

Title: Component Validation Process: Concept of component validation

Title:
Component Validation Process: Concept of component validation
Author(s)/Organisation(s):
Gyula IVÁN / FÖMI
Working Group:
WP6 – Component Validation Process
References:
-

Short Description:
This paper deals with the ideas and possible solutions of FÖMI for the component validation process
Keywords:
software verification, validation, test specification

History:			
Version	Author(s)	Status	Comment
000	Gyula IVÁN	new/rfc/final	
001	Gyula IVÁN, Gábor SZABÓ, László BÉRCZES		

Table of contents

1 Introduction.....	4
2 Definitions.....	4
2.1 Acceptance testing.....	4
2.2 Component testing.....	4
2.3 Final qualification testing.....	4
2.4 Software integration testing.....	5
2.5 Software verification.....	5
2.6 Software validation.....	5
2.7 System testing.....	5
2.8 Test case.....	5
2.9 Test design.....	5
2.10 Test plan.....	5
2.11 Software verification and validation plan.....	6
2.12 Software quality assurance plan.....	7
3 Verification and Validation phases.....	8
3.1 Support.....	8
3.1.1 Management and planning of verification and validation.....	8
3.1.2 Issue and Risk Tracking.....	8
3.1.3 Final Report.....	9
3.1.4 V&V Tool Support.....	9
3.1.5 Management and Technical review support.....	9
3.1.6 Criticality analysis.....	9
3.2 Concept.....	9
3.2.1 Reuse analysis.....	9
3.2.2 System architecture assessment.....	10
3.2.3 System requirements review.....	10
3.2.4 Concept documentation evaluation.....	10
3.2.5 Software/User requirements allocation analysis.....	10
3.2.6 Traceability Analysis-on system.....	11
3.3 Requirements.....	11
3.3.1 Traceability analysis-on requirements.....	11
3.3.2 Software requirement evaluation.....	11
3.3.3 Interface analysis – in requirements phase.....	13
3.3.4 System test plan analysis.....	13
3.3.5 Acceptance test plan analysis.....	13
3.3.6 Timing and sizing analysis.....	14
3.4 Design.....	14
3.4.1 Traceability Analysis – in design phase.....	14
3.4.2 Software design evaluation.....	14
3.4.3 Interface analysis – in design phase.....	15
3.4.4 Software final qualification test plan analysis.....	16

3.4.5 Software integration test plan analysis.....	16
3.4.6 Database analysis.....	17
3.4.7 Component test plan analysis.....	17
3.4.8 Data flow analysis.....	17
3.5 Implementation.....	18
3.5.1 Traceability Analysis – on code.....	18
3.5.2 Source code and documentation evaluation.....	18
3.5.3 Interface analysis - code.....	19
3.5.4 System test case analysis.....	20
3.5.5 Software final qualification test case analysis.....	20
3.5.6 Software integration test case analysis.....	20
3.5.7 Acceptance test case analysis.....	20
3.5.8 Software integration test procedure analysis.....	20
3.5.9 Software integration test results analysis.....	20
3.5.10 Component test case analysis.....	21
3.5.11 System test procedure analysis.....	21
3.5.12 Software final qualification test procedure analysis.....	21
3.6 Test	21
3.6.1 Traceability analysis – in test phase.....	21
3.6.2 Regression test analysis.....	21
3.6.3 Simulation analysis.....	21
3.6.4 System test results analysis.....	22
3.6.5 Software final qualification test results analysis.....	22
4 Test processes.....	23
5 Software component validation methods.....	24
6 Summary.....	26

1 Introduction

The aim of component validation process is to ensure that the developed software components are of sufficient quality to match the requirements formulated in WP5. Component validation process is a quality control procedure, which has to define the different characteristics of the implementation of the framework.

For the development of such a process different materials and knowledge bases must be taken into account:

- Evaluation of different standards on software quality, verification and testing,
- Experiences of software manufacturers of our consortium in software quality control and management,
- Experiences of public sector in software verification and validation,
- Experiences on reaction of software users.

Component Validation Process has many connection points with other workpackages within HUMBOLDT project. Validation process has to cover the implementation of framework development (WP8), scenario applications (WP9). It is not explicitly written in project documentation, but data harmonization (WP7) has a great influence on validation process too. As above mentioned, the main input of component validation process will be formulated in WP5 (Framework Interface). The first interface prototype will be delivered on the 12th month of the project. Therefore it is an initial paper, where FÖMI, as the component validation process workpackage leader, defines his ideas, definitions related to component validation process for the successful cooperation and thinking together with partners.

2 Definitions

2.1 Acceptance testing

Acceptance testing is testing conducted in an operational environment to determine whether a system satisfies its acceptance criteria (i.e. initial requirements and current needs of its user). Acceptance testing enables the customer to determine whether to accept the system.

2.2 Component testing

Component testing verifies the correct implementation of the design and compliance with program requirements for one software element (e.g. unit, module).

2.3 Final qualification testing

Final qualification testing the complete software program verifies that the software meets all of the software requirements and is ready to be integrated with system hardware.

2.4 Software integration testing

Software integration testing is an orderly progression of testing of incremental pieces of the software program in which software elements are combined and tested to show the compliance with the software design, capabilities and requirements. Software integration testing is also performed to verify operation under off-nominal and stress conditions.

2.5 Software verification

Software verification is the process, in which certified, that the software in a development phase satisfies all the requirements, which has been specified in the previous phase.

Verification step-by-step theoretically is enough to certify equivalency between the starting and finishing phase. But, because the whole process generally is not exact, a separate final verification is necessary, which have to be executed between the specification and the end product. If there was a mistake or defect in specification, the end-product will not satisfy the user requirements, therefore a separate control process is needed, in which the product is controlled from original function point of view. This process is validation

2.6 Software validation

Software validation is the process of evaluating software at the end of its software development process to ensure compliance with software requirements. This process ensures that the software system performs to the customer's expectations under operational conditions.

2.7 System testing

Systems testing is an orderly progression of testing of incremental pieces of the system in which both hardware and software elements are combined and tested until the entire system has been integrated. System testing verifies the requirements of the system and validates whether the system meets its original objectives.

2.8 Test case

Test case is a documentation that specifies inputs, predicted results, and sets of execution conditions for a test item

2.9 Test design

Test design is a documentation that specifies the details of the test approach for a software feature or combination of software features. The test design identifies the associated tests.

2.10 Test plan

Test plan is a documentation that specifies the scope, approach, resources and schedule if intended test activities.

2.11 Software verification and validation plan

Software verification and validation plan (V&V plan) fixes the regulations, which related to the control activities carried out during the software development. V&V plan provides a guideline for the checking the accomplishment of requirements including in software specification, and tests must be executed within the development phases.

The V&V plan describes the principles and methods of checks be used during the development, in order to provide, that the software satisfies the quality requirements.

The V&V plan unambiguously determines the liability, responsibility of testers and their contacts with the developers. This part must be refresh continuously during the project period.

The V&V plan must contain the followings:

- The location of V&V within the development, its organization method and the list of persons, who are accredited to V&V,
- Necessary resources,
- The persons, who are responsible for the execution of tasks,
- The instruments and methods are used during the testing,
- The system of documentation of results of testing, the contents of reports,
- Order of problem indication and solution, criterions of execution of retesting,
- Conditions and order of deviations from V&V plan,
- Systematisation and storing of documents, utilities, tools, mailing derived from testing,
- Standards, inner regulations, practices complied during testing activities.

Besides these content, the V&V plan appoints for each development phase separately:

- Connected checking (testing) tasks,
- Methods must be followed,
- Acceptance and decision criterions must be used,
- Input and Output document list of the phase,
- Time available for the execution of test,
- List of necessary instruments, equipments, professionals etc.,
- Risks derived from failure of the phase and handling of them,
- Role and responsibility of members of testing group.

2.12 Software quality assurance plan

Software quality assurance plan synthesizes the activities and rules related to quality assurance during the whole development cycle of the software. Software quality assurance plan must contain at least the followings:

- Aims of quality assurance plan,
- Catalogue of referenced documents,
- Method of project management and organization,
- Documents to be delivered during the development, and their contents,
- Standards, rules and conventions to be taken into account,
- Testing activities and order of testing,
- Scheme of problem indication and solution, personal responsibility,
- Used instruments, methods and methodologies,
- Archiving of source code, version management, handling of program (management of software configuration),
- Procedures against unauthorized access and physical damages,
- Methods of storing and systematisation of documentation.

3 Verification and Validation phases

3.1 Support

3.1.1 Management and planning of verification and validation

For the management and planning of verification and validation a V&V plan is necessary for all life cycle processes. It is possible to use a V&V template, when developing this plan. The V&V plan may require updating throughout the life cycle. Outputs of other workpackages (WP5, WP8, WP10) are inputs to the V&V plan and V&V plan should be updated when these inputs become available.

In the V&V plan the project milestones must be marked. V&V tasks should be scheduled to support project management and technical reviews. The interface among V&V effort, project management and the developer and architectural team must be planned. Data exchange requirement should be documented in V&V plan.

A time schedule is necessary for each V&V task, which should be planned. Identification of development processes and products (from WP8) to be evaluated by V&V process is necessary. The V&V plan should be coordinated with the developer of the softwares (WP8).

Reviewing and summarizing the V&V effort is necessary to define changes to V&V tasks or to redirect the V&V effort. Recommendation to whether to proceed the next set of V&V and development life cycle activities, provision task reports, anomaly reports and V&V activity summary report to project management are also needed.

Verification is necessary to:

- V&V tasks comply with task requirements defined in the V&V plan and/or task order,
- V&V task results have a basis of evidence supporting the results.

Assessment of all V&V results and provision of recommendations for program acceptance and certification is also needed as input to the V&V final report.

Management should evaluate proposed changes to the project (e.g. anomaly corrections, requirements changes) for effects on previously completed V&V tasks and future V&V tasks. Verification of this change that is consistent with system requirements and does not adversely affect other requirements directly or indirectly is necessary. An adverse effect is a change that could create new system hazards and risks or impact previously resolved hazards and risks. Planning of iteration of affected tasks or initiation of new tasks to address the software change or iterative development process is needed.

3.1.2 Issue and Risk Tracking

Tracking of V&V generated issues from initiation through closure by V&V team (WP6) is necessary. In issues and risk tracking the following activities are required:

- Identification and tracking V&V project risks and developmental project risks,
- Provision of recommendations to mitigate these risks,
- Communication of the issues and risks to the appropriate V&V and developer WPs.

3.1.3 Final Report

In final report the summarization of V&V activities, tasks and results, including status and disposition of anomalies and risks is required. Provision of an assessment of the overall software quality, lessons learned and recommendations for the future project must also be included in final report.

3.1.4 V&V Tool Support

A plan on the tools needed to support V&V effort is also recommended. The plan includes a description of each tool's performance, required inputs, outputs generated, and cost of tool purchase or development. The plan should also describe test facilities and integration supporting V&V effort. The scope and rigor of V&V effort as defined by the selected software integrity level should be considered in defining the performance required of each tool.

3.1.5 Management and Technical review support

For the supporting of project management reviews and technical reviews review materials, attending the reviews, presenting at the reviews, providing task and anomaly reports are also necessary.

3.1.6 Criticality analysis

For the criticality analysis the following actions are required:

- Elaboration of a software integrity level scheme using the available documentation (e.g. system requirements, subsystem requirements, software specifications, hazard analyses etc.),
- Assessment the software components against the software integrity level scheme,
- Documentation of software integrity level assigned to individual software components (e.g. requirements, detailed functions, software modules, subsystems, or other software partitions).

For V&V planning purposes, the most critical software integrity level assigned to individual elements shall be the integrity level assigned to the entire software. Therefore the following verification actions must be executed:

- Verification of that, whether any software component can influence individual software components assigned a higher software integrity level, and if such conditions exist, assign that software component the same higher software integrity level.
- Verification of that, whether any software component controls or mitigates a hazardous event and of that the proper software integrity level is assigned to those components.

3.2 Concept

3.2.1 Reuse analysis

In reuse analysis step the following actions should be taken into account:

- Analysis of the developers' documentation to verify that the original domain of the candidate reuse software will satisfy the domain of the new system (e.g. software integrity level, user needs, operating environment, safety, security, interfaces etc.). Domain analysis is also

Title: Component Validation Process: Concept of Component Validation

needed to compare the original domain and the new domain (HUMBOLDT) of the candidate reuse software.

3.2.2 System architecture assessment

For system architecture assessment the following actions should be done:

- Assessment of the proposed architecture schema for feasibility,
- Assessment of that, how the proposed architecture satisfies the users' needs in terms of user requirements,
- Examples of requirements are timing, storage, usability, safety, security and suitability for HUMBOLDT mission.

3.2.3 System requirements review

In system requirements the following tasks should be executed:

- Analysis of system requirements (e.g. system requirements specification, feasibility study report, business rules description to:
 - Verify the consistency of requirements to user needs,
 - Validate whether the requirements can be satisfied by the defined technologies, methods, and algorithms defined for HUMBOLDT,
 - Verify whether objective information that can be demonstrated by testing is provided in the requirements (testability). Review other requirements such as deliverable definitions, listing of appropriate compliance standards and regulations, user needs etc. for completeness, correctness and accuracy.

3.2.4 Concept documentation evaluation

Concept documentation must satisfy user needs and is consistent with acquisition needs. Validation of constraints of interfacing systems and constraints or limitations of proposed approaches is also needed. The user needs to satisfy are the following:

- System functions,
- End-to-end system performance,
- Operation and maintenance requirements,
- Migration requirements from an existing system, where applicable.

3.2.5 Software/User requirements allocation analysis

Verification of correctness, accuracy and completeness of the concept requirement allocation to software and user interfaces against user needs is also required. In verification process the following actions should be executed:

- Correctness: Verification of performance requirements (e.g. timing, response time, throughput) allocated to software and user interfaces, that satisfy user needs.

Title: Component Validation Process: Concept of Component Validation

- Accuracy: Verification of the internal and external interfaces, which specify the data formats, interface protocols, frequency of data exchange at each interface, and other key performance requirements to demonstrate compliance with user requirements.
- Completeness: In completeness verification process should:
 - Verify that application specific requirements such as functional diversity, fault detection, fault isolation, and diagnostic and error recovery satisfy user needs,
 - Verify that the users' maintenance requirements for the system are completely specified,
 - Verify that the migration from existing system and replacement of the system satisfy user needs

3.2.6 Traceability Analysis-on system

In traceability analysis the identification of all system requirements, which will be implemented completely or partially by the software, should be executed. Verification of these system requirements is required, that are traceable to user or program requirements. The traceability analysis should be started with the system requirements.

3.3 Requirements

3.3.1 Traceability analysis-on requirements

Tracing the software requirements (software specifications, interface specifications) to system requirements (concept) and system requirements to the software requirements is necessary. In this analysis the following actions should be carried out:

- Correctness: Validation of correctness of the relationships between each software requirement and its system requirement,
- Consistency: Verification of that, the relationship between the software and system requirements are specified to a consistent level of detail,
- Completeness: In completeness analysis should:
 - Verify that every software requirement is traceable to a system requirement with sufficient detail to show compliance with the system requirement,
 - Verify that all system requirements related to software are traceable to software requirements,
- Accuracy: Validation of that, the system performance and operating characteristics are accurately specified by the traced software requirements.

3.3.2 Software requirement evaluation

In software requirement evaluation the requirements (e.g. functional, capability, interface, qualification, safety, security, human factors, data definitions, user documentation, installation and acceptance, user operation and user maintenance) of system requirements and interface requirements specification

Title: Component Validation Process: Concept of Component Validation

should be investigated for correctness, consistency, completeness, accuracy, readability, and testability. Task criteria are the follows:

- Correctness:
 - Verify and validate that the software requirements satisfy the system requirements allocated to software within the assumptions and constraints of the system,
 - Verify that the software requirements comply with applicable standards, references, regulations, policies and business rules,
 - Validate that the flow of data and control satisfy functionality and performance requirements,
 - Validate data usage and format,
- Consistency:
 - Verify that all terms and concepts are documented consistently,
 - Verify that the function interactions and assumptions are consistent and satisfy system requirements and acquisition needs,
 - Verify that there is internal consistency between the software requirements and external consistency with the system requirements,
- Completeness:
 - Verify that the following elements are in software requirements and interface requirements specifications within the assumptions and constraints of the system:
 - Functionality (e.g. algorithms, state/mode definitions, input/output validation, exception handling, reporting and logging),
 - Process definition and scheduling,
 - Hardware, software and user interface descriptions,
 - Performance criteria (e.g. timing, sizing, speed, capacity, accuracy, precision, safety, and security),
 - Critical configuration data,
 - System, device, and software control (e.g. initialisation, transaction and state monitoring, self.testing),
 - Verify that the system requirements and interface requirements specifications satisfy specified configuration management procedures,
- Accuracy:
 - Validate that the logic, computational, and interface precision (e.g. truncation and rounding) satisfy the requirements in the system environment,
- Readability:

Title: Component Validation Process: Concept of Component Validation

- Verify that the documentation is legible, understandable, and unambiguous to the intended audience,
- Verify that the documentation defines all acronyms, mnemonics, abbreviations, terms and symbols,
- Testability:
 - Verify that there are objective acceptance criteria for validation the requirements of the system requirements and interface requirements specifications.

3.3.3 Interface analysis – in requirements phase

In interface analysis (in requirements phase) verification and validation of that, the requirements for software interfaces with hardware, user, operator and other systems are correct, consistent, complete, accurate and testable is needed. Criteria are the follows:

- Correctness: Validate the external and internal system and software interface requirements,
- Consistency: Verify that the interface descriptions are consistent between the system requirements and interface requirements specifications,
- Completeness: Verify that each interface is described and includes data format and performance criteria (e.g. timing, bandwidth, accuracy, safety and security),
- Accuracy: Verify that each interface provides information with the required accuracy,
- Testability: Verify that there are objective acceptance criteria for validating the interface requirements.

3.3.4 System test plan analysis

In system test plan analysis the verification of that, the system test plan conforms to HUMBOLDT defined test document purpose, format and content is needed. The criteria are the follows:

- Test coverage of system requirements,
- Appropriateness of test methods and standards used,
- Feasibility of system qualification testing,
- Feasibility and testability of operation and maintenance requirements.

3.3.5 Acceptance test plan analysis

In acceptance test plan analysis the verification of that, the acceptance test plan complies with HUMBOLDT defined test document purpose, format and content is necessary. Criteria are the follows:

- Test coverage of system requirements,
- Feasibility of operation and maintenance (e.g. capability to be operated and maintained in accordance with user needs).

3.3.6 Timing and sizing analysis

In this step collection and analysis of data about the software functions and resource utilizations are needed to determine if system and software requirements for speed and capacity are satisfied. The types of functions and resource utilization issues include, but are not limited to the following:

- CPU Load,
- RAM and secondary storage,
- Network speed and capacity,
- Input and output speed.

Sizing and timing analysis is started at software design and iterated through acceptance testing.

3.4 Design

3.4.1 Traceability Analysis – in design phase

In this step the analysis of trace relationships between design elements (System and Interface Design) and requirements (system and interface requirements) for correctness, consistency and completeness is needed. The criteria are the follows:

- Correctness: Validate the relationships between each design element and software requirement,
- Consistency: Verify that the relationships between the design elements and the software requirements are specified to a constant level of detail,
- Completeness:
 - Verify that all design elements are traceable from the software requirements,
 - Verify that all software requirements are traceable to the design elements.

3.4.2 Software design evaluation

In this section the evaluation of design elements for correctness, consistency, completeness, accuracy, readability and testability is required. Criteria are the follows:

- Correctness: Verify and validate that the software design satisfies the software requirements. Verify that the software design complies with applicable standards, references, regulations, policies and business rules. Validate the software design sequences of states and state changes using logic and data flows coupled with domain expertise, prototyping results, engineering principles and other basis. Validate that the flow of data and control satisfy functionality and performance requirements. Validate data usage and format. Assess to appropriateness of design methods and standards,
- Consistency: Verify that all terms and design concepts are documented consistently. Verify that there is internal consistency between the design elements and external consistency with architectural design,

Title: Component Validation Process: Concept of Component Validation

- **Completeness:** Verify that the following elements are in the System Design, within the assumptions and constraints of the system:
 - Functionality (e.g. algorithms, state/mode definitions, input/output validation, exception handling, reporting and logging),
 - Process definition and scheduling,
 - Hardware, software and user interface descriptions,
 - Performance criteria (e.g. timing, sizing, speed, capacity, accuracy, precision, safety, and security),
 - Critical configuration data,
 - System, device, and software control (e.g. initialisation, transaction and state monitoring, self.testing),

Verify that the System Design and Interface Design satisfy specified configuration management procedures.

- **Accuracy:** Validate that the logic, computational, and interface precision (e.g. truncation and rounding) satisfy the requirements in the system environment.
- **Readability:**
 - Verify that the documentation is legible, understandable, and unambiguous to the intended audience,
 - Verify that the documentation defines all acronyms, mnemonics, abbreviations, terms and symbols, and design language if any.
- **Testability:**
 - Verify that there are objective acceptance criteria for validating each software design element and the system design. Verify that each software design element is testable to objective acceptance criteria.

3.4.3 Interface analysis – in design phase

In interface analysis (in design phase) verification and validation of that, the software design interfaces with hardware, user, operator and other systems are correct, consistent, complete, accurate and testable is needed. Criteria are the follows:

- **Correctness:** Validate the external and internal software interface design in the context of system requirements,
- **Consistency:** Verify that the interface design is consistent between the system design and interface design,
- **Completeness:** Verify that each interface is described and includes data format and performance criteria (e.g. timing, bandwidth, accuracy, safety and security),
- **Accuracy:** Verify that each interface provides information with the required accuracy,

Title: Component Validation Process: Concept of Component Validation

- Testability: Verify that there are objective acceptance criteria for validating the interface design.

3.4.4 Software final qualification test plan analysis

In this step the verification of the final qualification test plan is carried out, that it complies with HUMBOLDT defined test document purpose, format and content. Criteria are the follows:

- Traceable to the software requirements,
- External consistency with the software requirements,
- Internal consistency,
- Test coverage of the software requirements,
- Appropriateness of test standards and methods used,
- Feasibility of software qualification testing,
- Feasibility on operation and maintenance (e.g. capability to be operated and maintained in accordance with user needs).

3.4.5 Software integration test plan analysis

In this phase the verification of integration test plan is necessary, that it complies with HUMBOLDT defined test purpose, format and content.

Criteria are the follows:

- Compliance with increasingly larger set of functional requirements at each stage of integration,
- Assessment of timing, sizing and accuracy,
- Performance at boundaries and under stress conditions,
- Traceable to the software requirements,
- External consistency with the software requirements,
- Internal consistency,
- Test coverage of the software requirements,
- Measures of requirements test coverage and software reliability,
- Appropriateness of test standards and methods used,
- Feasibility of software qualification testing,
- Feasibility on operation and maintenance (e.g. capability to be operated and maintained in accordance with user needs).

3.4.6 Database analysis

Evaluation of database design includes the following:

- Physical limitations analysis: Identify the physical limitations of the database such as maximum number of records, maximum record length, largest numeric value, smallest numeric value, maximum array length in a data structure and compare them to designed values,
- Index vs. Storage Analysis: Analyse the use of multiple indexes compared to the volume of stored data to determine if the proposed approach meets the requirements for data retrieval performance and site constraints,
- Data Structure Analysis: Some DBMS have specific data structures within a record, such as arrays, tables and date formats. Review the use of these structures for potential impact on requirements for data storage and retrieval,
- Backup and Disaster Recovery Analysis: Review the methods employed for backup against the requirements for data recovery and system disaster recovery and identify deficiencies.

3.4.7 Component test plan analysis

In this phase the verification of component test plan is necessary, that it complies with HUMBOLDT defined test purpose, format and content.

Criteria are the follows:

- Compliance with design requirements,
- Assessment of timing, sizing and accuracy,
- Performance at boundaries and interfaces and under stress and error conditions,
- Traceable to the software requirements and design,
- External consistency with the software requirements and design,
- Internal consistency between unit requirements,
- Test coverage of requirements in each unit,
- Measures of requirements test coverage and software reliability and maintainability,
- Feasibility of software integration and testing,
- Feasibility on operation and maintenance (e.g. capability to be operated and maintained in accordance with user needs).

3.4.8 Data flow analysis

Evaluation of data flow is including the followings:

- Symbology consistency check. Verify that each symbol is used consistently,

Title: Component Validation Process: Concept of Component Validation

- Flow Balancing. Compare the output data from each process block to the data inputs and the data derived within the process to ensure the data is available when required. This process does not specifically examine timing or sequence considerations,
- Confirmation of Derived Data. Examine the data derived within a process for correctness and format. Data designed to be entered into a process by operator action should be confirmed to ensure availability,
- Keys to Index Comparison. Compare the data keys used to retrieve data from data stores within a process to the database index design to confirm that no invalid keys have been used and the uniqueness properties are consistent.

3.5 Implementation

3.5.1 Traceability Analysis – on code

Tracing the source code components to the corresponding design specification, and design specification to source code components is needed. Analysis of identified relationships for correctness, consistency and completeness is required. Criteria are the follows:

- Correctness: Validate the relationship between source code components and design elements,
- Consistency: Verify that the relationship between source code components and design element are specified to a consistent level of detail,
- Completeness:
 - Verify that all source code components are traceable from design elements,
 - Verify that all design elements are traceable to the source code components.

3.5.2 Source code and documentation evaluation

This step means the evaluation of source code components for correctness, consistency, completeness, accuracy, readability and testability. Criteria are the follows:

- Correctness:
 - Verify and Validate that the source code component satisfies the software design,
 - Verify that the source code components comply with applicable standards, references, regulations, policies and business rules,
 - Validate the source code component sequences of states and state changes,
 - Validate that the flow of data and control satisfy functionality and performance requirements,
 - Validate data usage and format,
 - Assess the appropriateness of coding methods and standards,
- Consistency:

Title: Component Validation Process: Concept of Component Validation

- Verify that all terms and code concepts are documented consistently,
- Verify that there is internal consistency between the source code components,
- Validate external consistency with the software design and requirements,
- **Completeness:**
 - Verify that the following elements are in the source code, within the assumptions and constraints of the system:
 - Functionality (e.g. algorithms, state/mode definitions, input/output validation, exception handling, reporting and logging),
 - Process definition and scheduling,
 - Hardware, software and user interface descriptions,
 - Performance criteria (e.g. timing, sizing, speed, capacity, accuracy, precision, safety, and security),
 - Critical configuration data,
 - System, device, and software control (e.g. initialisation, transaction and state monitoring, self-testing),
 - Verify that the source code documentation satisfies specified configuration management procedures,
- **Accuracy:**
 - Validate the logic, computation, and interface precision (e.g. truncation and rounding) in the system environment,
- **Readability:**
 - Verify that the documentation is legible, understandable, and unambiguous to the intended audience,
 - Verify that the documentation defines all acronyms, mnemonics, abbreviations, terms and symbols, and design language if any.
- **Testability:**
 - Verify that there are objective acceptance criteria for validating each source code component,
 - Verify that each source code component is testable against objective acceptance.

3.5.3 Interface analysis - code

In interface analysis (in implementation phase) verification and validation of that, the software source code interfaces with hardware, user, operator and other systems are correct, consistent, complete, accurate and testable is needed. Criteria are the follows:

- **Correctness:** Validate the external and internal software interface code in the context of system requirements,

Title: Component Validation Process: Concept of Component Validation

- Consistency: Verify that the interface code is consistent between source code components and to external interfaces (e.g. hardware, user, operator and other software),
- Completeness: Verify that each interface is described and includes data format and performance criteria (e.g. timing, bandwidth, accuracy, safety and security),
- Accuracy: Verify that each interface provides information with the required accuracy,
- Testability: Verify that there are objective acceptance criteria for validating the interface code.

3.5.4 System test case analysis

In system test case analysis the verification of system test cases is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of system test cases is required, that they satisfy the criteria in System test plan.

3.5.5 Software final qualification test case analysis

In software final qualification test case analysis the verification of software final qualification test cases is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of software final qualification test cases is required, that they satisfy the criteria in software final qualification test plan.

3.5.6 Software integration test case analysis

In software integration test case analysis the verification of software integration test cases is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of software integration test cases is required, that they satisfy the criteria in software integration test plan.

3.5.7 Acceptance test case analysis

In acceptance test case analysis the verification of acceptance test cases is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of acceptance test cases is required, that they satisfy the criteria in acceptance test plan.

3.5.8 Software integration test procedure analysis

In software integration test procedure analysis the verification of software integration test procedures is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of software integration test procedures is required, that they satisfy the criteria in software integration test plan.

3.5.9 Software integration test results analysis

Usage of developers' integration test results is required to verify that the software components are integrated correctly. Verification of test results is needed, that they trace to the test criteria established by the test traceability in the test planning documents. Documentation of discrepancies between actual and expected results is required.

3.5.10 Component test case analysis

In component test case analysis the verification of component test cases is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of component test cases is required, that they satisfy the criteria in component test plan.

3.5.11 System test procedure analysis

In system test procedure analysis the verification of system test procedures is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of system test procedures is required, that they satisfy the criteria in system test plan.

3.5.12 Software final qualification test procedure analysis

In software final qualification test procedure analysis the verification of software final qualification test procedures is needed, that they comply with HUMBOLDT defined test document purpose, format, and content. Validation of software final qualification test procedures is required, that they satisfy the criteria in software final qualification test plan.

3.6 Test

3.6.1 Traceability analysis – in test phase

Analysis of relationships in the test plans, designs, cases and procedures is required for correctness and completeness. Criteria are the follows:

- Correctness:
 - Verify that there is a valid relationship between the test plans, designs, cases and procedures,
- Completeness:
 - Verify that all test procedures are traceable to test plans.
- Tracing must be executed only for test types and documents subject to V&V test analysis.

3.6.2 Regression test analysis

In this phase the determination of the extent of the testing is needed, that it must be repeated when changes are made to any previously examined software products. Assessment the nature of the changes is also required to determine the potential ripple of side effects and impacts on other aspects of the system.

3.6.3 Simulation analysis

This phase means the analysis of simulation for correctness, accuracy and completeness. Criteria are the follows:

- Correctness:
 - Verify that the simulation satisfies the simulation and system requirements,

Title: Component Validation Process: Concept of Component Validation

- Accuracy:
 - Validate that the simulation accurately represents the system environment,
- Completeness:
 - Verify that the simulation has all the functionality necessary to perform the intended testing.

3.6.4 System test results analysis

Usage of the developers' system test results is required to validate that the software satisfies the system requirements. Verification of test results is necessary, that they trace to the test criteria established by the test traceability in the test planning documents. Documentation of discrepancies between actual and expected results is also required.

3.6.5 Software final qualification test results analysis

Usage of the developers' software final qualification test results is required to validate that the software satisfies the software requirements. Verification of test results is necessary, that they trace to the test criteria established by the test traceability in the test planning documents. Documentation of discrepancies between actual and expected results is also required.

4 Test processes

Test process is worth executing in sections, where the testing can be carried out incrementally accordingly with the implementation of the system.

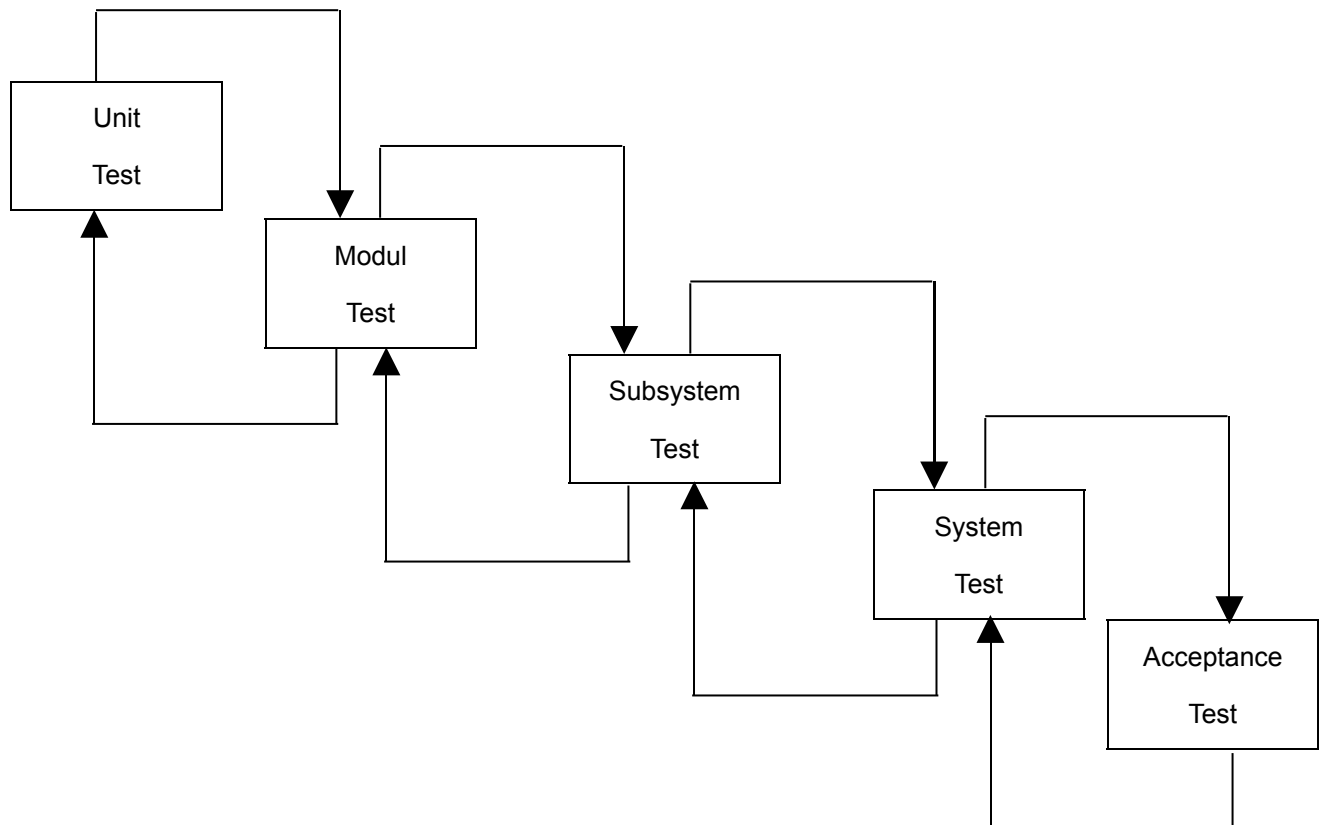


Figure 1.: Test process

Sections of test process:

Unit Test: The individual components have to be tested independently of the others, and be provided the faultless operating.

Modul Test: The modul is a group of interdependent components, the moduls also can be tested independently of the others

Subsystem test: The tests of moduls, which compose a subsystem. This test process focuses on the interface-errors of moduls, because the most problems are derived from the wrong connections of interfaces.

System test: This test phase deals with the errors, derived from the unanticipated interactions between the subsystems and their interfaces, and is concerned with validation too.

Acceptance test: The system is tested with the data of end-user, not with test data. Errors can be detected in real situation. Problems can be arisen, which show the system properties do not fulfil the user requirements.

Unit and modul test is generally the task of the programmer, who develops the component. The later sections of testing are the responsibility of independent groups of testers, based on test plans. Figure 2. shows the different testing phases in software developments.

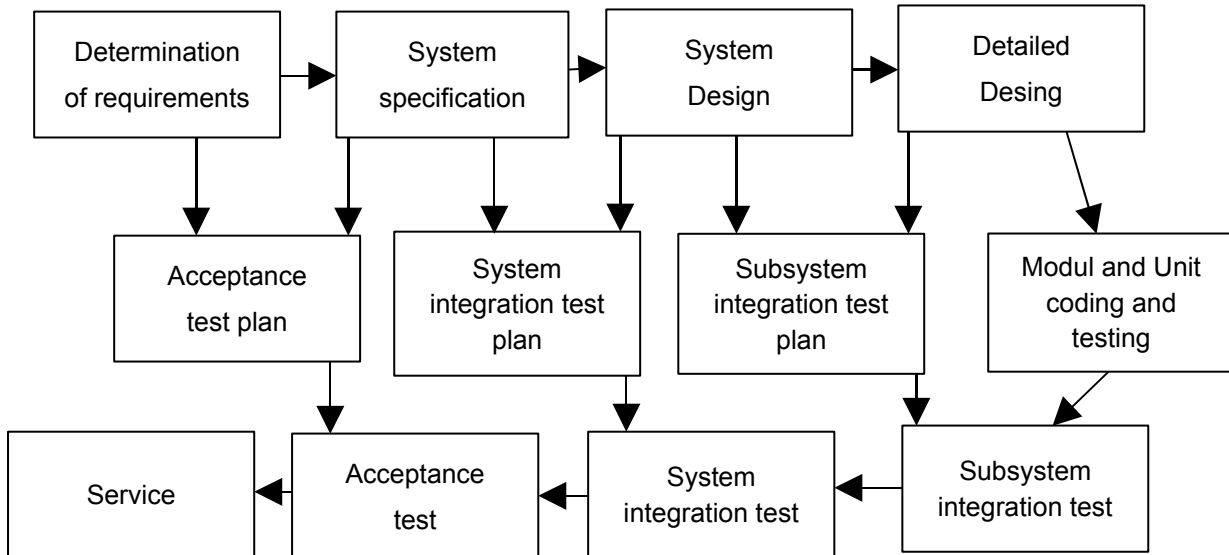


Figure 2.: Testing phases in software development

5 Software component validation methods

There are many different software validation methods for components. They could be classified into two classes:

- Black-box validation methods (also known as functional testing methods)
 - These methods refer to the systematic techniques for testers to design and generate test cases and data to achieve a certain test adequacy criteria for a component based on its component specifications.
- White-box validation methods (also known as program-based testing methods, or structure-based testing methods)
 - These methods refer to the systematic techniques for testers to design and generate test cases and data to achieve a certain test adequacy criteria for a component based on its component program and structure,

Black-box method's base idea is a testing of a component as a black box. It focuses only on the external accessible behaviors and functions. It checks a component's outputs for selected component inputs.

Black-box testing methods for components can be classified into three groups:

- Usage-based black-box testing techniques – focusing on user oriented accesses
 - User-operation scenario testing,
 - Random testing,

Title: Component Validation Process: Concept of Component Validation

- Statistical testing,
- Error-based black-box testing techniques – focusing on error-prone points
 - Equivalence partitioning testing,
 - Category-partitioning testing,
 - Boundary-value analysis,
 - Decision table-based testing,
- Fault-based black-box testing techniques – focusing on targeted fault points
 - Mutation testing,
 - Fault injection method.

White-box testing methods are focusing on the internal structures of a component, on the internal component logics and behaviors of a component. These testing methods can be used in two purposes:

- Component developers use white-box testing methods to discover program-oriented errors in a component development process,
- Component users use white-box testing methods to check program errors in a component based software development process in the following cases:
 - Creating and checking newly created components,
 - Validating the extended and customized parts of adopted components.

From white-box testing methods for components can be classified into three types of testing:

- Path testing (based on program flow graph),
- Data flow testing,
- Object-oriented testing.

Path testing:

- Exercise program control flows of a component to check its outputs based on the given inputs
 - Use program flow graph as a test model,
 - Generate test cases based on the test model,
 - Apply test coverage criteria based on the test model.

Data flow testing:

- Focuses on program errors relating to incorrect data definition and usage
 - Program flow graphs are used as test model,
 - Use program flow graph to identify various data flow paths, such as data define and use path,

Title: Component Validation Process: Concept of Component Validation

- Generate test cases to exercise various data flow paths to achieve the selected data flow test coverage criteria.

Object-oriented testing:

- Testing object-oriented program by focusing on object-oriented program structures and features,
- Generate test cases based on selected object-oriented test models to achieve pre-defined test coverage criteria,
- Basic approaches:
 - Object state-based testing,
 - Class relation-based testing, inheritance testing,
 - Class function-based dependency testing,
- Test models:
 - Object state chart,
 - Class relation graph,
 - Class function dependency graph.

6 Summary

In the last sections we wanted to describe the main elements necessary for component validation process. Of course in the implementation of HUMBOLDT framework the concrete solution will be different from component to component. Therefore the close cooperation with the developer workpackages is required very much.

Other important issue in verification/validation is the definition of measure factors, values, indicators for each testing results and phases. The classification of these values is also very important for a reliable test result.

Management of risks is another important issue. We have to define very accurately the risks and link to them the adequate acceptance level.

So there are many themes, which must be worked out in a close cooperation among HUMBOLDT partners.