Jan Jacobson, Bengt Johansson (Ed.)

MID Software Work Package 2

Methods for Validation and Testing of Software

Revision 29 September 2004

MID-Software WP2 Testing and Validation
www.mid-software.org

Abstract

The European Commission has issued a directive for measuring instruments. The Measuring Instruments Directive contains the essential requirements for measuring instruments used for legal purposes. Since many measuring instruments contain software, these requirements will also apply to the software of the instruments.

Testing of measuring instruments requires testing of software-based functionality. This includes aspects of the measuring functionality as well as aspects going beyond metrology-related functionality. This report collects validation methods (analysis and test methods) applicable to different measuring instruments.

Revision no	**Date**	**Changes made**	**Distribution**
0	2002-03-06	First issue	Project coordinator
1	2002-06-20	Chapter 3: More methods added Chapter 4: Ideas for validation plans introduced Chapter 5 added Several other modifications.	MID-Software Partners
2	2002-12-20	Chapter 3: Large restructuring of validation methods Chapter 4: Validation plans are based on the risks and on the functions of the instrument Annex D: Optional validation methods listed	MID-Software Partners
3	2003-02-28	Chapter 3: FMEA and Event Tree Analysis included. Chapter 4: validation plans re-designed to recommend validation methods for each requirement.	Working document to project coordinator, DELTA, JV
4	2003-11-27	Chapter 4 deleted. Validation plans integrated into Software Requirements Guide. Several other modifications.	MID-Software Partners
5	2004-09-17	Old chapter 3.3 and annex B moved to the MID Software Guide. Validation methods in chapter 3 rearranged. Several minor modifications.	MID-Software Partners
6	2004-09-29	Minor modifications.	MID-Software Partners

Contents

Preface

This report has been developed within the thematic network MID-Software. The network was started as a part of the European Research Programme for Competitive and Sustainable Growth (project no. G7RT-CT-2001-05064). Testing and validation has been included as work package 2 of the network. The description of validation methods is a supplement to the Software Requirements Guide developed by the MID-Software project. The document was developed step-by-step until a final WP2 report was achieved at the end of the MID-Software project.

Following organisations have been participating in WP2 of MID-Software:
PTB (Germany)
DELTA (Denmark)
GUM (Poland)
HALE (Austria)
Herbert and Sons (UK)
IPQ (Portugal)
Justervesenet (Norway)
Marconi (Italy)
Mettler Toledo (Switzerland)
NMI (The Netherlands)
NWML (UK)
Sartorius (Germany)
MIRS (Slovenia)
SP Swedish National Testing and Research Institute (Sweden)
LNE (France)

We thank all the above partners and everyone else who has commented and contributed to the report.

Jan Jacobson and Bengt Johansson
SP Swedish National Testing and Research Institute
Editors of the MID-Software WP2 report

Summary

A set of validation methods is needed to validate the requirements of software in measuring instruments. Different methods are applied depending on the risks associated with the instrument, and on the functions implemented by the instrument. There is no single “silver bullet” suitable to solve any validation task.

The validation methods may be grouped in three groups; checks based on documentation, functional checks and checks based on the source code.

Some of the validation methods can be used without detailed knowledge of the software. An example of such a method is examination of the operator’s manual. The examination of the operator’s manual will answer basic questions about the functionality without examining the software in detail.

Detailed knowledge of the software is needed for in-depth examination. High risks or complex questions may require validation methods such as source code analysis or design reviews with the software engineers.

Validation guidance is given for each requirement in the Software Requirements Guide. The validation methods in this report “Methods for Validation and Testing of Software” are intended to be used together with that guide.

1 Introduction

Tests of measurement accuracy are well known to both testing laboratories and to manufactures of measuring instruments. The requirements for accuracy are established and well understood. Exchange of experience and comparative measurements ("round robin tests") are regularly performed to improve the comparability of measurements at different laboratories.

Tests of software and functionality of measuring instruments is not yet a mature discipline. Manufactures have to test their instruments before they can be put into serial production. Some testing laboratories make software tests based on recommendations of WELMEC or the OIML. Also nationally developed test methods for specific measuring instruments exist. A common understanding for how to use different test methods and validation techniques should be developed.

The European users of measuring instruments regard the tests of measuring instruments performed by the notified bodies to be exhaustive and high quality. That is definitely true for many aspects, but tests of functionality and software may not always be state-of-the-art. There is often a gap between the reasonable efforts applied by the independent laboratory, and the exhaustive testing which might be presumed by the user. A balance must be established between what can be expected as reasonable at the laboratory, and the need for confidence in the measuring instrument.

One of the intentions behind the examination of the software of a measuring instrument is that the national authorities should be able to check whether the software is in conformance with the type approval.

The different methods for test and validation have to be harmonised, a common frame of reference has to be established and recommendation for which test methods to use should be established.

These test methods are explained and collected as a "tool box" which can be used differently for different measuring instruments. The test methods can then be combined for use with different measuring instruments. The Software Requirements Guide [MIDSwReq] lists the requirements and gives the corresponding validation guidance.

2 Testing and Validation of Software

2.1 Examination, testing, analysis and validation

Different words are used to describe the examination of software. The terms used depend on the situation and on the established usage of language in each sector. A development engineer may "check" his software before he releases a software module for integration with other modules. A project manager may "test" a software package before it is released. A notified body may "validate" the software of a measuring instrument before a type approval is issued.

There are some established definitions of terms:
verification : confirmation by examination and provision of objective evidence that the requirements have been fulfilled [IEC 61508-4, clause 3.8.1]

validation : confirmation and provision of objective evidence that the particular requirements for a specific intended use are fulfilled [IEC 61508-4, clause 3.8.2]

software validation : confirmation by examination and provision of objective evidence that software specification conform to the user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled [FDA, clause 3.1.2]

The term "software validation" is used in this report to describe the activity to confirm and provide evidence that the requirements of the MID are fulfilled also by the software. The terms “examination”, “checking” and “evaluation” are also used, and are considered to be synonymous to “software validation”.

2.2 Requirements

The essential requirements for all measuring instruments, and the type-specific requirements for different types of instruments are listed in the measuring instruments directive. [MID]

A measuring instrument of high quality must have a suitable and fault-free functionality. The functions specified and documented by the manufacturer must be implemented in a fault-free way. The testing of the functionality of the instrument is an important part of the type testing. However, this report is focused on the examination of software to validate the requirements of the MID, and not to make an exhaustive test of all functionality. Some of the validation methods described may also be used to validate additional functional requirements, but that is not in the scope of this report.

2.3 Process or product?

Software can be evaluated either by evaluation of the software itself (the product) or how it has been developed (the process). Both aspects are important to gain confidence in a software package.

The validation methods described in this report are focused on the software product. This is consistent with the traditional approval procedures applied by independent testing laboratories. Existing approvals of measuring instruments in Europe are based on the fulfilling of national or European standards or regulations. Measurement accuracy, resistance to environmental stress and other properties of the instrument are specified. How the measuring instrument was developed is not addressed in product standards.

The use of programmable electronics in measuring instruments has increased the complexity of the product. The instrument can make different types of measurements and perform a lot of functions that were impossible for an "unintelligent" instrument. It is no longer possible to claim that an instrument has been evaluated and tested in all possible configurations and applications. Confidence in a well-defined development process can then supplement a test result, which cannot reach 100% coverage.

European procedures for conformance testing give the possibility to rely on the manufacturers quality assurance system for CE marking in some cases. This puts emphasis on the development process.

2.4 Software Examination Level

The amount of work required at examination of software varies greatly. A company developing software-based measuring instruments may have a dedicated test department that may spend man-months to verify the design. An independent test laboratory might use only a few days to verify the essential requirements for a measuring instrument when the consequences of a fault are limited.

This report mainly concerns the efforts required at type approval testing by an independent test laboratory. There is a need to describe how thorough the testing and validation of software should be. WELMEC has defined three software examination levels: low, medium and high. The corresponding definitions are found in chapter 11 of the Software Requirements Guide. [MIDSwReq]

The definitions of the examination levels are:

Low: Standard type approval functional testing of the instrument is performed. No extra software testing is required.
Middle: In addition to the low level, the software is examined on the basis of its documentation. The documentation includes the description of the software functions, parameter description, etc. Practical tests of the software-supported functions (spot checks) may be carried out to check the plausibility of documentation and the effectiveness of protection measures.
High: In addition to the middle level, an in-depth test of the software is carried out, usually based on the source code.

2.5 Identification of software

Software can be described as a set of data, parameters and executable code. The executable code ("the programme") is the instructions for how the processing unit shall operate. The parameters are non-volatile data programmed into a specific measuring instrument (e.g. a metering constant).

All test objects including all pieces of software have to be unambiguously identified. The identification of software is especially important since it is practically impossible to identify software by visual inspection. Physical parts can be checked simply by looking at them and comparing them to the original test object. Software has to be identified by the specific issue (or version) of the software package.

Measuring instruments may include different types of software. A microprocessor based utility meter will probably contain an fixed programmed firmware i.e. a piece of software which is firmly programmed into the memory. The firmware may be the only software in the instrument, but there may also be downloadable replaceable pieces of software.

A PC based measuring instrument will be more complex. The PC itself has an internal BIOS, an operating system (e.g. Windows) and an application software developed by the manufacturer of the instrument.

An embedded system may be built on a real-time operating system that has been purchased from a software house by the meter manufacturer. The real-time operating system may exist in several versions, but only one version at a time can be expected to be included in the measuring instrument.

It is important to identify the software and the operating system of the measuring instrument. The identification should be made in a way that provides easy checking (verification) by non-software specialists.

Where threats to the integrity of the software exists, the identification shall also provide adequate level of protection against manipulation e.g. through a checksum of the legally relevant parts. The software protection level will be high if the risks are high. The use of a checksum may then not be enough. An alternative authentication method may be required, e.g. a hash algorithm.

2.6 Validation guidance

The test methods are explained and collected as "tool boxes" which can be used differently for different measuring instruments. The test methods can be combined into different validation plans for use with different measuring instruments. (See figure 2a.) The validation plans are referred to as “validation guidance” in the Software Requirements Guide [MIDSwReq].

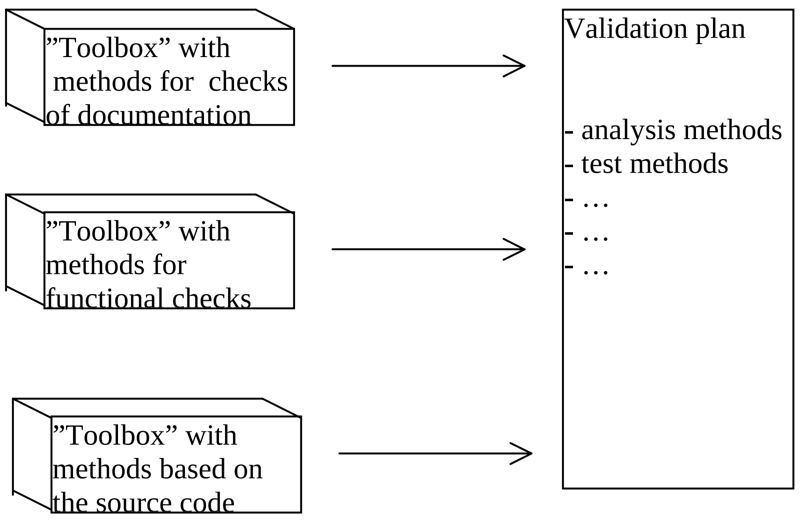

”Toolbox” with methods for checks of documentation
”Toolbox” with methods for functional checks
”Toolbox” with methods based on the source code
Validation plan
- analysis methods
- test methods
- …
- …
- …

Figure 2a. Combination of validation methods into a validation plan

The validation methods are expected to be adopted by testing laboratories to demonstrate compliance with the software requirements derived from the MID. The ambition is NOT to provide the complete set of test methods needed by the manufacturer at product development. Testing and validation at product development requires far more efforts than can be justified for conformity testing at an independent laboratory.

Different measuring instruments will need different validation plans depending on the risks associated with the instrument and the functions of the instrument. However, the type of meter (hot water, cold water, petrol, gas, electrical energy etc.) is not expected to influence the amount of validation needed as long as the risks are considered equal.

Two different sets of basic requirements are identified [MIDSwReq]:

- basic requirements for a built-for-purpose / dedicated device
  (e.g. a meter for electrical energy based on a microcontroller)
- basic requirements for systems with software on a computer
  (e.g. a PC based weighing instrument)

Four additional functions are identified [MIDSwReq] :

- instruments with long-term storage of measurement values
  (e.g. a gas meter)
- instruments with separation between software subject to legal control and software not subject to legal control
  (e.g. a PC based instrument)
- instruments with the ability to download software
  (e.g. a taximeter with the possibility to update software through the radio communication channel)
- instruments capable of transmitting measurement values via networks
  (e.g. a meter for electrical energy designed for remote reading by power line communication)

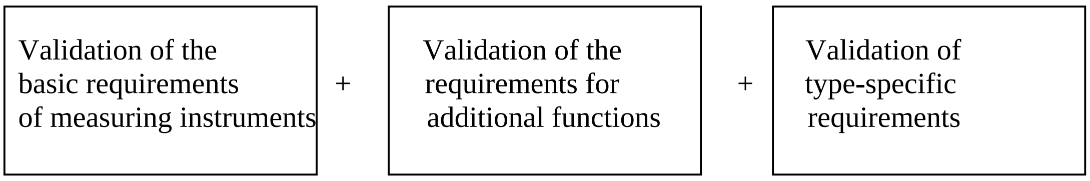
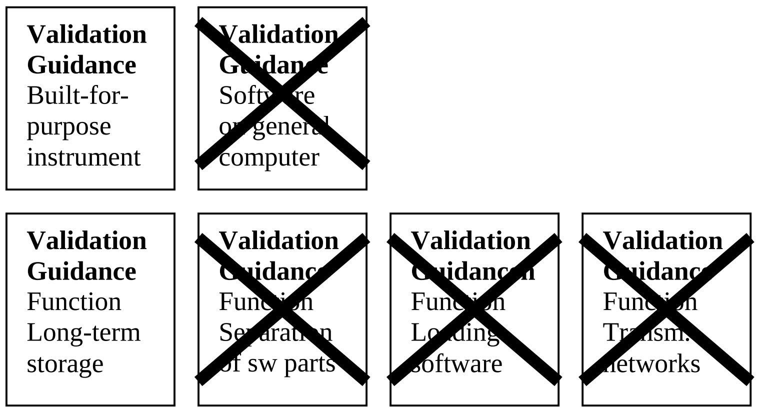

Validation of the basic requirements of measuring instruments
+
Validation of the requirements for additional functions
+
Validation of type-specific requirements

Figure 2b. Validation of basic requirements, additional functions and type-specific requirements

The validation activity can be broken down into three parts; basic requirements, additional functions and type-specific requirements (See figure 2b.) The same validation method can be used in more than one of the validation plans, e.g. functional tests can be used to validate all three kinds of requirements.

Example: A “simple” measuring instrument is designed as a stand-alone unit without networking capabilities. A display is provided to present the measured values and to indicate faults detected. No remote reading is possible, but an interface for downloading of measured values into a hand-held terminal exists. The instrument is used for continuous measurements and stores measured values on a long-term basis. (Reading of the stored values is expected at least once a year.)

The applicable requirements are the basic requirements for a built-for-purpose instrument, and the requirements for the function of long-term storage. The corresponding validation guidance of the guide shall be used.


Validation Guidance Built-for-purpose instrument
Validation Guidance Software on general computer
Validation Guidance Function Long-term storage
Validation Guidance Function Separation of sw parts
Validation Guidance Function Loading software
Validation Guidance Function Transm. networks

Validation methods can then be chosen from chapter 3 of this report. Suitable validation methods for “Built-for-purpose instrument” and “Long-term storage” can be:

- identification of software
- completeness of the documentation
- examination of the operating manual
- functional testing
- review of the software documentation

2.7 Documentation of examination and approval

A software validation should result in a test report. The test report shall describe the validation performed and the results achieved. A test report used as one of the documents supporting an EC type approval needs to connect firmly to the requirements of the MID.

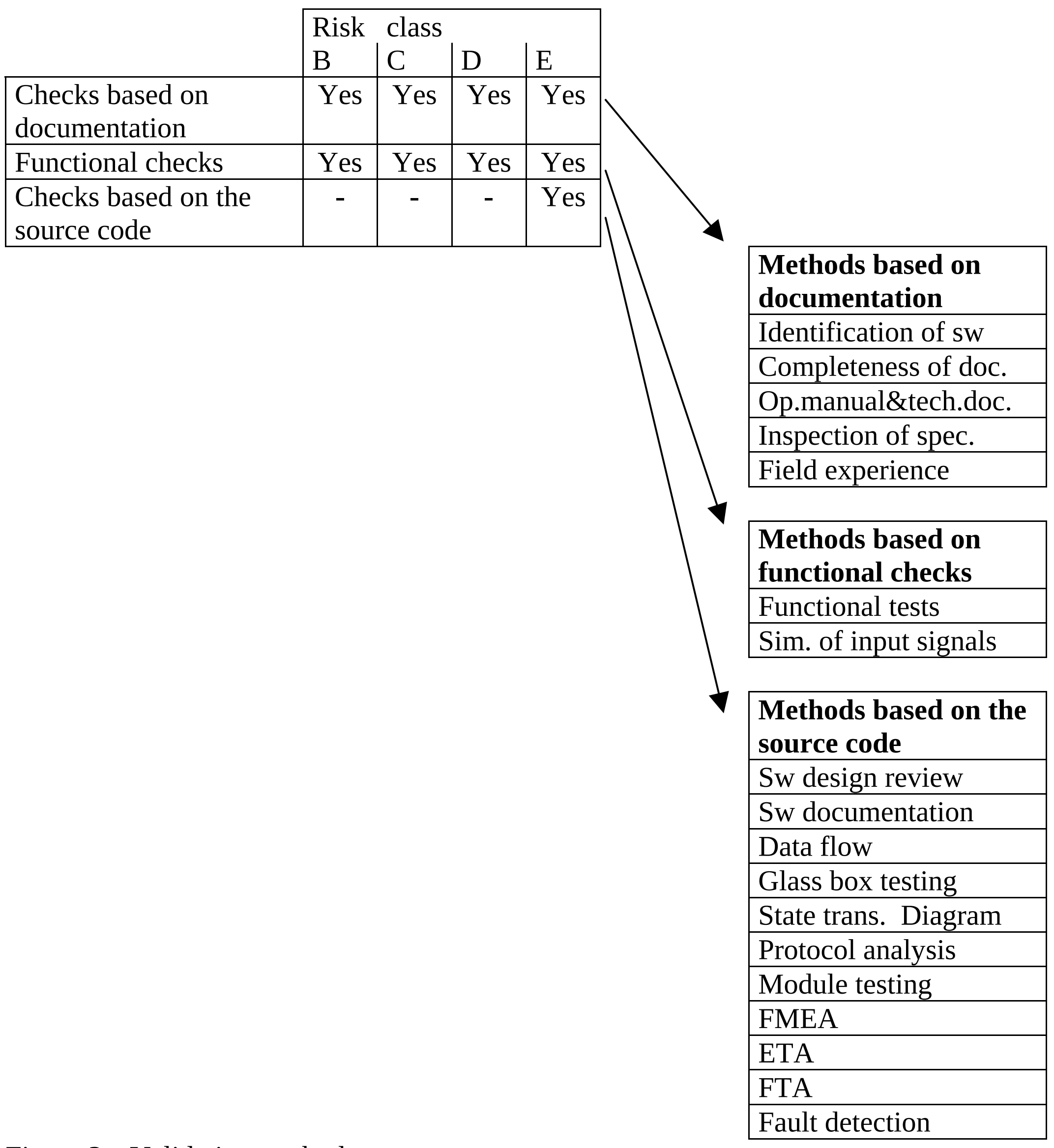
A software test report can present difficulties not experienced at other testing. The identification of the software under test must be made in a clear way. The validation methods applied must be described clearly enough to be understood by other test labs. The approval criteria to state if the requirements are fulfilled are often hard to specify, and may require interpretations.

It is nevertheless important that the test report is unambiguous and it must be possible to compare the result between different software testing laboratories.

3 Validation Methods

Validation of software can be performed in different ways. Analysis means examining the measuring instrument design without operating it. Testing means examining the instrument by observing its behaviour. The Software Requirements Guide [MIDSwReq] describes three types of validation; checks based on documentation, functional checks and checks based on the source code.

The extent of the validation required depends on the risks associated with the measuring instrument. Risk classes B, C, D and E are used. Risk class E is regarded for the highest risks in the MID. [MIDSwReq] Checks based on documentation and functional checks are used for all risk classes, while checks based on the source code are only mandatory for the higher risk classes. (See figure 3a.)



	Risk class			
	B	C	D	E
Checks based on documentation	Yes	Yes	Yes	Yes
Functional checks	Yes	Yes	Yes	Yes
Checks based on the source code	-	-	-	Yes

Methods based on documentation
Identification of sw
Completeness of doc.
Op.manual&tech.doc.
Inspection of spec.
Field experience

Methods based on functional checks
Functional tests
Sim. of input signals

Methods based on the source code
Sw design review
Sw documentation
Data flow
Glass box testing
State trans. Diagram
Protocol analysis
Module testing
FMEA
ETA
FTA
Fault detection

Figure 3a. Validation methods

The validation methods listed in this chapter are applicable in different situations. Some of the methods are fairly quick to use, while other methods require extensive effort. It should not be mistaken that all validation methods are to be used with all instruments. Validation guidance is given in the Software Requirements Guide [MIDSwReq].

Analysis based on documentation of the measuring instrument depends heavily on the quality of documents for design, installation, use and maintenance. It may be that further information is required in addition to the document. Functional checks or source code checking might then be performed to supplement the checking of documents.

There may also be situations where thorough functional checks are impossible. Checking of documentation or checking of source code might then be used as a supplement.
An efficient validation procedure often combines different kinds of validation methods.
The validation methods of this report are presented in three groups:
- validation methods for checks based on documentation
- validation methods for functional checks
- validation methods for checks based on source code

3.1 Methods based on checking of documentation

3.1.1 Identification of the software

Aim: To identify the software under examination.

Description: Information about all parts of the software included in the measuring instrument shall be collected. It will be necessary to understand the structure of the software system and to identify all pieces of software. The identification of the version of the software shall be noted for each piece of software. Following checklist may be used to support the work[1]:

	Identification of the Software					
		Ap.	Prot	Yes	No	Comment
A	Have all physical parts of the measuring instrument been identified?					
B	Have all software packages of the measuring instrument been found?					
C	Is the software divided into legally relevant and not legally relevant parts?					
D	Are operating systems used? (If yes, note name and version of the OS.)					
E	Is it possible to download new versions of the software?					
F	Is there a unique identification of the software version?					
G	Is the software version possible to read on the display of the device?					
H	Is the software version possible to read through a communication link?					
I	Is there a unique software identification on the physical hardware? (E.g. a label on a PROM)					
J	Has the version of the software development tools (compiler, linker etc.) been identified and documented?					
The evaluation of the software version gives the result ☐ Pass (The requirements according to the MID are fulfilled.) ☐ Fail (The requirements according to the MID are **not** fulfilled.)						

Abbreviations: Ap.= applicable, Prot. = Test protocol from manufacturer exists.
The answer Yes/No states the answer to the question. The intention is not to state whether a requirement is fulfilled or not.
The Pass/Fail judgement is made based on a number of Yes/No questions.

Applicable at software examination level : low, middle or high. (This validation method has to be supplemented by other methods.)

Reference: ISO/IEC 17025 clause 5.8.2.
Software Requirements Guide [MIDSwReq], requirements P2 and U2.

[1] This checklist is (all following checklists are) exclusively a supporting mean for the application of this validation method (the corresponding validation methods). They must not be mistakenly understood as an alternative to the checklists of chapters 13 and 14 of [MIDSwReq].

3.1.2 Completeness of the documentation

Aim: To check the completeness of the documentation describing the measuring instrument and its software.

Description: The documentation describing the instrument is checked for completeness. Following checklist may be used to support the work:

	Completeness of the documentation					
		Ap.	Prot	Yes	No	Comment
A	Is there an operating manual?					
B	Are there manuals for advanced users, e.g. installation manuals, service manuals and manuals describing communication?					
C	Are there design descriptions for the hardware and the software of the instrument?					
D	Has the functionality of the instrument been documented?					
E	Have the legally relevant parts of the instrument been suitably documented?					
F	Has the manufacturer declared that the software controlling the instrument to be type approved fully complies with the documentation?					
G	Has the manufacturer declared that there are no functions other than the documented ones?					
H	Is there a description of all legally relevant functions, legally relevant parameters, switches and keys?					
I	Is there a description of the accuracy of the measuring algorithms?					
J	Is there a description of the menus and dialogues?					
K	Is there an overview of the security aspects of the operating system, e.g. protection, user accounts, privileges etc.					
The evaluation of the software version gives the result ☐ Pass (The requirements according to the MID are fulfilled.) ☐ Fail (The requirements according to the MID are **not** fulfilled.)						

Abbreviations: Ap.= applicable, Prot. = Test protocol from manufacturer exists.
The answer Yes/No states the answer to the question. The intention is not to state whether a requirement is fulfilled or not.
The Pass/Fail judgement is made based on a number of Yes/No questions.

The measuring instrument and its use is described in several documents. Examples of such documents are:

- the operating manual describing the functionality of the instrument
- the service manual describing how to maintain the instrument
- a design description of the hardware of the measuring instrument
- a design description of the software of the measuring instrument
- the design description of the measuring system describing the system where the instrument will be used

written declarations by the manufacturer concerning the legally relevant function

Applicable at software examination level : low, middle or high. (This validation method has to be supplemented by other methods.)

Reference:
Software Requirements Guide [MIDSwReq], requirements P1 and U1.
WELMEC guide 7.1, clauses 4.2, 6.1.4 (ER5.1) and 6.2.4 (ER5.1)

3.1.3 Examination of the operating manual and technical documentation

Aim: To reveal faults in the legally relevant functions described in the operators manual and in the technical documentation.

Description: In addition to the normal type examination tests the software is examined on the basis of a description of the software functions. All functions described in the documentation should be identified and understood.

The software implementation of the functions of a measuring instrument is expected to be documented. The documentation can be in plain text, but also graphical representation such as flow charts or UML charts can be used. The aim of the author should be to make an overview of the software and to explain the overall logic in other representation than source code.

The examiner uses the documents to check if the requirements have been fulfilled. Following checklist may be used to support the work:

	Examination of the operating manual and the technical documentation	Ap.	Prot	Yes	No	Comment
A	Have all the legally relevant functions been described in the operating manual?					
B	Have all the legally relevant functions been described in the technical documentation of the functionality?					
C	Is there a detailed description of all legally relevant software functions?					
D	Is there a detailed description of all legally relevant parameters that determine the functionality of the instrument?					
E	Is there a description of the measuring algorithms?					
F	Is there a description of the menus and the dialogues?					
G	Is there a reference to the requirements listed in WELMEC guide 7.1?					
The evaluation of the software version gives the result ☐ Pass (The requirements according to the MID are fulfilled.) ☐ Fail (The requirements according to the MID are **not** fulfilled.)						

Abbreviations: Ap.= applicable, Prot. = Test protocol from manufacturer exists.
The answer Yes/No states the answer to the question. The intention is not to state whether a requirement is fulfilled or not.
The Pass/Fail judgement is made based on a number of Yes/No questions.

Applicable at software examination level : low, middle or high. (This validation method has to be supplemented by other methods to reach software examination level middle or high.)

Reference: WELMEC guide 7.1, clauses 4.2, 6.1.4 (ER5.1) and 6.2.4 (ER5.1)

3.1.4 Inspection of the specification

Aim: To examine the design objectives for the legally relevant functions, and to make sure that all relevant requirements have been considered.

Description: Inspection is a general technique in which various qualities of a specification document are assessed by an independent team. The inspection team puts questions to the creator, who must answer them satisfactorily. The examination should (if possible) be carried out by a team that was not involved in the creation of the specification. The required degree of independence is determined by the software examination level demanded of the system. The independent inspectors should be able to reconstruct the operational function of the system in an indisputable manner without referring to any further specifications. They must also check that all relevant technical aspects in the operational and organisational measures are covered. This procedure has proved itself to be very effective in practice.

Applicable at software examination level: low, middle or high

Reference: IEC 61508-7, clause B.2.6

Example:
The security aspects of the operating system have been specified and a risk analysis of the threats to information security has been made. The document is inspected to find

- security threats
- aspects of availability of the software platform
- aspects of integrity of the measured values stored under the operating system
- aspects of confidentiality of the measured values
- security mechanisms implemented

3.1.5 Field experience

Aim: To use field experience from different applications as one of the methods to validate the legally relevant functions of the measuring instrument.

Description: Use of components or subsystems, which have been shown by experience to have no, or only unimportant, faults when used, essentially unchanged, over a sufficient period of time in numerous different applications. Particularly for complex components with a multitude of possible functions, the developer shall pay attention to which functions have actually been tested by the field experience. For example, consider self-test routines for fault detection: if no break-down of the hardware occurs within the operating period, the routines cannot be said to have been tested, since they have never performed their fault detection function.

For field experience to apply, the following requirements must have been fulfilled:
– unchanged specification;
– 10 systems in different applications;
– 10^5 operating hours and at least one year of service history.

The field experience is demonstrated through documentation of the vendor and/or operating company. This documentation must contain at least
– the exact designation of the system and its component, including version control for hardware;
– the users and time of application;
– the operating hours;
– the procedures for the selection of the systems and applications procured to the proof;
– the procedures for fault detection and fault registration as well as fault removal.

Applicable at software examination level: low, middle or high

Reference: IEC 61508-7, clause B.5.4

3.2 Methods based on functional checks

3.2.1 Functional testing

Aim: To reveal faults in the legally relevant functions of the measuring instrument.

Description: During the functional tests, reviews are carried out to see whether the specified characteristics of the system have been achieved. The system is given input data which adequately characterises the normally expected operation. The outputs are observed and their response is compared with that given by the specification. Deviations from the specification and indications of an incomplete specification are documented.

The functional testing examines functions of a complete device intended to be used in a measuring system.

The functions of a system or program are executed in a specified environment with specified test data that is derived systematically from the specification according to established criteria. This exposes the behaviour of the system and permits a comparison with the specification. No knowledge of the internal structure of the system is used to guide the testing. The aim is to determine whether the functional unit carries performs according to the essential requirements of the MID [MID]. The technique of forming equivalence classes is an example of the criteria for blackbox test data. The input data space is subdivided into specific input value ranges (equivalence classes) with the aid of the specification. Test cases are then formed from the
– data from permissible ranges;
– data from inadmissible ranges;
– data from the range limits;
– extreme values;
– and combinations of the above classes.

All legally relevant functions shall be tested.

Definition-based or specification-based testing is also known as functional testing or "black-box" testing. It identifies test cases based on the definition of what the software product is intended to do. These test cases challenge the intended use or functionality of a program, and the program's internal and external interfaces. [FDA clause 5.2.5]

Applicable at software examination level: low, middle or high

Reference: IEC 61508-7, clauses B.5.1 and B.5.2

3.2.2 Simulation of input signals

Aim: To reveal faults in the legally relevant functions of the measuring instrument by functional testing.

Description: A measuring instrument cannot always be tested in the environment it is going to be used. Some input signals may only be available at a real installation. This will require the facilities to simulate certain input signals.

Functional tests can be carried out according to the description in clause 3.1.4, but one or more of the real input signals can be simulated. Simulation may typically be realised with pulse generators, voltage sources, current sources or resistors. Simulation of data communication may require the use of a personal computer executing a software package developed to generate messages to the measuring instrument.

Applicable at software examination level: medium or high

Reference: Software Requirements Guide [MIDSwReq], requirements P3, P4, U3, U4 etc.

Example:

A meter for electrical energy is equipped with digital inputs to control the active tariff, an interface for power line communication, an interface for optical communication and manual push buttons on the front on the meter (used to scroll the display).

The operation of the meter is checked at full communication load (both power line communication and optical communication running at high speed), random manual activation of the push buttons and automatic signals requesting change of tariff. The accuracy of the measurements must be kept also at high input signal load.

3.3 Methods based on checking of the source code

The methods for Failure Mode and Effect Analysis, Event Tree Analysis and Fault Tee analysis are listed as methods based on checking of the source code. However, they can be used also in situations when the source code is not available. The analysis of the behaviour of the measuring instrument will then be based on documentation and/or functional checking.

The methods for examination of fault detection might by possible to use also without access to the source code.

3.3.1 Software design review

Aim: To analyse the software of the measuring instrument by having it presented by and/or discussing it with the development engineers.

Description: Design reviews are documented, comprehensive, and systematic examinations of a design to evaluate the adequacy of the design requirements, to evaluate the capacity of the design to meet these requirements and to identify problems. [FDA clause 3.5]

A software design review can be held on the software architecture of the instrument and the basic legally relevant functions, or on added functions (e.g. transmission of measured values over networks).

The software examined in a meeting between the author and the examiner. Specified legally relevant functions of the measuring system are examined and evaluated to ensure that the measuring instrument conforms to the requirements given in the specification. Any points of doubt concerning the implementation and use of the product are documented so they may be resolved. In contrast to a walk-through, the author is passive and the inspector is active during the inspection procedure.

A design review may also be called inspection.

The review is often called a walk-through when it is based on a presentation by the development engineers. The software examined in a meeting between the manufacturer and the examiner. An engineer with good knowledge of the software shall represent the manufacturer. In contrast to a design review (i.e. an inspection), the author is active and the inspector is passive during the walk-through. A code walk-through consists of a walk-through team selecting a small set of paper test cases, representative sets of inputs and corresponding expected outputs for the program. The test data is then manually traced through the logic of the program.

Applicable at software examination level: high

Reference: IEC 61508-7, clauses B.3.7, C.5.16

3.3.2 Review of software documentation

Aim: To analyse the software of the measuring instrument by reviewing the documentation of how a function is implemented in software.

Description: The software implementation of the functions of a measuring instrument is expected to be documented. The documentation can be in plain text, but also graphical representation such as flow charts or UML charts can be used.

The examiner reviews the documents by himself without the support of the development engineers. The purpose is to get detailed knowledge of a legally relevant function. Often the review of documents has to be supplemented by reviews of parts of the source code.

Applicable at software examination level: high

Reference: None

3.3.3 Data flow analysis

Aim: To detect poor and potentially incorrect program structures.

Description: Data flow analysis is a static testing technique that combines the information obtained from the control flow analysis with information about which variables are read or written in different portions of code. The analysis can check for
– variables that may be read before they are assigned a value – this can be avoided by always assigning a value when declaring a new variable;
– variables that are written more than once without being read – this could indicate omitted code;
– variables that are written but never read – this could indicate redundant code.

A data flow anomaly will not always directly correspond to a program fault, but if anomalies are avoided the code is less likely to contain faults.

Data flow diagrams document how data input is transformed to output, with each stage in the diagram representing a distinct transformation.

Data flow diagrams are made up of three components:
– annotated arrows – represent data flow in and out of the transformation centres, with the annotations documenting what the data is;
– annotated bubbles – represent transformation centres, with the annotation documenting the transformation;
– operators (and, xor) – these operators are used to link the annotated arrows.

Each bubble in a data flow diagram can be considered as a stand-alone black box which, as soon as its inputs are available, transforms them to its outputs. One of the principal advantages is that they show transformations without making any assumptions about how these transformations are implemented. A pure data flow diagram does not include control information or sequencing information, but this is catered for by real-time extensions to the notation.

The preparation of data flow diagrams is best approached by considering system inputs and working towards system outputs. Each bubble must represent a distinct transformation – its output should, in some way, be different from its input. There are no rules for determining the overall structure of the diagram and constructing a data flow diagram is one of the creative aspects of system design. Like all design, it is an iterative procedure with early attempts refined in stages to produce the final diagram.

Applicable at software examination level: high

Reference: IEC 61508-7, clauses C.2.2 and C.5.10

3.3.4 White-box testing

Aim: To reveal faults in the software-based functions by functional testing.

Description: Glass-box testing means that the operation of the measuring instrument is tested using knowledge on how the software is designed. Test cases for the functional testing will be selected based on knowledge of the source code.

Code-based testing is also known as structural testing, “glass-box” testing or "white-box" testing. It identifies test cases based on knowledge obtained from the source code, detailed design speciation, and other development documents. These test cases challenge the control decisions made by the program; and the program's data structures including configuration tables. [FDA clause 5.2.5]

Glass-box testing can also mean testing of individual software modules in an artificial environment. Such glass-box testing is not the same as functional glass-box testing intended.

Applicable at software examination level : high

Reference: None

3.3.5 State transition diagram

Aim: To examine the operation of the legally relevant function of a measuring instrument.

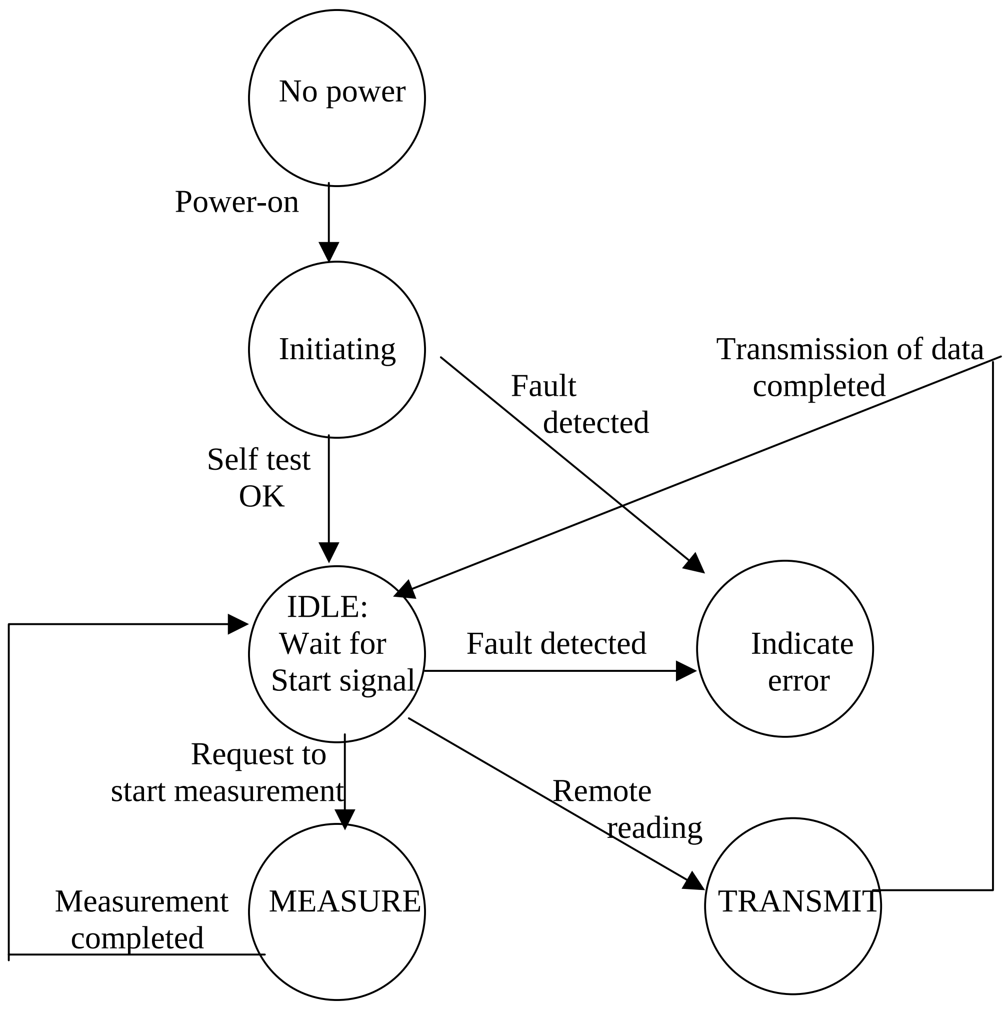
Description: A measuring instrument can be described as being in different states; power-off, fully operational, stand-by, faulty etc. The different states and the transition between states is software controlled.

A state transition diagram (state chart) describes the different states and the conditions for transition between states. The states are best derived from the specification or the functional description of the instrument. The conditions for transition must sometimes be found in the source code.

Applicable at software examination level : high

Reference: None

Example: A simple example of the power-on procedure of a measuring instrument is given. The intention is to illustrate the diagram, not to give requirements for a correct function.



3.3.6 Protocol analysis

Aim: To examine the communication protocol used by the measuring instrument for legally relevant functions, such as software download or transmission of measured values.

Description: The communication protocols used for communication with a measuring instrument will include techniques and measures to make the data transfer as secure and reliable as possible. Transmission of measured values over networks and download of software will be affected by the qualities of the communication protocol.

The protocol can be analysed by examining the documentation.

Applicable at software examination level : high

Reference: None

3.3.7 Testing of software modules

Aim: To examine the functionality of individual software modules by testing in a development environment.

Description: A module in the source code is identified as important for a legally relevant function. This software module can then be tested as a stand-alone module in a software development environment. Access to software development tools will be required.

The module can be tested either as a “black box” without considering the logic of the source code. Test cases are selected based on the input variables to the module. The result in the output modules is observed. The module can also be tested as a “glass box” when the test cases are selected with knowledge of the source code. The test cases can be selected to cover different parts of the code, different algorithms, different data structures or different sequences.

The test is best automated using a “script” (“batch file” or “command file”) which runs all the test cases automatically. The output values are automatically registered for each test run.

Testing of software modules is suitable to find faults in logic. It is less suitable to find timing problems since the tests are made in an artificial environment.

Applicable at software examination level : high

Reference: None

3.3.8 Failure Mode and Effects Analysis (FMEA)

Aim: To examine how faults lead influence the legally relevant function of the measuring instrument. (N.B. Fault tolerance is outside the scope of the MID requirements for measuring instruments.)

Description: Each component of a system is analysed in turn, and a number of fault modes are assumed. The effects of the component faults on the measuring instrument are analysed.

The FMEA can be carried out on hardware components. For measuring systems, the components can be expected to be measuring instruments or subsystems used in the complete system. An FMEA can also be used to analyse faults in communication, e.g. download of software or remote reading of measures values.

Applicable at software examination level: high

Reference: IEC 61508-7, clauses B.6.6.1 and B.6.6.4
IEC 60812

Example: A simple example of how faults are handled in a taximeter measuring system is given. The intention is not to give an exhaustive description of the functions of a taximeter, but to illustrate the principles of an FMEA. The intention is neither to describe the desired functionality.

FMEA Taximeter			
Component	**Fault mode**	**Fault effect**	**System effect**
Printer	No supply voltage	No receipt printed	Meter in operation
"	No connection	No receipt printed	Meter in operation
"	Out of paper	No receipt printed	Meter in operation
….	….	….	….
Central control unit	No supply voltage	All functions stop	Meter stops
"	Memory fault detect	Fault indicated	Meter in operation
….	….	….	….
Display	No supply voltage	No display	Meter stops
"	No connection	No display	Meter stops
"	LCD hardware fault	No display	Meter in operation
….	….	….	….
Velocity transducer	No supply voltage	No velocity signal	Meter stops
"	No connection	No velocity signal	Meter stops
"	Velocity > 300 km/h	Absurd velocity	Meter in operation
….	….	….	….

3.3.9 Event tree analysis

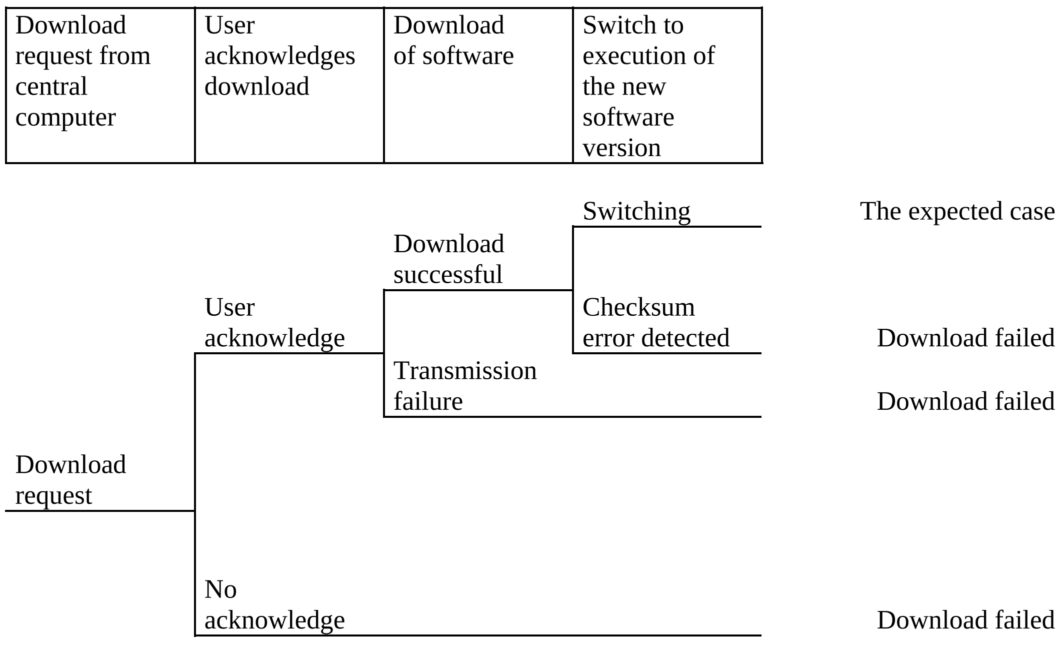
Aim: To examine how a sequence of events can develop in a measuring system, and if failures of the legally relevant functions can be caused.

Description: On the top of the diagram is written the sequence conditions that are relevant in the progression of events that follow the initiating event. Starting under the initiating event, which is the target of the analysis, a line is drawn to the first condition in the sequence. There the diagram branches of into "yes" or "no" branches, describing how future events depend on the condition. For each of these branches one continues to the next condition in a similar way. Not all conditions are, however, relevant for all branches. One continues to the end of the sequence, and each branch of the tree constructed in this way represents a possible consequence.

Applicable at software examination level: high

Reference: IEC 61508-7, clause B.6.6.3

Example: A simple example of how software download can be examined is given. The intention is not to illustrate the proper download procedure, but to illustrate the principles of an event tree.



3.3.10 Fault Tree Analysis

Aim: To examine how faults influence the legally relevant function of the measuring instrument. (N.B. Fault tolerance is outside the scope of the MID requirements for measuring instruments.)

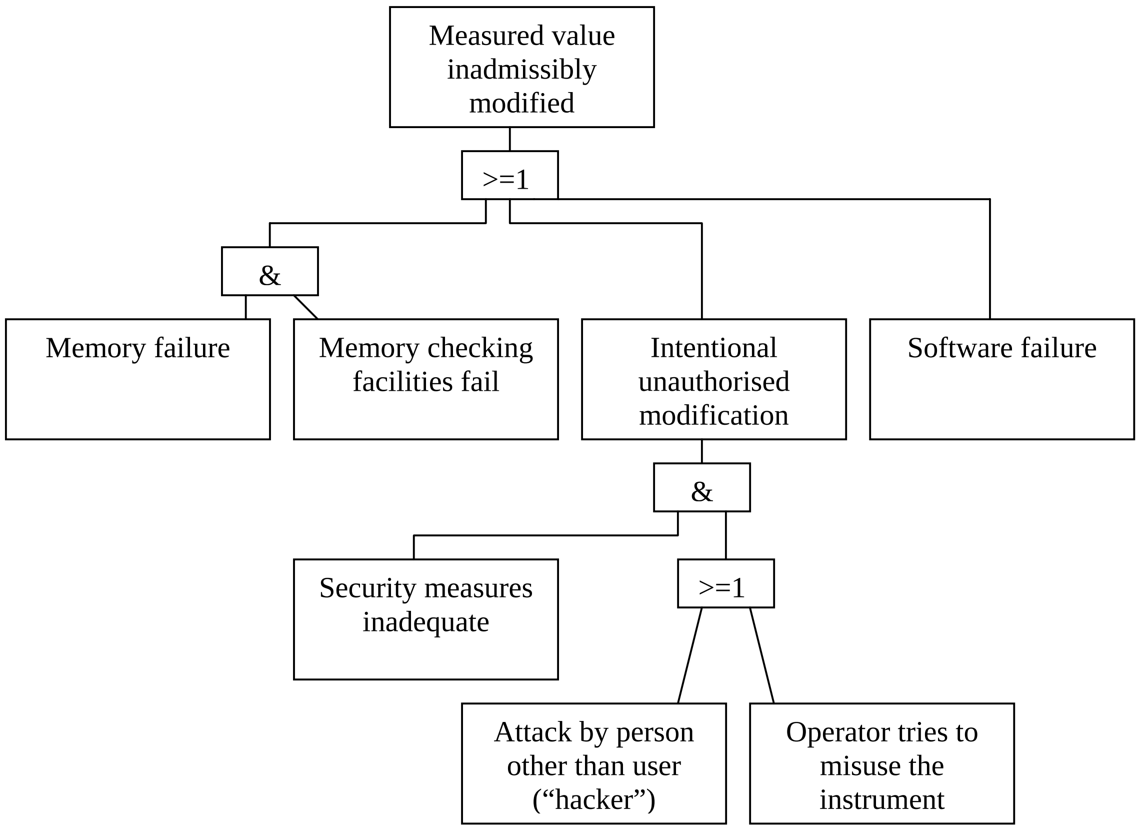
Description: Starting at an event that would be the immediate cause of a failure of a legally relevant function, analysis is carried out along a tree path. Combinations of causes are described with logical operators (AND, OR). Intermediate causes are analysed in the same way, and so on, back to basic events where analysis stops.

The method is graphical and a set of standardised symbols is used to draw the fault tree. The resulting fault tree is often easily explained and can be used as a common base for discussion between different partners.

Applicable at software examination level : high

Reference: IEC 61508-7, clause B.6.6.5
IEC 61025

Example: A simple example of a fault tree is given. Inadmissible modification of measured values is chosen as top event. The fault tree can be used to indicate possible weaknesses in the design.


Measured value inadmissibly modified
>=1
&
Memory failure
Memory checking facilities fail
Intentional unauthorised modification
Software failure
&
Security measures inadequate
>=1
Attack by person other than user ("hacker")
Operator tries to misuse the instrument

3.3.11 Analysis of Fault detection: Programme sequence monitoring

Aim: To examine the programme sequence monitoring (watchdog).

Description: Techniques and measures may be designed into the measuring instrument to detect if normal processing is disturbed. The “watchdog” may be based on hardware circuits or on software solutions. Following checklist may be used as a support in the examination.

	Programme sequence monitoring					
		Ap.	Prot.	Yes	No	Comment
A	Is there a hardware device (watchdog) to monitor the execution of the software?					
B	Is there a software monitoring of the execution of the software?					
C	Will a watchdog alarm result in a reset of the microprocessor?					
D	Will a watchdog alarm result in the loss of the latest measured values?					
E	Is the watchdog triggered at a period less than 5 seconds?					
F	Is the time base of the watchdog independent from the processor time base?					
G	Is the watchdog a separate hardware circuit (not integrated into the micro controller) ?					
H	Is an unintended change in the program flow likely to lead to a watchdog alarm?					
I	Is the watchdog triggered during the execution of the main program?					
J	Has the software implementing the sequence monitoring been checked without finding logical faults?					
The evaluation of the software version gives the result ☐ Pass (The requirements according to the MID are fulfilled.) ☐ Fail (The requirements according to the MID are **not** fulfilled.)						

Applicable at software examination level : high

Reference: IEC 61508-2, table A.10.

Annex A References

[FDA]
General Principles of Software Validation; Final Guidance for Industry and FDA Staff"
US Food and Drug Administration
Document issued on January 11, 2002.
Download from www.fda.gov/cdrh/comp/guidance/938.pdf

[IEC60812]
International standard IEC 60812
"Analysis techniques for system reliability –
Procedure for failure mode and effects analysis"
Possible to order at www.iec.ch

[IEC61025]
International standard IEC 61025
"Fault tree analysis"
Possible to order at www.iec.ch

[IEC61508]
International standard IEC 61508, part 1- 7
"Functional safety of electrical/electronic/programmable electronic safety-related systems"
Possible to order at www.iec.ch

[ISO17025]
International standard ISO/IEC 17025
"General requirements for the competence of testing and calibration laboratories"
Possible to order at www.iec.ch

[MID]
DIRECTIVE 2004/22/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 31 March 2004 on measuring instruments
("The Measuring Instruments Directive")
Download from
http://europa.eu.int/eur-lex/pri/en/oj/dat/2004/l_135/l_13520040430en00010080.pdf

[MIDSwReq]
MID-Software Software Requirements Guide
Draft version 0.055, 31 August 2004
Download from www.mid-software.org

[WELMEC2.3]
WELMEC Guide 2.3
"Guide for Examining Software (Non-automatic Weighing Instruments)"
Juni 2002
Download from www.welmec.org

[WELMEC7.1]
WELMEC Guide 7.1
"Software Requirements on the Basis of the Measuring Instruments Directive"
Issue 1, October 1999.
Download from www.welmec.org

[WELMEC7draft] WELMEC draft Guide 7.2
"Software Requirements for Measuring Instruments Used for Supply to the Customer by Mains"
Draft version 0.10, 2001-08-31