Requirements

CxOne Standard

CxStand_Requirements.doc

November 3, 2002

Construx
SOFTWARE

cxone

Advancing the Art and
Science of Commercial
Software Engineering

CONTENTS

Copyright Notice

1 Introduction

This standard defines key organizational structure and terminology relating to software requirements engineering. There is considerable industry literature available that organizes requirements knowledge and deliverables, but there are no universally accepted taxonomies. This document defines a standard CxOne taxonomy that will provide a common playing field to discuss, organize, and use requirements concepts.

1.1 Overview

CxOne defines this knowledge area as the elicitation, analysis, and specification of requirements.

Requirements development consist of identifying sources of requirements (elicitation); ensuring the requirements correctly and completely describe the desired work end product (analysis); and documenting the requirements (specification).

The requirement phase can, and often needs to be, a highly iterative process. This taxonomy does not imply a workflow, rather provides a toolbox from which approach materials can be drawn.

1.2 Goals

CxOne support for software requirements development focuses on the following goals:

- Defining a common terminology.
- Distillation of common principles into readily available checklist items.
- Provide a resource for information about existing best practices, methodologies, and techniques.
- Increase productivity by addressing issues in a uniform fashion, reducing common errors, duplication of effort, and multiple solutions for the same problem.

1.3 Background

CxOne requirements materials are synthesized from many sources including the SWEBOK, IEEE software standards, Steve McConnell's works, other industry sources and literature, and Construx's experience with software projects, consulting, and training.

1.3.1 Types of Requirements

CxOne breaks requirements down into three types. There are many techniques used to gather and capture requirements. Although some techniques work particularly well for different types, the type of requirement does not constrain the technique used to capture or gather it. Rather the type is a way to organize the intent of the requirement.

Business Requirements

Business requirements can be considered **why requirements**. These requirements capture the high level objectives of the organization or customer requesting the system or product. They can be considered the "why are we doing this" requirements and should capture the fundamental reason for the projects existence.

Functional Requirements

Functional requirements can be considered **what requirements**. These requirements capture the functionality that must be built into the system to satisfy the business requirements.

Non-Functional Requirements

Non-functional requirements can be considered **how or how well requirements**. These requirements capture the technology specific and -ility requirements of the system. The -ilities include items like compatibility, usability, performance, reliability, etc.

1.4 Relationship to other CKAs

While each CKA interacts to some level with all other CKAs, Requirements interacts to the highest degree with Design, QA, Testing, Engineering Management, and CM.

1.4.1 Requirements vs. Design

It can often be difficult to determine when requirements gathering stops and system design begins. As all projects are unique the final decisions on this must be left to the project team, however some general guidelines are:

- Requirements focus on determining "what" the solution is.
- Design focuses on determining "how" the solution will be accomplished.

Some requirements will be of the "how" type. Either for a binding authority or an overriding business need, a particular solution will be "required". For example, a contract may state that the code be in C++. The contract here represents a binding authority. For a more discussion of the spectrum of requirements, design, and construction see *CxStand_Design*.

1.4.2 QA of Requirements

The identification and resolution of mistakes, omissions, and inaccuracies of the requirements is far more cost and time efficient at this stage than later in the project lifecycle. Peer reviews can be used to review textual and model based specifications. Prototypes, especially of the UI, are effective in clarifying requirements.

1.4.3 Testing of Requirements

The creation of test cases for the requirements is an excellent way to assure the correctness of requirements. These test cases should be created prior to the acceptance of the requirements and the creation of designs based on the requirements.

1.4.4 Engineering Management and Requirements

As the definition of the software, requirements are critical components into the scope of a project as well as the estimation practices. Prior to a reasonably complete understanding of the requirements are understood, any estimates will have a high amount of uncertainty. Negotiation of requirements to be “in” or “out” is a way to achieve parity between the desired targets of a project and the project estimates.

1.4.5 CM of Requirements

Requirements are the definition of the software and as such should be placed under explicit change control as described in the Configuration Management CKA.

1.5 Relationship to External Standards

A detailed discussion of how CxOne requirements organization relates to other common standards is provided in the Appendix.

1.5.1 SWEBOK

The CxOne requirements area has made the following adaptations of the SWEBOK organization:

1. *Requirements Engineering Process* is dealt with in the Process CKA.
2. *Requirements Validation* is dealt with in the most relevant CKA. E.g. Reviews are in the Quality CKA, Acceptance Tests are dealt with in the Testing CKA, etc.
3. Portions of *Requirements Management* are dealt with in the Configuration Management CKA and the remainder have been rolled into the *Requirements Specification* section.

These changes were made to support a more pragmatic, usable taxonomy for materials.

1.5.2 IEEE Standards

CxOne materials explicitly and implicitly supports IEEE 1002-1987, but the requirements CKA extends the scope and level of support provided by the IEEE standards.

1.5.3 UML

The Unified Modeling Language (UML) has a heavy influence on CxOne requirements modeling materials. UML is becoming the de facto industry standard for requirements modeling as well (it is already the industry standard for object oriented design).

CxOne does not treat the UML as a monolithic whole; notation and concepts are separated from processes and techniques.

2 Organization

The organization of CxOne Requirements CKA materials is described below.

2.1 Requirements Elicitation

Elicitation identifies source of requirements and then specific requirements through a wide variety of techniques such as interviews, prototypes, facilitated meetings, etc.

2.2 Requirements Analysis

Analysis examines the requirements gathered from elicitation to ensure they correctly and complete describe the application domain. Analysis can include determining items like:

- Requirement type (business, functional, non-functional)
- Absolute and relative priority
- Requirement scope (application wide, component specific, etc.)
- Likelihood of change during the project

Analysis includes negotiation when requirements conflict one another, the available project resources, project constraints, etc.

2.3 Requirements Specification

Specification records the requirements through a natural language documents (SRS), models (UML class diagrams, etc), requirements database, or a combination of techniques.

Specification provides support for or enables:

- Traceability to downstream artifacts such as test cases, design documents, code, etc.
- Requirements attributes such as priority, target release, status, etc.

3 Appendix

3.1 Alternative Organizations

This section lists some alternative ways that requirements are broken down.

3.1.1 SWEBOK

Term	Definition/Description
Product Parameters	Requirements of the system being developed. Further broken down as: • Functional Requirements of the system. (aka capabilities) • Non-Functional Requirements. These act to constrain the solution.
Constraints or Quality Attributes	A constraint on the development of the system

3.1.2 Robertson and Robertson

Term	Definition/Description
Functional	What the product must do: actions to take, information to remember, business rules and policies to enforce
Non-Functional	Properties that the product must have: look and feel, usability, performance, operational environment, maintainability and portability, security, cultural and political, legal. Usually attached to individual functions
Constraints	Global factors that apply to the entire product rather than specific functions: purpose of product, stakeholders and users, etc.

3.1.3 Weigers

Term	Definition/Description
Business	High level objectives of the organization or customer requesting the system or product.
User	Describe the tasks the users must be able to accomplish with the product: what do they want to be able to do.
Functional	Define the software functionality the developers must build into the product to enable users to accomplish their tasks, thereby satisfying the business requirements

3.1.4 Software Engineering Institute

Term	Definition/Description
Functional	Specifies a function that a system or system component (i.e., software) must be capable of performing.
Non-Functional	Includes items such as performance, reliability, interfaces, and design constraints. Generally considered the "how well" requirements. They state characteristics of the system to be achieved that are not related to functionality.
Inverse	Describe the constraints on allowable behavior. In many cases, it is easier to state that certain behavior must never occur than to state requirements guaranteeing acceptable behavior in all circumstances. Software safety and security requirements are frequently stated in this manner.
Design Constraints	Boundary conditions on how the required software is to be constructed and implemented.