Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Standards & Interfaces 28 (2006) 286–296

COMPUTER STANDARDS & INTERFACES

www.elsevier.com/locate/csi

Software testing and preventive quality assurance for metrology

Norbert Greif

Physikalisch-Technische Bundesanstalt, Abbestr. 2-12, 10587 Berlin, Germany

Available online 26 September 2005

Abstract

Software controlled measuring systems can be approved with the help of different kinds of conformity assessment techniques based either on the final product, the product design or corresponding development and production processes. To validate the software as an integral part of the measuring system, different approaches of software quality assurance have to be applied for different conformity assessment procedures. There are two essential categories of software quality assurance, both of which supplement each other. On the one hand side, analytical methods of software testing, static analysis, and code inspection are used in the scope of conformity assessments of final or intermediate products. On the other hand, preventive audits of software development processes are applied to evaluate and improve appropriate software processes and to consequently support process related conformity assessment procedures. Depending on the validation objectives, validation methods, audit areas, and the appropriate requirements have to be selected and refined. A major problem of validation efforts, namely the process of defining and refining testable requirements, can be solved with the help of international software standards. In Germany, the accredited software testing laboratory at PTB supports software quality in metrology.

Keywords: Software validation; Software testing; Software quality assurance; Requirements; Metrological software

1. Introduction

By integrating software into measuring systems, the functionality and usability of such systems is improved. This is not least due to the fact that significant tasks in the measurement process such as

- the control of the entire measurement process,
- the acquisition, processing and storage of the measurement data, and
- the presentation of the measurement results

are now performed using software. By no means, however, should we ignore that the use of computer software always involves risks and that it can be of vital importance for our health – and even our lives – that computer programs function correctly. For two reasons, the use of software-controlled systems will always remain a factor of risk: on the one hand, implementation errors can never be completely ruled out, and on the other hand, software solutions will always provide a target for intentional or unintentional manipulation.

It thus is evident that software quality assurance must play an important role in metrology [1,2]. From the various areas of metrology which are very sensitive to software questions, only two will

E-mail address: norbert.greif@ptb.de

0920-5489/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2005.07.007

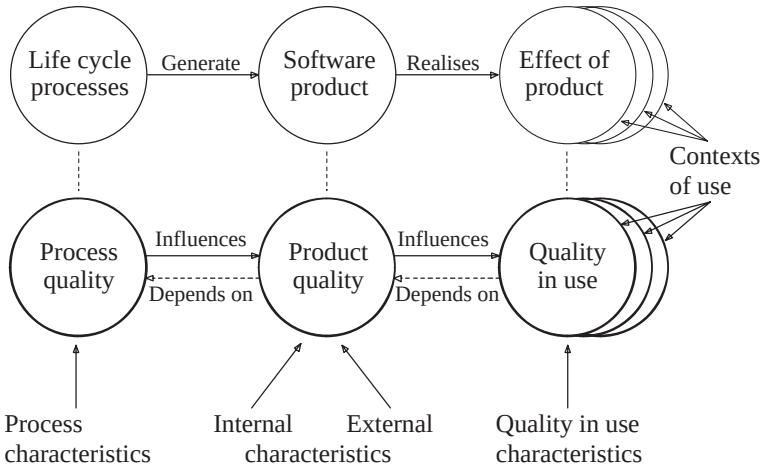

Fig. 1. Product and process oriented aspects of software quality.

be picked out here as examples: on the one hand medical measuring techniques and, on the other hand, the area of legal metrology which we owe to consumer protection. Be it electronic weighing instruments, speedometers, fuel dispensers at petrol stations, exhaust meters or medical measuring devices — in all cases, consumers must be able to trust the measurement results calculated by the computer. To maintain consumer confidence in the correct and reliable functioning of the software in measuring systems, the existing risk has to be minimised. Today this is mainly achieved by different kinds of conformity assessment techniques:

1.1. Product testing

The final software product (or an intermediate product) is thoroughly tested by an independent test centre. In analogy to the general inspection carried out for motor vehicles, the independent body certifies – if the test result is positive – that a certain software product complies with the latest state of the art as regards its quality and that it is suited for its intended use.

1.2. Process assessment and improvement

The software development process is regularly accompanied by preventive quality assurance measures (audits) already at the producer's. If the result of the audits is positive, the independent auditor certifies that the producer's software development process, including all its sub-processes, complies with the latest state of the art. If the outcome of the audit is negative, suitable suggestions for improvement are derived from the audit results.

1.3. Producer's declaration

The software producer himself declares compliance of his software products or his software development processes with the technical and quality standards applicable.

For the implementation of the two first points, and to verify producer's declarations, a software test centre has been established at the PTB. It offers its services to both internal and external customers and was accredited as a test centre according to ISO/IEC 17025 in February 2001 (see Section 6).

2. Product testing versus preventive quality assurance?

Software quality assurance consists of (a) the analytical testing of software products and (b) the assessment and improvement of the software development processes as preventive action to be taken as early as possible in the development phase. These two components are inseparable, and they supplement each other. The quality of software is

determined by the quality of the final software product and its intermediate products, and is also a result of the quality of the underlying processes of the software life cycle, especially of the development processes (see Fig. 1). Not only is the quality of software evaluated by its internal characteristics (e.g. the results of static analyses), its external characteristics (e.g. the results of dynamic function tests) and its special quality in use characteristics (see also Section 4), but it is directly influenced by the quality of the different processes contributing to its development. For the assessment and permanent improvement of development processes by means of special process characteristics, models such as ISO/IEC 15504 (SPICE) or CMMI are available and already in use worldwide (see Section 5). In the long run, high software quality can be achieved only if the underlying software development processes are of high quality, too. The logical consequence from this is that analytical and preventive aspects of quality assurance have to be linked and directed towards the common objective of software quality.

The relationship between testing and preventive quality assurance also becomes clear when we look at the application of quality requirements (see Section 3). The requirements which will finally be applied by the test engineer as quality criteria for the software test must be defined as quality objectives already in the design and development phases as it is not possible to "test quality into a product" once it is completed. The requirements applying to a software product and its development processes should be known not only to the test engineer but also to the device manufacturer and to the software producer. Efforts should therefore be made to provide them with the necessary information, ideally in the form of checklists, guidelines and process models. If these guidelines are applied at an early stage already, this will have a positive impact both on the manufacturing process and on the quality of the software development process, and finally lead to an improvement of the software product itself. The aim of this approach is to strengthen customer confidence in the producer, to enhance product testability, to facilitate explicit software testing and render it more cost-effective, and to replace final product testing in part or completely by analysing and assessing the software development process right from the beginning by audits at the producer's.

3. Definition of testable requirements

It is a precondition for assessing the quality of software used in measuring systems that testable requirements are defined for the software products and their development processes [1–3]. These requirements, which must cover both, metrological and software-related aspects, will serve the software producer as target functions in the development of his product and the test engineer or software user as quality criteria. Unfortunately, at the beginning of a test cycle, testable requirements are often rare and the producer's ideas about the objective of the test usually rather vague. All he generally wants is his product to be certified as being "of high quality," or as "functioning correctly." For the test engineer, however, these formulations are much too imprecise. He needs detailed test specifications and defined targets. Each time a product is tested, concrete test objectives are therefore determined beforehand in close cooperation with the producer, and a catalogue of specific technical requirements is compiled.

For the definition of requirements, existing standards and guidelines can be used as a basis. It is, however, to be noted that the state of standardisation in the field of software quality assessment is not very clear.

A survey of the situation is given in [4] with particular reference to standards and regulations containing special software requirements for testing and calibration laboratories [6,7].

As an example, Fig. 2 shows the quality characteristics for software products according to ISO/IEC 9126-1 [8].

For the specific problems encountered in metrology, most of the software requirements specified in standards and guidelines are formulated in a much too general way. Before they can be applied to metrological software, they have to be tailored for their intended use. Only in a few regulations, the special needs of metrology are taken into account, e.g. in the GLP Principles of the OECD [6] and in a previous draft computer guideline of the EA (European co-operation for Accreditation) [7]. Unlike general standards such as ISO/IEC 9126 [8], the GLP and EA guidelines also account for the special conditions in testing and calibration laboratories. In

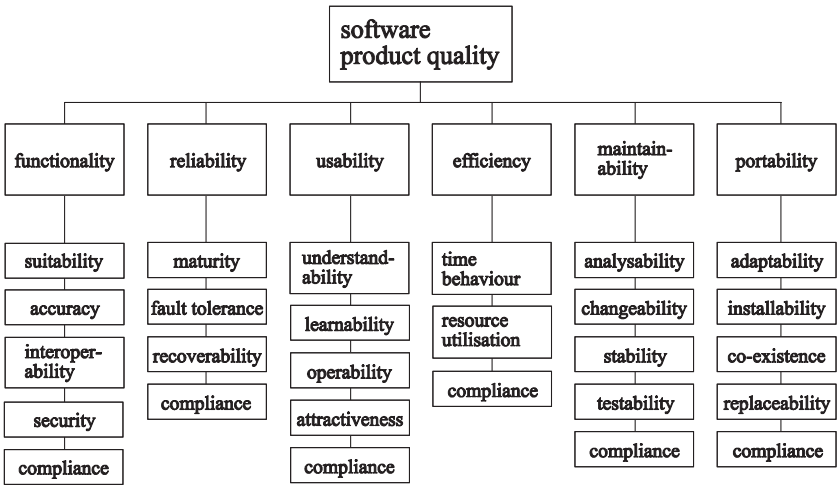

Fig. 2. Software quality characteristics according to ISO/IEC 9126.

certain cases, these corpora of regulations can therefore be directly used for the testing of measuring software. Special features of legal metrology are considered in the WELMEC-Guide 7.2 [18]. Based on the MID [17], concrete requirements and validation guidance are provided for measuring systems subject to legal verification.

Usually, however, the requirements must be separately drawn up and defined for the different classes of measuring instruments based on the standards and guidelines referred to in [4]. The time and work spent on drawing up the requirements catalogues takes up a large amount of the total test work. To reduce this amount, the PTB Software Test Centre has prepared a series of catalogues covering different types of requirements and allowing a software test to be carried out from different aspects. The following catalogues are available:

- General non-functional software requirements;
- Requirements for software used as testing equipment;
- Ergonomic software requirements;
- Requirements for the use of software-controlled systems in testing and calibration laboratories;
- Checklist for auditing a software producer.

The catalogues are permanently updated. Also, new catalogues for further aspects are under preparation. Detailed requirements lists of the various catalogues have been compiled in [2].

From the requirements catalogues and the applicable standards and guidelines, concrete requirements for special device classes can be derived. For a weighing instrument, for example, the requirement "functionality" means that the mass of a weight must be correctly indicated (within the limits of the declared measurement uncertainty). For the software of this weighing instrument, this implies that, e.g., the zero position of the instrument (the offset of the weighed value) is determined in a self-check and stored in a suitably dimensioned program variable.

For example, the Software Test Centre of the PTB has worked out catalogues of concrete requirements for the following fields of application: Calculation of measurement uncertainties, calibration of gauge blocks (length measuring technique) [3], calculation of the radiation dose for flight attendants, electronic voting machines. One objective of the future work will be the definition of specific metrological quality criteria for further classes of test applications. Such criteria would allow concrete requirements for different types of software tests and for different instrument classes to be derived more easily. It is also to be noted here that reasonable software requirements are also needed for preventive quality assurance and for the development or purchase of measurement software.

4. Software product testing

The objective of a software product test can strongly vary from case to case, with the test method varying accordingly. This involves that different methods and procedures are applied in the test. Software tests can be distinguished according to the following levels:

- With regard to the different software components, a distinction can be made between the testing of the executable program (including the user interface), the testing of the source code, the testing of the documentation and, where appropriate, the testing of data stocks (see Fig. 3).
- With regard to the instantaneous position of a software product within the software life cycle, we can distinguish between the testing of the final product and the testing of intermediate products. Apart from the testing of the final product (which is described below in more detail), the scope of testing also includes the testing of products in the early stages of a software life cycle, e.g. reviews of requirement and design documents. The test of a final software product against the original user requirements is called software validation.
- With regard to the testing techniques applied, a distinction can be made as follows (see Fig. 4):
  - manual inspections (e.g. evaluation of program source codes, documentation, requirements specification, design documents),
  - tool-supported program function tests (systematic execution of programs including actual/set-point value comparison, testing of user interfaces),
  - structural tests (e.g. tests based on data/control flow, determination of test coverage),
  - tool-supported static analyses of the source code (e.g. check whether programming rules are observed, data flow analyses).
- With regard to the technical objective of software tests, the following uncompleted distinctions can be made:
  - test of functionality,
  - test of conformity with standards or normative regulations,
  - test of the programming technique,
  - test of ergonomic characteristics,
  - test for data and program security.

Though not complete, the bullets regarding the test objectives serve as a guideline for the explanations below.

4.1. Functionality test

All functions of a program are checked for compliance with an individual set of requirements. To test the functionality, a specification, a list or catalogue of requirements must be available or at least precise ideas about the required or desired scope of functions.

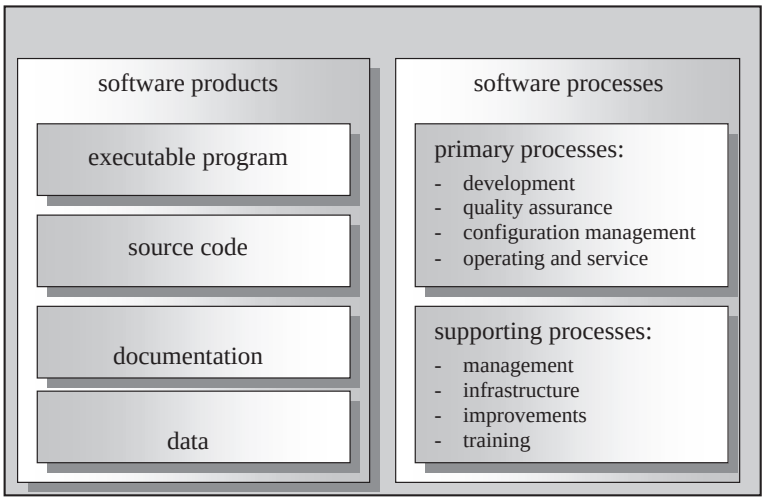

Fig. 3. Components of the software under test.

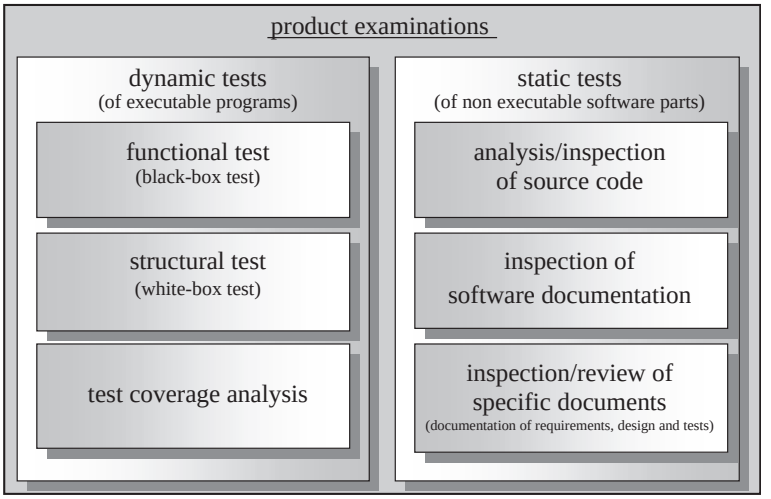

Fig. 4. Techniques of software product evaluation.

Experience has shown that the formulation of such requirements is often imprecise (see Section 3). As testing techniques, mainly functional and structural tests are applied. Code inspections, too, are of some help.

4.2. Conformity test

The software is tested for conformity with software-specific or domain-specific standards or normative regulations. Tests are carried out for example with regard to the following regulations:

- ISO/IEC 12119 [9]: Testing whether a software is suitable as COTS product.
- ISO 9241 [10]: Testing whether a software is fit for its intended use (see further below).
- OECD Principles of Good Laboratory Practice (GLP) [6] and EA Guideline [7]: Testing of compliance with regulations prescribed or recommended for the use of computers and software in (accredited) laboratories.

If necessary, compliance with other software- and domain-specific standards or guidelines for laboratories is tested. Testing techniques frequently applied are functional and structural tests as well as the inspection of program source code and documentation.

4.3. Test of the programming technique

By testing the quality of the programming technique, evidence is to be given that the software works in a robust and reliable manner. For this code-specific test of non-functional requirements, code inspections and tool-supported static analyses, especially data flow analyses [5], are used. The following questions can be answered by this category of test:

- Has the software been implemented according to the state of the art, and have any specified programming rules been complied with? For example, the following error categories can be detected by the tests: Initialisation problems, memory access problems, side effects.
- Are the data processed correctly from the moment of their input until the moment of their output? Is it possible to reconstruct the relationship between output value (e.g. price of the product indicated on the weighing instrument) and input values (sensor value, measurement range) by means of the program source code?

To answer the questions of the last bullet, methods of data flow analysis [5] are of essential help. This special method can, for example, be used to test whether security against software manipulation in measuring systems subject to legal control is assured.

Software which has to be protected against tampering is found, for example, in electronic weighing instruments and in fuel dispensers at petrol stations. To protect the software, it cannot, however, be "sealed" completely. From time to time, it must be possible for the shop assistant or service-station attendant to adjust the price of his products per kilogram or litre. For this purpose, he must have limited access to certain data storage areas of the software. He is not, however, supposed to influence the formation of the weight values, for example. When the weighing instruments are type-approved, evidence must be provided that those components of the software which are subject to legal control and form the measurement value are not influenced by the permitted price adjustments made by the shop assistant and that no criminal manipulation of the software is possible or at least it is not possible by simple means. For this purpose, extensive software-specific tests have to be carried out based on the analysis of the program data flow.

4.4. Test of ergonomic characteristics

Here, the usability and the fitness for the intended purpose of a software product according to the international standard ISO 9241-10/11 [10] is tested for a specific context of use. Essential criteria to evaluate how the interaction between user and software is designed and implemented are:

- *Suitability for the task*. The user shall be supported in such a way that he can accomplish his task effectively and efficiently.
- *Self-descriptiveness*. All dialog steps are immediately understandable and explained on request.
- *Controllability*. The user shall be able to start the dialog and to influence its direction and speed until he reaches the objective.
- *Conformity with user expectations*. The dialog shall be consistent and tailored to the knowledge of the user.
- *Fault tolerance*. Even if the user makes inputs which obviously are erroneous, he shall be able to achieve the intended objective with only very little effort.
- *Suitability for individualisation*. The program shall be so designed that it can be adjusted to the requirements of the task and to the individual capabilities and preferences of the user.
- *Suitability for learning*. The user shall be supported and guided in learning the program dialogs.

The test of ergonomic characteristics includes a functionality test according to the above-mentioned ISO/IEC 12119 [9].

4.5. Security test

The testing and evaluation of software security within the scope of information technology security is carried out according to special security criteria laid down in international regulations, such as:

- ITSEC: Information Technology Security Evaluation Criteria [11],
- CC: Common Criteria for Information Technology Security Evaluation [12].

Conformity tests are mainly carried out for the following quality characteristics:

- *Availability*: Data and services must at any time be available to authorised users,
- *Confidentiality*: Information shall be available to authorised users only,
- *Integrity*: Data and programs must be protected from unintended or unauthorised modifications (including complete loss),
- *Authenticity*: Programs must clearly identify the communication partner (user, process) of protected transactions.

In particular, the existence and efficiency of security measures is tested. Such measures can be:

- controls of access to programs, password systems;
- restrictions of access to certain data storage areas or documents;
- role concepts, graded granting of rights;
- measures for anti-virus protection, firewalls;
- plausibility checks for all input data;
- database integrity rules;
- archiving and backup measures;
- disaster recovery.

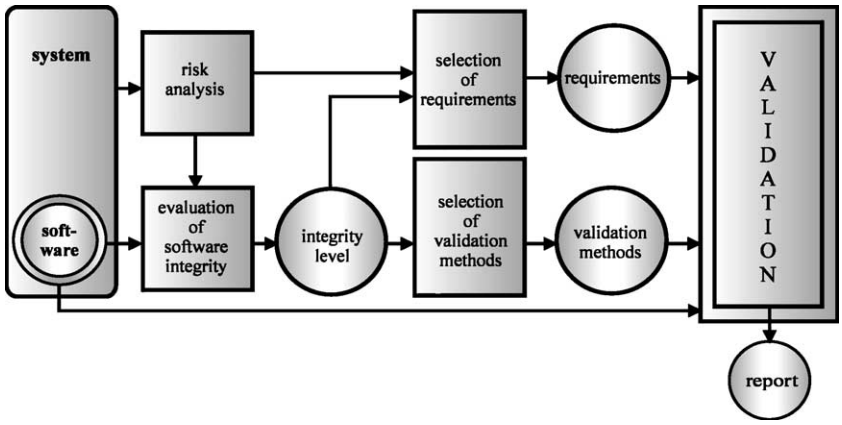

Fig. 5. Schematic representation of the software validation process.

4.6. Execution of software product tests

To warrant the repeatability and comparability of tests, all requirements, test data and methods used in the tests must be well defined and documented. Also, the organisational sequences which are repeated in each test have to be laid down as working instructions. For example, the following questions must have been clarified in advance for each test:

- What requirements are to be met by the software to be tested (see Section 3)?
- Which criticality (risk/integrity level) is allocated to the software system, to its components and, if appropriate, to each individual requirement to be checked?
- What normative documents (standards, guidelines, regulations) are to be observed?
- Which test concept or testing procedure is used (testing depth, selection of test data and methods, selection and adjustment of procedures, analyses of testing results)?

Fig. 5 shows the general diagram and the essential elements of a software validation process.

5. Process assessment and improvement

It is the precondition for high quality of a software product that its overall development process and all its sub-processes are of high quality, too. This is achieved by assessing the overall process and its sub-processes and improving them on the basis of the assessment results. The assessment of processes and their improvement are cyclical elements.

Formal or informal assessments of software development processes may be applied in the form of audits at the software producer's site. On the one hand, formal assessment systems are offered by the International Standards series ISO/IEC 15504 (SPICE) [13] or the assessment models CMMI (Capability Maturity Model Integration) [14]. On the other hand, in informal assessments, made-to-measure lists of questions based on relevant process standards (such as ISO/IEC 12207 [15]) can be compiled and applied. If the answers to the audit questions are positive and if the overall assessment of the development process is positive, too, this can be regarded as an important indication that the producer's software is in compliance with the latest state of the art. All in all it can be assumed that in the case of an overall positive judgement, good testability and maintainability of the software product is ensured.

A catalogue of questions used by the PTB Software Test Centre in audits at the producer's site has been compiled in [2] and at www.softwarepruefstelle.de. To illustrate the procedure and thoroughness of an audit, some questions are given in the following.

5.1. Software testing

- Are adequate software testing procedures used in the project (test planning, test case determination,

test data generation, performance of tests, test analysis, test documentation)?
- Are the methods used documented?
- Who carries out the tests? Is there a separate group for this?
- Are reviews of the results of the different stages carried out at regular intervals (e.g. review of the requirements specification, design documents, test schedules)?
- Who draws up the test schedules and by what means?
- Who carries out the functional test?
- Are there test records and summarising test reports?
- Is there a procedure to record and archive the test results?
- By what means is it assured that all user requirements and product functions have been checked?
- Who authorises test schedules, test records, test reports?
- Are confidence-building internal audits of the software development processes carried out?
- Can the producer guarantee access to the test environment (testing tools, test data) and to the test documentation?

5.2. Software documentation

- Are there any company-specific standards or guidelines for the preparation and maintenance of the software documentation?
- Is the documentation compiled parallel to the project?
- Is the documentation subject to regular reviews?
- What is the scope of the documentation supplied?
- For which time is the documentation valid?
- Is the documentation subject to version management?
- Can the producer guarantee access to the software development documentation?
- Can the producer guarantee access to the source code (if necessary)?

5.3. Error messages and change management

- Does a formal problem message procedure with feedback to the customer or to the test team exist?
- How does the producer react to error messages and suggestions for improvement and how does he deal with them?
- In what way are the customers or the test team informed about how the matter is handled?
- Are error statistics (error types, frequencies) compiled?
- Who initiates software modifications?
- Who approves software modifications?
- Are there documented methods for change and version management?
- Do the procedures of the change management also apply – apart from programs – to the software documentation and other documents such as test schedules, test reports or design schedules?
- Are new software and document versions systematically identified?
- How are the customers informed about new versions?
- Is a configuration management tool used?
- Are there any provisions as to which tests have to be repeated in case a version is modified, and in which way they have to be carried out?

If necessary, these audit questions have to be extended or refined to allow producer- or domain-specific checklists to be derived.

Finally, the quality of the software development processes can be improved by the following means:

- the results of the audits having been carried out at the producer's are directly put into practice (this has a direct impact on the processes and, consequently, on the products),
- use of auxiliary means in software development: standards, working instructions, checklists, process models, guidelines (e.g. for design, programming, documentation, testing),
- use of validated software tools (compiler, configuration management),
- use of validated requirements catalogues,
- re-use of high-quality software or software components.

6. The accredited Software Test Centre at PTB

To raise customer confidence in the correctness and reliability of software systems, a software test centre

has been established at PTB which was accredited as test laboratory according to ISO/IEC 17025 [16] by DATech (Deutsche Akkreditierungsstelle Technik e.V.) in February 2001. This accreditation confirms the competence of the Software Test Centre, the observance of widely accepted requirements specified in international standards for test laboratories, and the comparability of the laboratory's test results with those of other software test centres throughout the world. The scope of accreditation ranges from the testing of software for functionality, reliability, security and usability (ISO/IEC 12119 [9], ISO/IEC 9126 [8]) to the detailed testing by ergonomic criteria (ISO 9241-10-17 [10]). Test objects are measuring device software, software in measuring facilities and test assemblies, software in calibration devices, software for the processing of measurement data, and software in other devices to be tested by the PTB or other national institutes (e.g. voting machines). If appropriate, software development processes, too, can be subjected to testing.

For the testing of software products, different test methods are applied, e.g. manual inspection of the design documents, the program source code and its documentation (inspection and evaluation); tool-supported program function test (systematic execution of programs including setpoint/actual-value comparison); tool-supported analysis of the program source code (to check observance of programming rules); and special ergonomics-oriented test methods. Other services of the Software Test Centre are preventive software quality assurance measures accompanying the development process, e.g. audits at the producer's to monitor the software development processes and consultation and support of software producers. The Test Centre supports software producers, especially when their audit results have been negative, by developing, adjusting and providing auxiliary means for improving their development processes. These auxiliary means include process models for software development, guidelines for software documentation and for the use of programming languages. Such programming guidelines were developed for the languages C, C+, Fortran, Pascal, and Visual Basic. The Software Test Centre offers its services not only to PTB laboratories and other national institutes but also to the producers of measuring and testing devices and to other testing and calibration laboratories. The guidelines and requirements catalogues mentioned above, and further information on the Software Test Centre can be found in the Internet at www.softwarepruefstelle.de.

7. Outlook

Main elements of conformity assessment are testing and certification of products or processes, the producer's declaration, and accreditation. For these elements, two trends become apparent in software quality assurance which are of importance for metrology in particular.The first trend is the increasing orientation towards preventive quality assurance for the development processes. Extensive time and labour consuming product tests are simplified or replaced by process audits, producer's declarations or the testing of samples. In the field of legal metrology, this trend is accounted for by the introduction of specific conformity assessment modules H and H1 within the framework of the MID [17]. In future it will be our task to support this shift of emphasis by organisational and technical means. Besides, it is necessary to work out instructions for process audits at the producer's site. For areas requiring a high degree of measurement correctness or running an increased risk of manipulation, it is advisable to carry out systematic tests of intermediate products based on specific risk analyses.

The second trend regards product tests: There is an increasing demand that these tests should be repeatable and comparable. This would ensure the traceability of the test results and the objectivity of the evaluation. This can, however, be achieved only if the tests are carried out on the basis of detailed working instructions. During the test, relevant intermediate results, all final results and the respective test environment (test engineer, hardware, software…) are to be documented. Accredited software test centres must meet these requirements. In future, greater attention will have to be paid to the reliability of product testing. For this purpose, studies are necessary in the field of test coverage and software metrics. The following questions will have to be answered more profoundly in the future: When can the test be terminated? Is the entire scope of the program covered by test cases? Which test cases are still missing in which program components and have therefore to be carried

out subsequently? Which software metrics help evaluate the software product and the test results?

References

[1] N. Greif, Richtig funktionierende Software: Anforderungen und ihre Überprüfung, Sensoren und Messsysteme, VDI-Berichte 1829, VDI-Verlag, ISBN: 3-18-091829-2, 2004, pp. 85–95.

[2] N. Greif, H. Schrepf, Software requirements for measuring systems — Examples for requirements catalogues, PTB Laboratory Report, PTB-8.31-2000-2, Braunschweig and Berlin, July, 2000.

[3] N. Greif, H. Schrepf, D. Richter, Software Evaluation in Calibration Services: Requirements and Testing Procedure, in: D. Richter, V. Granovski (Eds.), Methodological Aspects of Data Processing and Information Systems in Metrology, PTB Report PTB-IT-7, 60–72, ISBN 3-89701-379-7, Braunschweig and Berlin, June, 1999.

[4] N. Greif, D. Richter, Software Engineering Related Standards and Guidelines for Metrology, in: P. Ciarlini, A. Forbes, F. Pavese, D. Richter (Eds.), Advanced Mathematical and Computational Tools in Metrology IV, World Scientific, London, ISBN: 9810242166, 2000, pp. 109–121.

[5] N. Greif, H. Schrepf. Towards Secure Measurements Through Software Analysis, in: D. Richter, V. Granovski (Eds.), Methodological Aspects of Data Processing and Information Systems in Metrology, PTB Report PTB-IT-7, 52–59, ISBN 3-89701-379-7, 1999, June.

[6] OECD Series on Principles of Good Laboratories Practice, No. 10: The Application of the Principles of GLP to Computerised Systems, OECD/GD(95)115, 1995.

[7] EA Guidelines for the Use of Computers and Computer Systems in Accredited Laboratories, Version 4, May, 1998.

[8] ISO/IEC 9126-1: 2001, Software Engineering — Product Quality — Part 1: Quality Model.

[9] ISO/IEC 12119: 1994, Information Technology — Software Packages — Quality Requirements and Testing.

[10] ISO 9241, Ergonomic Requirements for Office Work with Visual Display Terminals, 1997.

[11] Information Technology Security Evaluation Criteria (ITSEC), Version 1.2, Commission of the European Communities, ISBN: 92-826-3004-8, 1991.

[12] Common Criteria for Information Technology Security Evaluation (CC), Version 2.1, 1999, published also as ISO/IEC 15408:1999.

[13] ISO/IEC 15504-1/9: 1998, Information Technology — Software Process Assessment (SPICE).

[14] Capability Maturity Model Integration (CMMI), http://www.sei.cmu.edu/cmmi/cmmi.html, 2002.

[15] ISO/IEC 12207: 1995, Information Technology — Software Life Cycle Processes.

[16] ISO/IEC 17025, General requirements for the competence of testing and calibration laboratories, 2000.

[17] Measuring Instruments Directive (MID), Directive 2004/22/EC, European Council, 2004.

[18] WELMEC Guide 7.2: Software Guide (MID), http://www.welmec.org, 2005.

Norbert Greif graduated in Mathematics at the Technical University of Freiberg, Germany, in 1979. He received a Ph.D. in Computer Science in 1988. Since 1993, he has been working at Physikalisch-Technische Bundesanstalt (PTB), the National Metrology Institute of Germany in Berlin. In the department “Metrological Information Technology”, he deals with advanced software engineering, quality assurance techniques and software testing in the scope of software for metrology and voting systems. He is head of an accredited software testing laboratory.