

## Quality inside Scia Engineer

**CONTENTS**

<b><u>QUALITY INSIDE SCIA ENGINEER</u></b>	<b><u>1</u></b>
<b><u>1 DOCUMENT PERSPECTIVES</u></b>	<b><u>3</u></b>
<b><u>2 INTRODUCTION</u></b>	<b><u>3</u></b>
<b><u>3 POLICY</u></b>	<b><u>5</u></b>
3.1 QC POLICY	5
3.2 QA POLICY	5
<b><u>4 THE V-MODEL</u></b>	<b><u>6</u></b>
4.1 CUSTOMER REQUIREMENTS / EXPERT SYSTEM REQUIREMENTS	6
4.2 SYSTEM REQUIREMENTS	7
4.3 TECHNICAL DESIGN	7
4.4 COMPONENT DESIGN	7
4.5 UNIT TESTING	7
4.6 INTEGRATION TESTING	7
4.7 SYSTEM TESTING	7
4.8 ACCEPTANCE TESTING	8
<b><u>5 RELATED PROCESSES AND PROCEDURES</u></b>	<b><u>9</u></b>
5.1 SOFTWARE PRODUCT LIFE CYCLE	9
5.2 QUALITY IMPROVEMENT PROCEDURE	9
5.3 QUALITY IMPROVEMENT ACCEPTANCE CRITERIA	11
5.3.1 BUG FIXING ACCEPTANCE CRITERIA	11
5.3.2 TEST ROUND ACCEPTANCE CRITERIA	12
5.3.3 DEFINITION BUG PRIORITIES	12
5.4 AUTOMATED TESTING	13
5.5 QA ACTIVITIES	13
<b><u>6 REFERENCES</u></b>	<b><u>14</u></b>

## 1 Document Perspectives

Subject	QA and QC in Scia Engineer
Purpose	To give an overview of the QA and QC activities (policy, procedures) related to the Scia Engineer package.
Author	CVL
Release	
Document	QA_QC_SE(CVL20100208)
Distribution	
Date	23/09/2008

## 2 Introduction

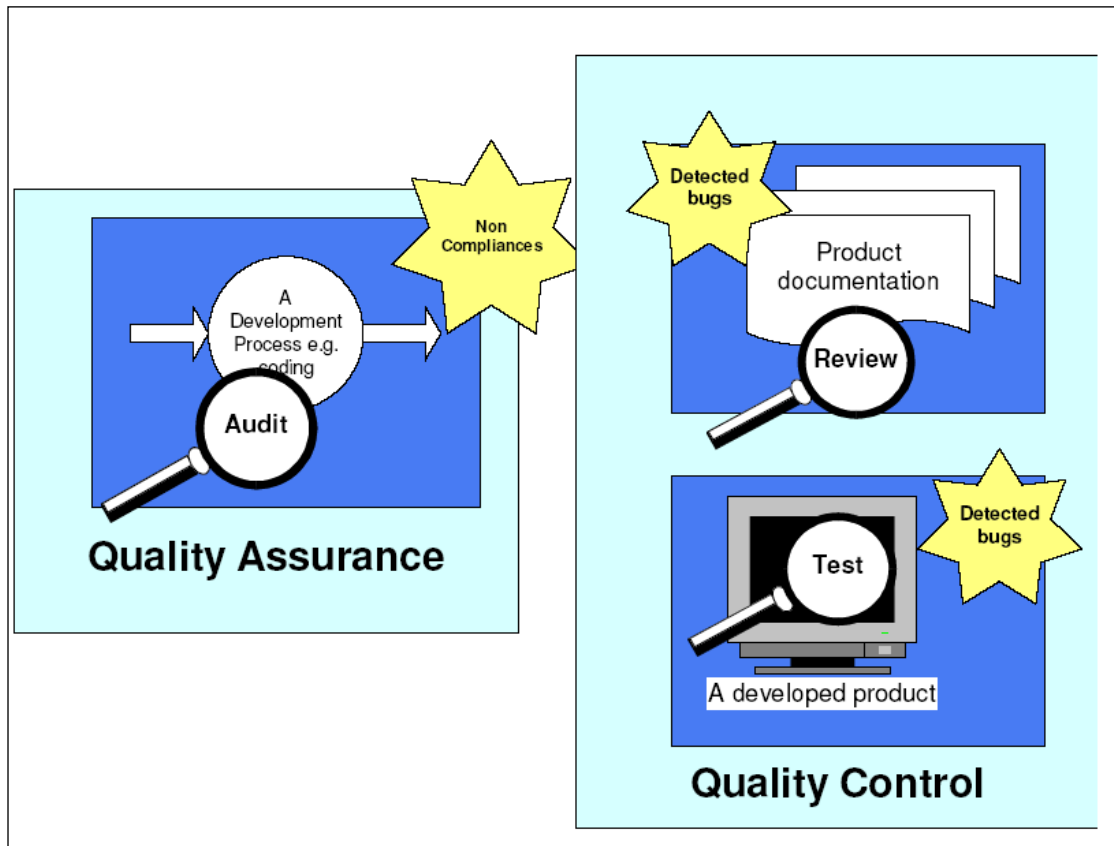
Quality Control (QC) or 'Verification and Validation' (V&V, VER, VAL) is the process of ensuring that software being developed or changed will satisfy functional and other requirements (validation) and each step in the process of building the software yields the right products (verification).

The purpose of Validation (VAL) is to demonstrate that a product fulfils its intended use when placed in its intended environment. The purpose of Verification (VER) is to ensure that selected work products meet their specified requirements. In other words, verification ensures that "you built it right"; whereas, validation ensures that you "you built the right thing". See also Ref.[1], Chapter 'Verification' and 'Validation'.

The two major QC activities are reviews and testing. The review of work products is performed by peers during development of the work products to identify defects for removal. Testing is the operation of the software with real or simulated inputs to demonstrate that a product satisfies its requirements and, if it does not, to identify the specific differences between expected and actual results.

Quality Assurance (QA) is defined as the planned and systematic pattern of all actions necessary to provide adequate confidence that the software has been designed and developed using adequate processes. Quality Assurance evaluates the process and the associated work products. It provides staff and management with objective insights into these processes and work products. See also Ref.[1], Chapter 'Process and Product Quality Assurance'.

The difference between Quality Assurance (QA) and Quality Control (QC) is depicted in the figure below :



In this document, the SCIA policy concerning QC and QA are described (Chapter 3). The used model (Chapter 4) and supporting processes and procedures (Chapter 5) are explained.

## 3 Policy

### 3.1 QC Policy

The SCIA policy for the Quality Control is the following :

"  
*Software products ( standard and customized ) are inspected and tested according to documented procedures and benchmarks to assure compliance of the software to the functional specifications.*  
"

### 3.2 QA Policy

The SCIA policy for the Quality Assurance is the following :

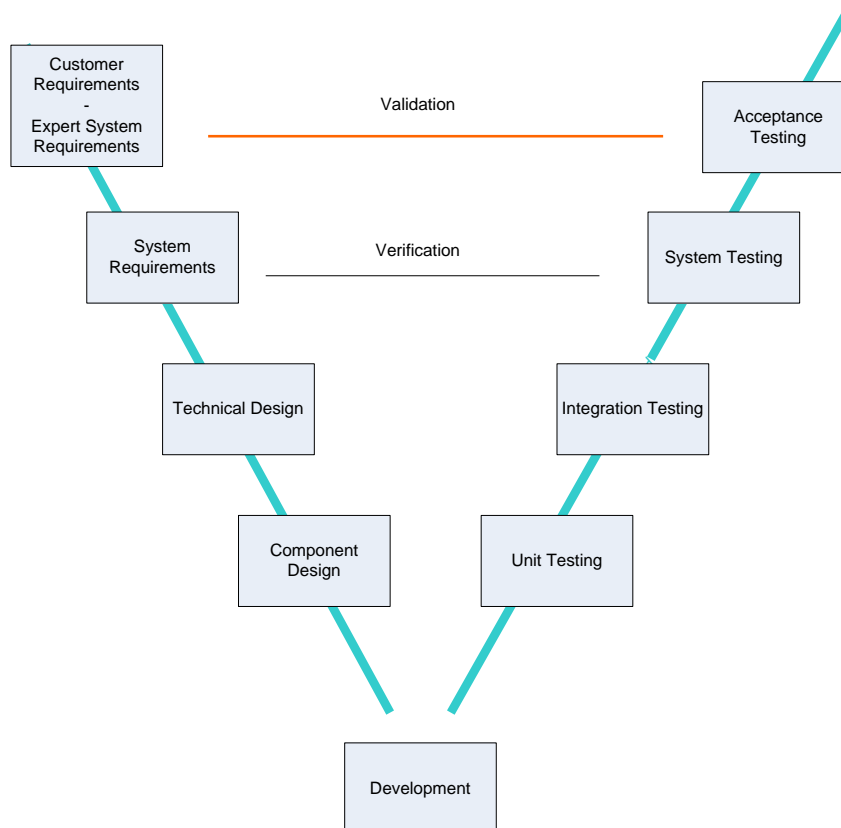
"  
*SCIA has competent employees in each division: business units ( product engineering & development and projects ), sales & marketing, support & training, administration ( finances, delivery, purchase, general administration ), SDS/IT ( System Development Services for internal hardware/software ) and management. Through procedures, documentation, training, audits the employees have a solid awareness to control the major business processes: development of software, delivery, support, training, sales, administration, etc.*  
"

## 4 The V-Model

The applied model for achieving the QC policy is the so-called V-Model. The V-Model, also called the Vee-Model, is a product-development process. The V-Model gets its name from the fact that the process is often mapped out as a flowchart that takes the form of the letter V.

The development process proceeds from the upper left point of the V toward the right, ending at the upper right point. In the left-hand, downward-sloping branch of the V, the business requirements, the application design parameters and the design processes are defined. At the base point of the V, the code is written. In the right-hand, upward-sloping branch of the V, testing and debugging is done. The unit testing is carried out first, followed by bottom-up integration testing. The extreme upper right point of the V represents product release.

The V-model illustrates how QC (testing) activities are integrated into each phase of the software development life cycle.



### 4.1 Customer Requirements / Expert System Requirements

The Customer Requirements (initiated by the Product Department) are translated into Expert System Requirements, representing the high level solution for the Customer Requirements. The Expert System Requirements are composed by the

Expert Team (Development Department) and are approved by the Product Department.

## **4.2 System Requirements**

This represents the list of requirements. It captures the user's wishes from the system. System Use Cases are added to clarify these requirements. This document is composed by the Product Development Engineer. It is reviewed by the Development Manager, and the Software Engineer has to accept the document. The Product Development Engineer describes the Test Cases for the System Testing, related to the System Requirements.

## **4.3 Technical Design**

The Software Engineer creates the technical solution, based on the System Requirements. The Technical Software Architect approves the Technical Design document.

## **4.4 Component Design**

The Software Engineer breaks the designed system in to smaller units. The unit test design is developed in this stage.

## **4.5 Unit testing**

In the Unit Testing, both module structure and functionality are tested. These Unit Tests are performed on the units and modules while the programmer is coding them. The Software Engineer of a unit owns this unit and is responsible for testing it and finding any bugs contained in it.

## **4.6 Integration Testing**

In the Integration Testing, the separate modules are tested together to find weaknesses and bugs in the system. This way any bottlenecks in the system can be identified and the corresponding module can be redesigned. To evaluate the influence of the new developed modules on the system, automated tests of the existing functionalities are used. Software Engineers and Product Development Engineers are involved.

## **4.7 System Testing**

The System Testing compares the System Requirements against the actual system. Test Cases are derived from the Systems Requirements and use cases. The functionality of the system is tested based on these tests. New automated test files are initiated. The QC Team and the Product Development Engineers are involved.

## ***4.8 Acceptance Testing***

The goal of acceptance testing is to verify that the Customer Requirements and the agreed Expert System Requirements have been achieved. Acceptance testing is the final testing stage before taking the system into full operation. The Product Engineers are involved.

## 5 Related Processes and Procedures

### 5.1 *Software Product Life Cycle*

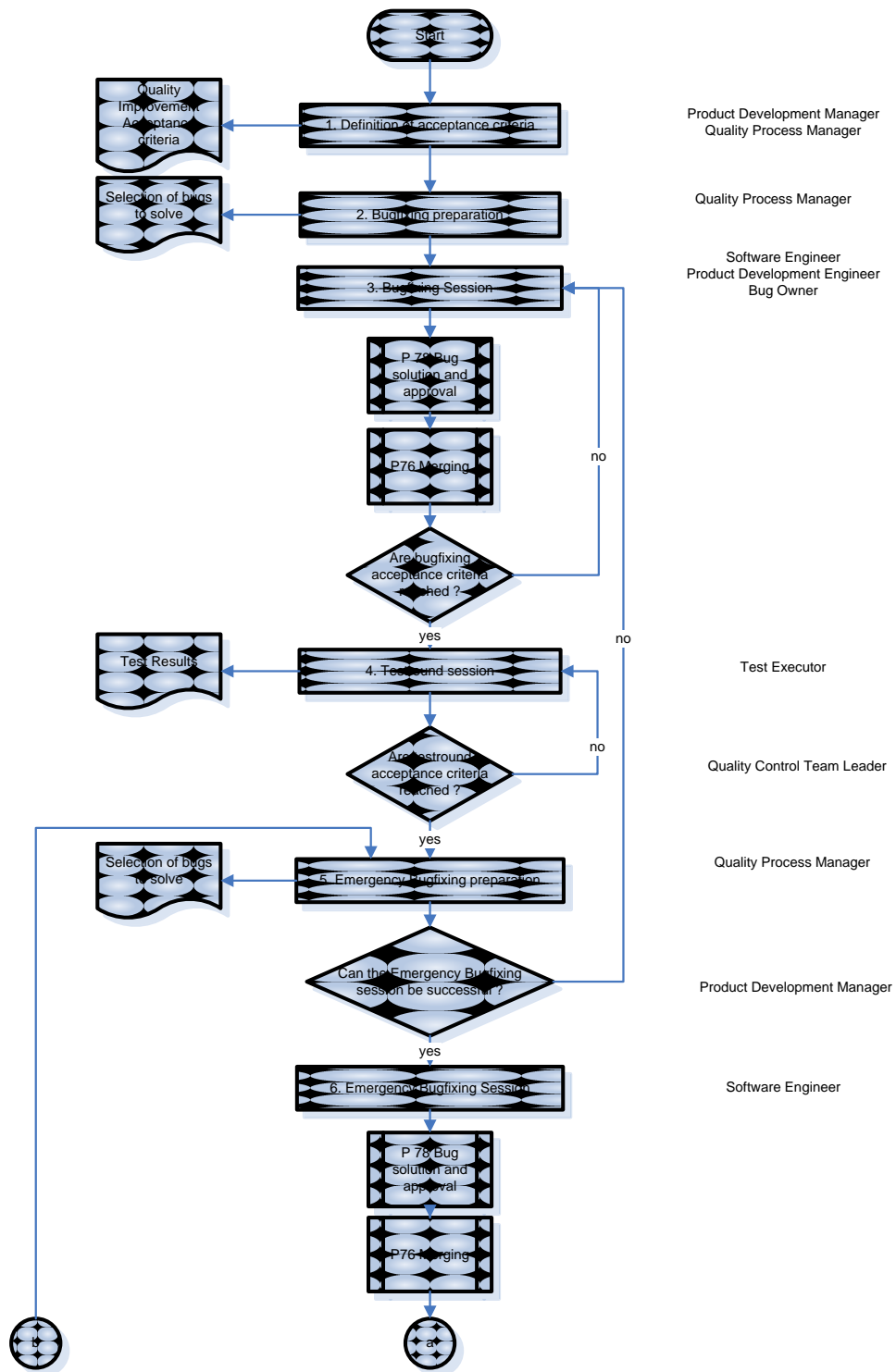
To support the SCIA QC policy and the V-Model, and taking into account the CMMI guidelines (Ref. [1]) , a set of processes and procedures are established and maintained. The total of these processes are representing the Software Product Life Cycle for Scia Engineer.

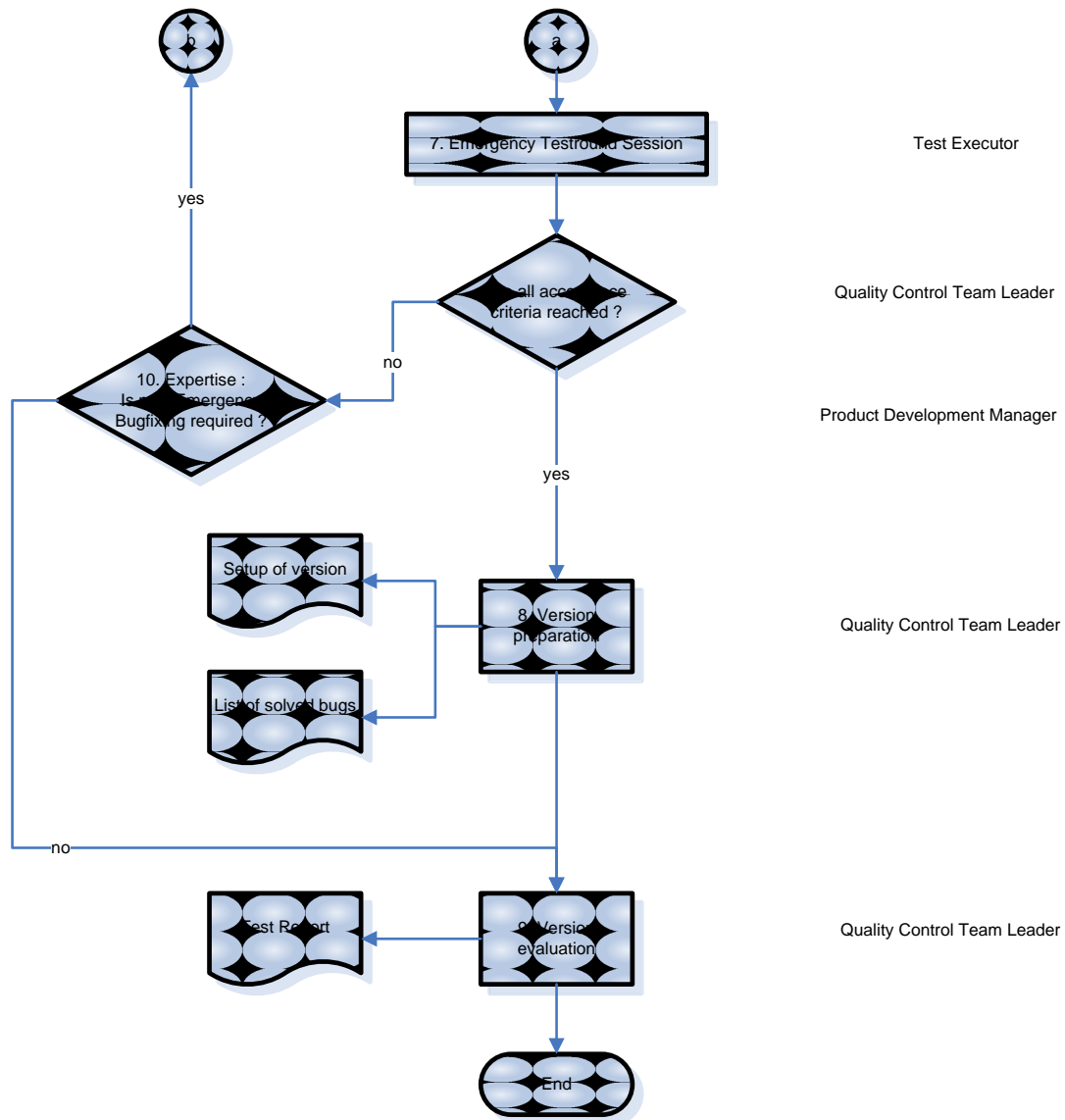
The development of the standard release is split in three main phases: the product vision and planning, the release planning and implementation, and the external launch of the release. The implementation is executed in several iterations, the so-called Sprints.

Inside this Software Product Life Cycle, the V-Model is applied on two levels: on the version (Release) level and on the iteration (Sprint) level.

### 5.2 *Quality Improvement Procedure*

Before a version is released (patch, release), the Quality Improvement Procedure is followed. The Quality Improvement Procedure is a sequence of bug fixing and testing, which ends when the Quality Improvement Acceptance criteria (Chapter 5.3) are reached.





### 5.3 Quality Improvement Acceptance Criteria

The standard Quality Improvement Acceptance Criteria contains the following criteria:

- Bug Fixing Acceptance criteria
- Test Round Acceptance criteria

#### 5.3.1 Bug Fixing Acceptance criteria

At the start of the Bug Fix Session, the number of open bugs Priority 2 + Priority 3 (see Chapter 5.3.3 for the bug priority definition) on the version is known. (= NP). The goal is to fix all NP bugs during the Bug Fix Session. Due to practical reasons, this goal is not realistic. Therefore exceptions can be accepted. The number of the allowed exceptions is set by a relative number (RN) and an absolute number (AN). The allowed exceptions are limited to  $\min(RN \times NP1 / 100, AN)$ .

The allowed exceptions must be justified by one of the following reasons: Holiday, Illness, External reason

If the completion criteria are not met, the Product Development Manager extends the session for the relevant persons. The extension can be overruled by the Product Board.

The actual values are RN = 10 %, AN = 25.

The Bug Fix Session Completion Criteria are as follows:

- All Priority 1, Priority 2 and Priority 3 Bugs must be solved
- All automated tests (see Chapter 5.4.) must be OK – exceptions must be clarified
- The allowed exceptions for Priority 2 + Priority 3 bugs are limited to min (RN x NP1 / 100, AN).

The 'Not Open' Bugs must be approved:

- All Priority 1 and Priority 2 & 3 bugs must be approved
- All P2+P3 must be approved with the exception criteria similar to the solved bugs, i.e. max. 25 or 10%, can be 'Not Approved'

### 5.3.2 Test Round Acceptance criteria

For the Test Round session, a set of manual test cases are selected to verify. The number of selected test cases varies with the nature of the intended test round (patch, release). The Test Round is considered as completed if 90 % of the test cases are executed (i.e. Test Execution Coverage – TEC of 90 %).

### 5.3.3 Definition Bug Priorities

Priority	Version	Definition	solution until
0 (Critical Bug)	R	Software cannot be used any more, crash or dangerous results	immediately
1 (Emergency Bug)	R	Serious problem	Next bug fix session, or actual bug fix session (if relevant)
2 (Standard Bug)	R	Serious problem	Next bug fix session
3	R	Small problem, UI, cosmetics bug, exceptional cases	Next bug fix session

A **Critical bug (P0)** is characterized by:

- The bug is found by a client and the client cannot continue with his project.
- Using of the program can cause wrong design of the structure or it could lead to the finance obligation of SCIA.
- The final assessment of the critical bug is performed by the Product Service Manager
- The bug is reported as Priority 0 in the R-version by the Product Service Manager
- The bug is taken up by the Product Development Director to be fixed immediately
- The follow-up is executed by the Product Service Manager

In case of absence of the Product Service Manager, the Quality Process Manager will replace him.

For practical reasons, the Development Manager has also the right to set the Priority 0.

An **Emergency Bug (P1)** is characterized by :

- The bug was reported before the start of the previous bug fix session, and was left as allowed exception for Priority 2.
- Important bug found during the test round, which has to be solved in next patch / release.
- The bug is reported as Priority 1 in the R-version

## **5.4 Automated Testing**

For each new build, the complete set of automated test cases is executed. The automated test cases are used to test the results of the solver and post processing (steel code, concrete code, ...).

For this purpose, a tool called Poirot is developed. It enables to run Scia Engineer in batch mode, export the defined results and compare them to the correct expected status. The tool can check numerical and graphical results.

Currently, there are approximately 7500 automated test cases that are run after each build. The total time to run the cases is 15 hours. By using dedicated hardware and virtual machines, the elapse time is reduced to 1 à 2 hours.

The results of the automated test cases are part of the acceptance criteria – see Chapter 5.3.1.

## **5.5 QA activities**

Quality Assurance (QA) is a planned and systematic means for assuring management that the defined standards, practices, procedures and methods of the products are applied.

The processes are evaluated by means of regular internal audits.

On top of the regular peer reviews, executed inside the Product Department and the Development Department, the documents are evaluated if they are consistent, brief, clear, complete, unique, verifiable and traceable, as described in the relevant check lists, by a member of the Quality Department. The selected work products for evaluation are the Customer Requirements, the System Requirement and the Test Case.

## 6 References

- [1] Chrissis M.B., Konrad M., Shrum S.  
CMMI Guidelines for Process Integration and Product Improvement  
Addison Wesley, 2003