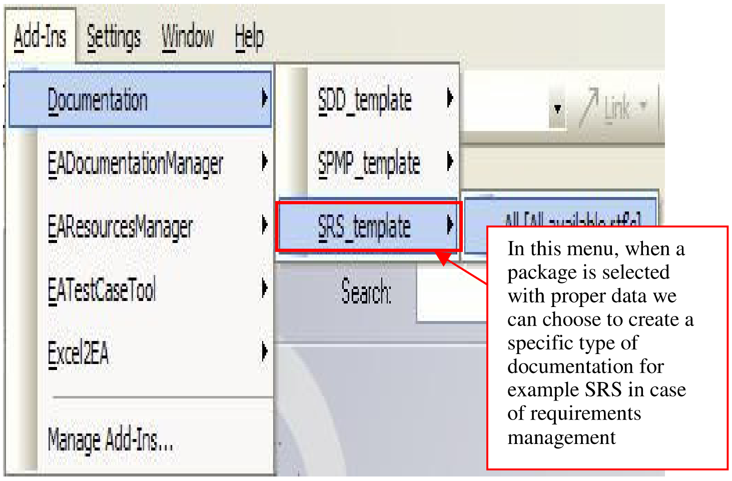
Fig. 12. SRS table of contents

This document was made in a certain way to fulfill all higher mentioned conditions and may be treated as best practice.

SRS document is an example of how you can provide required data for:

- Interaction with the client
- Project managers base their estimates about the product on this document in later documentation like SPMP[†]
- Tells the software development team what to build.
- The testing group uses the SRS to develop test plans, cases, and procedures.

We achieve proper documentation generation thanks to the possibility of using an EA general add-in. In SSC humble opinion, there are no tools which will precisely fit requirements management. That made Software Solution Center tailor EA especially for our requirements. The tool responsible for creation of the documentation is called EADocumentationGenerator. It facilitates creation of project documentation. With a single mouse click you can generate various types of documents (Master Documents), that combine different types of Enterprise Architect generated RTF files together. Each package has a set of Master Documents that can be generated for him. Master Document can have set of versions that define which RTF files should be included in the output. Defining Master Documents its versions and assigning packages can be made through EADocumentationManager. Below you can see some functional User Interfaces of the program:

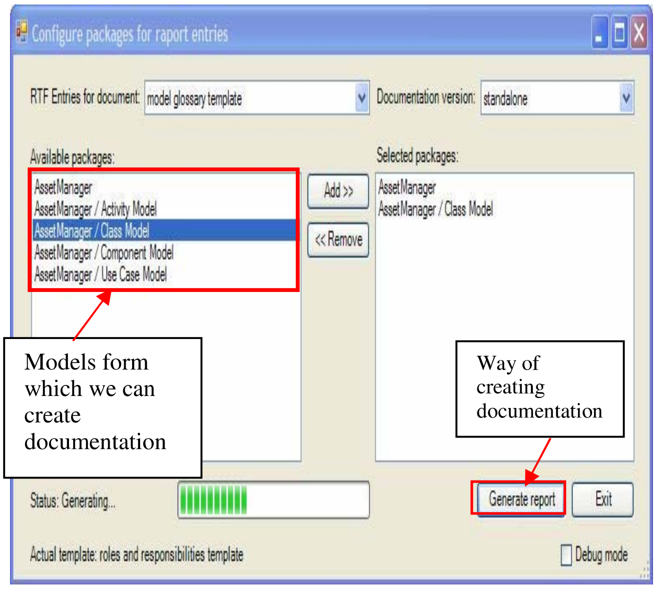


In this menu, when a package is selected with proper data we can choose to create a specific type of documentation for example SRS in case of requirements management

Fig. 13. Creating SRS documentation from EA add-in

[*]SPMP – Software Project Management Plan – a type of documentation

Picking the type of documentation to create from an EA add-in:



Configure packages for raport entries

RTF Entries for document: model glossary template

Documentation version: standalone

Available packages:
AssetManager
AssetManager / Activity Model
AssetManager / Class Model
AssetManager / Component Model
AssetManager / Use Case Model

Selected packages:
AssetManager
AssetManager / Class Model

Status: Generating...

Actual template: roles and responsibilities template

Add >>
<< Remove
Generate report
Exit
Debug mode
Models form which we can create documentation
Way of creating documentation

Fig. 14. EA add-in UI created by SSC

CONCLUSIONS

Proper management of requirements is important in every stage of project development. It is even more important at the transition phase when there is the need to assure customer that we have fulfilled his/hers needs. It will be also crucial in the future maintenance of the system when we need to validate new requirement against the existing ones. By implementing requirements management SSC gained a lot of advantages in every manner of business. Business requirements can do a lot of good for projects. After the lecture of the article we can say that those benefits include:

- Serving as the foundation for product design and implementing. Therefore, well-understood and well-communicated requirements help developers devise the most appropriate solution to the problem
- Enhancing decisions which projects to finance
- Making testing possibilities better thanks to better referencing
- Well done requirement specification allows the team to prioritize its remaining work. Most projects need to make compromises to ensure that they implement the most critical and most timely functionality.
- Helping tracking of the project completeness

- Effective user involvement in establishing the requirements reduces the chance that users will reject the new system upon delivery

Additionally requirements management makes process in the company faster and more productive which increases the ROI (Return on Investment). It is important to understand the whole process to improve work of analytics, designers, programmers and interactions with the client. Best practices and methodologies mentioned through the paper should present how a good process of requirement management should be implemented and understood.

References

[1] Wiegers Karl E. :Thorny issues and practical advice, 2005
[2] Clouse Aron, Turner Richard, Ahern Denis:. CMMI Product Team: CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing, Version 1.1 Carnegie Mellon, Software Engineering Institute, Pittsburgh 2002.
[3] Enterprise Architect User Guide, Version 6.5 Geoffrey Sparks SparxSystem 2006
[4] Unified Modeling Language: Superstructure version 2.0, Object Management Group, 2006
[5] W. Fitzpatrick Brian, C. Pilato Michael, Ben Collins-Sussman: Version Control with Subversion For Subversion 1.3,
[6] Bergström Stefan, Råberg Lotta: Adopting the Rational Unified Process: Success with the RUP, 2003
[7] Eeles Peter: Capturing Architectural Requirements, 2001
[8] Wikipedia