Requirements Document Template

for Software Development Projects

Version 1.0 – 28th January 2008

by Adriano Comai – www.analisi-disegno.com

Foreword

Goal of this template

The goal of this template is to provide useful suggestions for the documentation of software requirements in a development project.

Requirements of a software system may be documented, or not. There may be many reasons not to document software requirements, but we don't need to discuss it here. This template is for those who need to document requirements.

Software requirements may be documented in different ways. There is not a single universal standard for documenting requirements.
The documentation of requirements is usually more or less formalized depending on the criticality of the product to be developed.
If your product is life critical, that is if a software malfunction may harm people, you need a very thorough and formalized management of software requirements. In this case, don't use this template. This template is intended for use in non-life-critical software projects.

If you already use or need to use another - internal or industry-specific - existing type of document or template, I hope you will anyway find in this template some useful suggestion. For people involved in an already started project, or maintaining an existing software system, the template may also be used as a checklist for missing aspects.

The template does not contain any specific treatment of contractual aspects (rules governing the client-supplier relationship) and validations (acceptance by the client and other stakeholders of a specification), because these characteristics depend on the specific organizational contexts, and can't be treated in general terms.

Structure of the template

This template is a composition of various sections. Each section can be a separate document, or remain just a section of the single composite document. Each section may be more or less formalized, depending on your needs and preferences. What's more, each section may be used or not used, again depending on your needs and preferences.

The main sections are:

- Vision
- Release Plan
- Software requirements specification
- Glossary - Data Repository

The **Vision** is a high level system specification. In its simplest form, it may be a two-pages product data sheet. In its most complex form, it may be a 20 pages executive-level system description.
It is the main communication medium among stakeholders about the work to be done. It contains no details.

The **Release plan** specifies the releases planned for the system, and other high-level milestones if significant for the client.
Whenever possible, it is a good strategy to build and ship the system incrementally, according with client's priorities and system's architecture. The release plan can and must be changed, whenever client and developers need to change it.

Every project, even the smallest and most informal one, gets benefits from having a vision and a release plan.

Every project, but the most informal ones, needs some form of **Software requirements specification**, in order to document the shared understandings between client and developers. Depending on needs and preferences, the software requirements specification may be more or less formal, and it may contain every detail or just the most significant points of agreement.

The most common elements that constitute a software requirements specification are:

1. A set of user stories, or use cases. User stories are small chunks of system functionality; use cases are more complete operational scenarios of system usage. User stories are short, and may correspond to a single atomic functional requirement (see below) ; use cases may be longer because they describe end-to-end usage scenarios.
2. A list of atomic requirements (“the system shall...”). Some of them specify a system functionality, and are called functional requirements. Others specify other types of characteristics for the system (like usability, performance, reliability level, internationalization) and are called non-functional requirements, or quality attribute requirements.

The form of the specification may vary from the most informal (index cards for user stories to post on a wall) to the formal (several documents with hundreds or thousands of pages).

The **Glossary - Data Dictionary** section deals with informations and data. It may be specified at different complexity levels, depending on actual needs.

A Glossary is an often informal list of terms and definitions. It defines a common language, the minimal basis for mutual understanding in a project.
A Data Dictionary is a structured repository of data definitions, which specifies structures and relationships of data.

Who, when and how (to use the template)

In the tradition of software development, the role which has the responsibility to document requirements is called “analyst”. In actual organizational contexts and situations, this role may be fulfilled by people with different labels.

The timing of the specification depends on the software development process used. Old “waterfall” methods require the specification of all detailed requirements upfront, and a global client validation (client signature) before development begins.

Iterative and agile methods suggest a just-in-time approach to requirements management. Partial sets of requirements (say, those concerning a specific system functionality) are detailed just before their implementation, in order to make as short as possible the cycle of definition-build-test, and to reduce the risks and the impacts of misunderstandings and reworks.
In these contemporary approaches, the software requirements specification may be created and validated incrementally. It is a growing mosaic, not a monolith.

But all requirements are not equal. While just-in-time requirements definition works effectively for functional requirements, it may be dangerous for non functional requirements, because a late discovery of a non functional requirement (say, related to internationalization, or portability, or security) may cause extensive rework. When it is possible, it is far better to clarify non functional requirements at the beginning of a project.

System <_____>

Requirements

Version History

Version n.	Date	Author	Changes

Index

Purpose and audience for this document

Purpose

State the purpose of this document (e.g. "Document the software requirements for Project YYY"). When useful, describe briefly what's included in the document.

Audience

State the intended audience for this document (e.g. "This document is targeted to anyone involved in the project, and in particular to the following roles...").

Vision

The **Vision** is a high level system specification. In its simplest form, it may be a two-pages product data sheet. In its most complex form, it may be a 20 pages executive-level system description.
It is the main communication medium about the work to be done. It contains no details.

System background

Business Context, Opportunities and Risks

Explain the business reasons and the origin for this system.

Stakeholders Profiles and Needs

List the stakeholders for this system. The scheme of the following example may be used:

Stakeholder	Why is involved	Interests and Needs
Marketing	Project client	Competitive advantage Fast project completion Cost control
Seller	System user	Speed and usability Estimate simulation features
Legal department	Legal implications	Compliance with international laws

Main Characteristics of the System

Goals of the System

Describe briefly the goal of the system from the point of view of the client.

System Context: relationships with users and external systems

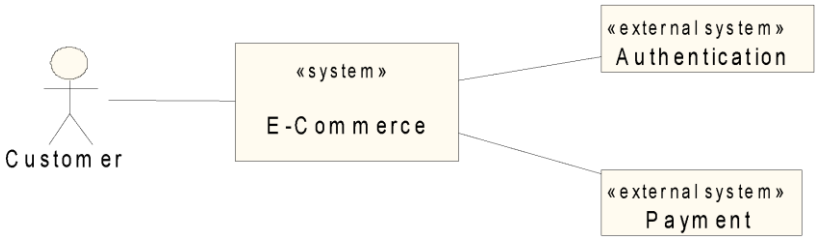
Describe synthetically the system context, that is the relationships of this system with users and external systems.
The system context draws the boundary that distinguishes what's in the system and what's outside of it. The context also represents the relationships between the system and the external entities with which the system interacts.

For complex systems there may be up to 3 different levels of context, specifying the boundaries for the system at different levels:

0. Socio-technical level.
   At this level, a system may be made of human, hardware and software components. It is a business level, which includes both the aspects (activities, communications) to be automated, and those that will not be automated. It is a very useful analysis level, because it may help to discover opportunities and risks of automation. The socio-technical level may be omitted when the projects is a minor evolution of an existing system, or if the boundaries of the physical and software levels are already defined.
2. Physical level.
   At this level, a system may be made of hardware and software components, but not of human components. Every human activity is outside the system. It is a useful level for systems in which the hardware components must be built, because they are not standard (e.g. medical apparels, car alarm systems). It is not useful to describe it with a separate context level for software systems which will run on standard hardware.
3. Software level.
   At this level, a system may be made only of software components . Everything that is not software is outside the system. Often, this is the only level that needs to be described.

If there are a lot of different user roles and external systems, simplify with abstractions. The detailed requirements for the relationships of the system with the external world are better expressed in the Requirements Specification than in the Vision.

The system context may be expressed in graphical form, with a UML (Unified Modeling Language) diagram simple as the following one:



Major Features

List the major features of the product, that is the most relevant functions and characteristics (operating environment, level of service, usability, security ecc.).

Dependencies and Critical Success Factors

Describe which factors may influence the quality and the success of the product, such as the availability of an external component in a certain date.

Release Plan

The release plan documents, at a high level, the release milestones shared between client and developers. It specifies the progression of releases and their content. Whenever the agreements between client and developers should change, the release plan must be changed accordingly.

The intent of the release plan is to clarify our forecasts about the incremental delivery of system functions, because in most projects not every requirement are satisfied at the same time. The release plan is a road map, not a detailed work plan.

The level of detail of the release plan must be coherent with the vision and the requirements specification. For example, if the Requirements specification is organized around use cases, it is useful that the release plan references them.

Example

Feature or use case	1st release (date)	2nd release (date)	3rd release (date)
UC1	x (partial, only base scenario)	*	
UC2			*
UC3		* (partial, without alternate scenarios 2 and 3)	*
UC4	x		
UC5		* (partial)	*
UC6		*	
UC7	x		

Requirements Specification

The requirements specification contains details. It has two main Parts: Use Cases AND Atomic Requirements.

Use Cases / User Stories

This section is organized around user functionalities. I suggest the use of an existing template, such as the one provided by Alistair Cockburn - http://alistair.cockburn.us/index.php/Basic_use_case_template, or a similar one.

Atomic Requirements

The following sections contain a list of atomic requirements, organized upon a classification scheme such as the one used in the guide Requirements-By-Example, http://www.analisi-disegno.com/requisiti/Requirements-By-Example.htm, used below, or a similar one.

Functionality

Detailed Functional Requirement

Data and Accuracy Requirements

Interoperability

Responsibility

Operativeness

Availability

Performance

Capacity

Scalability

Reliability

Installation

Portability

Compliance

Laws and Regulations

External and Internal Standards

Audit

Business Rules

Technologies

Cultural and Political Requirements

Usability

Physical Environment

Appearance and Style

Ease of Use

Personalization

Internationalization

Learning Time

Accessibility

Safety and Security

Safety

Access Protection

Integrity

Privacy

Project Time Requirements

Project Budget Requirements

Documentation, Maintenance and Support

Documentation

Maintenance

Support

Training

Glossary - Data Dictionary

The **Glossary - Data Dictionary** section deals with informations and data. It may be specified at different complexity levels, depending on actual needs.

A Glossary is an often informal list of terms and definitions. It defines a common language, the minimal basis for mutual understanding in a project. It is highly recommended.
A Data Dictionary is a structured repository of data definitions, which specifies structures and relationships of data.

A Data Dictionary is extremely useful as a complement to software requirements in complex projects.

For the requirements documentation of simpler projects, a Glossary may be sufficient. A good example of Glossary is the Human Genome Glossary:
http://www.ornl.gov/sci/techresources/Human_Genome/glossary/