Applying the ISO 9126 Quality Model to Test Specifications
—
Exemplified for TTCN-3 Test Specifications

Benjamin Zeiss*‡, Diana Vega**‡‡, Ina Schieferdecker‡‡,
Helmut Neukirchen‡, Jens Grabowski‡

‡ Software Engineering for Distributed Systems Group, Institute for Informatics,
University of Göttingen, Lotzestr. 16–18, 37083 Göttingen, Germany.
{zeiss,neukirchen,grabowski}@cs.uni-goettingen.de

‡‡ ETS, Technical University Berlin, Franklinstr. 28–29, 10587 Berlin, Germany.
{vega,ina}@cs.tu-berlin.de

Abstract: Quality models are needed to evaluate and set goals for the quality of a software product. The international ISO/IEC standard 9126 defines a general quality model for software products. Software is developed in different domains and the usage of the ISO/IEC quality model requires an instantiation for each concrete domain. One special domain is the development and maintenance of test specifications. Test specifications for testing, e.g. the Internet Protocol version 6 (IPv6) or the Session Initiation Protocol (SIP), reach sizes of more than 40.000 lines of test code. Such large test specifications require strict quality assurance. In this paper, we present an adaptation of the ISO/IEC 9126 quality model to test specifications and show its instantiation for test specifications written in the Testing and Test Control Notation (TTCN-3). Example measurements of the standardised SIP test suite demonstrate the applicability of our approach.

1 Introduction

Test specifications developed today by industry and standardisation are usually voluminous and are, in our experience, often regarded as complex. Such statements are based on subjective opinions about few quality aspects that stand out, but neither is the term "complexity" clearly defined nor is it evident how it relates to the quality of a test specification. In the context of software engineering, metrics are a common means to quantify quality aspects of software. Metrics are classified into those that concern products, processes, and resources. While metrics support the measurement of quality aspects, they do not provide an answer what constitutes quality in software. To gain reasonable statements from metrics, a quality model is required which defines distinct characteristics and corresponding subcharacteristics that relate to software quality. ISO/IEC 9126 [ISO04] is a standard describing such a model for software products.

*Supported by a PhD scholarship from Siemens AG, Corporate Technology.
**Supported by Alfried Krupp von Bohlen und Halbach-Stiftung.

Quality aspects of test specifications are somehow related to the characteristics stated in ISO/IEC 9126. However, a more elaborated analysis on the peculiarities and differences between the quality aspects that constitute test specifications and software has not been made yet. Still, quality aspects concerning test specifications and test implementations are constantly subject of discussions and various test metrics have been developed measuring single aspects only [Sne04, VS06, ZNG+06a, ZNG+06b]. Thus, a more general view on the different quality aspects of test specifications is needed. The contribution of this paper is a quality model for test specifications which is derived from the ISO/IEC 9126 quality model.

For concrete investigations, we selected the *Testing and Test Control Notation* (TTCN-3) [ETS05a] which is standardised by the *European Telecommunications Standards Institute* (ETSI). TTCN-3 is a language for test specification and implementation, i.e. it supports abstract test specifications which can be compiled and executed if additional implementation components (such as an *SUT adapter*) are provided.

This paper is structured as follows: Section 2 introduces ISO/IEC 9126. Subsequently, our quality model for test specifications is presented and discussed in Section 3. Section 4 describes an instantiation of our quality model for TTCN-3. An application of our TTCN-3 specific model to different versions of *Session Initiation Protocol* (SIP) test specifications is given in Section 5. We conclude with a summary and an outlook.

2 Overall View of ISO/IEC 9126

Based on previous attempts for defining software quality [MRW77, BBK+78], the *International Organization for Standardization* (ISO) and the *International Electrotechnical Commission* (IEC) have published the multipart standard ISO/IEC 9126 [ISO04] which defines a software product quality model, quality characteristics, and related metrics. These constituents can be used to both evaluate and set goals for the quality of a software product.

Part 1 of ISO/IEC 9126 contains a two-part quality model: one part of the quality model is applicable for modelling the internal and external quality of a software product, whereas the other part is intended to model the quality in use of a software product. These different quality models are needed to be able to model the quality of a software product at different stages of the software lifecycle. Typically, *internal quality* is obtained by reviews of specification documents, checking models, or by static analysis of source code. *External quality* refers to properties of software interacting with its environment. In contrast, *quality in use* refers to the quality perceived by an end user who executes a software product in a specific context. These product qualities at the different stages of development are not completely independent, but influence each other. Thus, internal metrics may be used to predict the quality of the final product – also in early development stages.

For modelling internal quality and external quality, ISO/IEC 9126 defines the same model. This generic quality model can then be instantiated as a model for internal quality or for external quality by using different sets of metrics. The model itself is based on the six characteristics *functionality*, *reliability*, *usability*, *efficiency*, *maintainability*, and *portability*. As shown in Figure 1, each of these characteristics has further subcharacteristics.

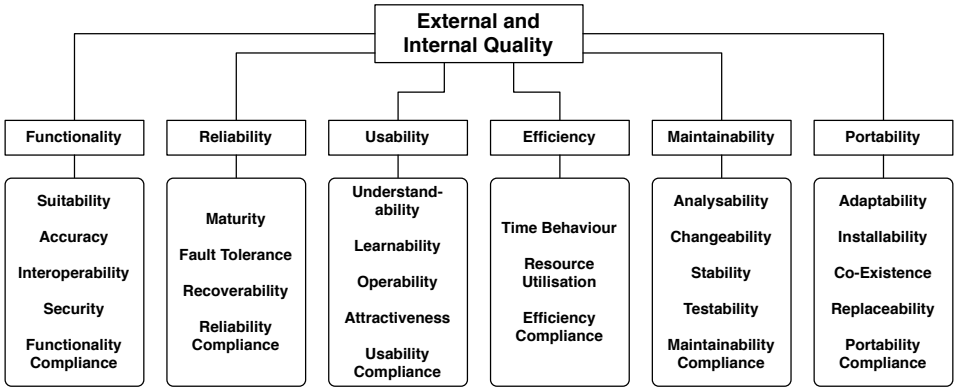

Figure 1: The ISO/IEC 9126-1 Model for Internal and External Quality

The model of quality in use is based on the characteristics *effectiveness*, *productivity*, *safety*, and *satisfaction* and does not elaborate on further subcharacteristics. In the further parts of ISO/IEC 9126, metrics are defined which are intended to be used to measure the attributes of the (sub)characteristics defined in Part 1: The provided metrics are quite abstract which makes them applicable to various kinds of software products, but they cannot be applied without further refinement.

The actual process of evaluating a software product is not part of ISO/IEC 9126, but it is defined in ISO/IEC 14598 [ISO01]: To be able to take different requirements of different products into account, the model needs to be *instantiated* by weighting the different (sub-) characteristics and by choosing appropriate metrics.

3 A Quality Model for Test Specifications

Our quality model for test specification is an adaptation of ISO/IEC 9126 to the domain of test specification. While the ISO/IEC 9126 model deals with internal quality, external quality, and quality in use, the remainder of this paper will only address internal quality characteristics.

Figure 2 illustrates our test specification quality model. The model is divided into seven main characteristics: test effectivity, reliability, usability, efficiency, maintainability, portability, and reusability. Each main characteristic is structured into several subcharacteristics.

While most of the characteristics defined in ISO/IEC 9126 can be generously re-interpreted and thus applied for test specifications as well, we preferred to introduce names which are more appropriate in the context of testing. To indicate the relationship of our model to ISO/IEC 9126, we provide the corresponding name of the ISO/IEC 9126 characteristics in parenthesises. Test quality characteristics are printed in bold letters. Characteristics which have no corresponding aspect in ISO/IEC 9126, are denoted by the sign (–). The seven

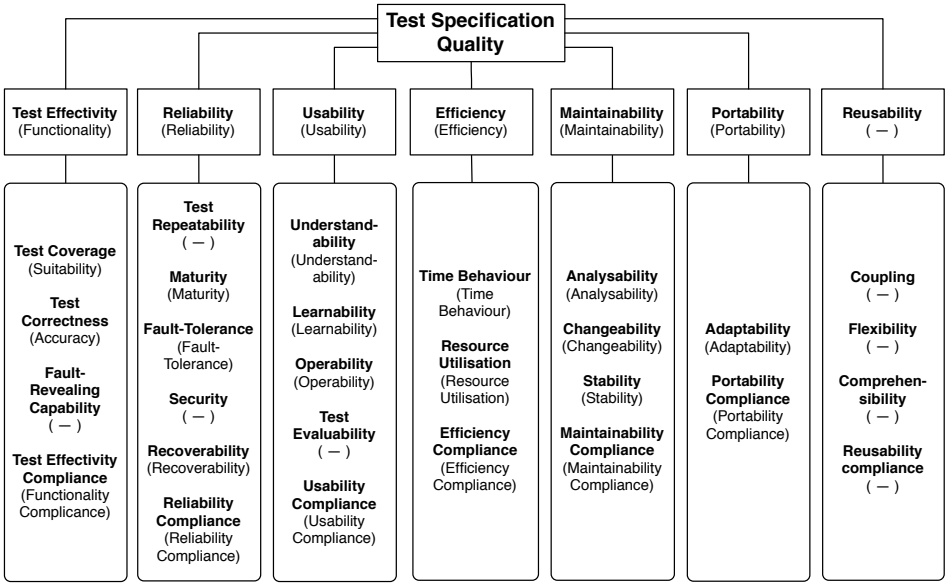

Figure 2: The Test Specification Quality Model

characteristics are explained in more detail in the following paragraphs. Characteristics that are not applicable for test specifications are also reviewed.

The main characteristic *reusability* (right-hand side of Figure 2) is not explicitly covered in ISO/IEC 9126. We added it to our model, because test specifications and parts of them are often reused for different kinds of testing, e.g. test cases and test data for system level testing may be reused for regression testing, performance testing, or testing different versions of the *System Under Test* (SUT). Thus, design for reusability is an important quality criterion for test specifications.

Each main characteristic contains a *compliance* subcharacteristic which denotes the degree to which the test specification adheres to potentially existing standards or conventions concerning this aspect. Since such conventions also exist for test design, they are also included in our model. However, they will not be covered any further in the following descriptions since such conventions and standards are company- or project-specific.

Test Effectivity. The *test effectivity* characteristic describes the capability of the specified tests to fulfil a given test purpose. *Test effectivity* is the characterisation of the term "functionality" in the context of test specification and was thus renamed from ISO/IEC 9126. In the context of test specification, the *suitability* aspect is characterised by *test coverage*. Coverage constitutes a measure for test completeness and can be measured on different levels, e.g. the degree to which the test specification covers system requirements, system specification, or test purpose descriptions.

The *test correctness* characteristic denotes the correctness of the test specification with respect to the system specification or the test purposes. Furthermore, a test specification is only correct when it always returns correct test verdicts and when it has reachable end states.

The *fault-revealing capability* has been added to the list of subcharacteristics. Obtaining a good coverage with a test suite does not make any statement about the capability of a test specification to actually reveal faults. Usage of cause-effect analysis [Mye79] for test creation or usage of mutation testing may be indicators for increased attention to the fault-revealing capability.

The *interoperability* characteristic has been omitted from the test specification quality model. Test specifications are too abstract for *interoperability* to play a role. The *security* aspect has been moved to the *reliability* characteristic.

Reliability. The *reliability* characteristic describes the capability of a test specification to maintain a specific level of performance under different conditions. In this context, the word "performance" expresses the degree to which needs are satisfied.

The *reliability* subcharacteristics *maturity*, *fault-tolerance*, and *recoverability* of ISO/IEC 9126 apply to test specifications as well. However, new subcharacteristics *test repeatability* and *security* have been added. Test results should always be reproducible in subsequent test runs if generally possible. Otherwise, debugging the SUT to locate a defect becomes hard to impossible. *Test repeatability* includes the demand for deterministic test specifications.

The *security* subcharacteristic covers issues such as included plain-text passwords that play a role when test specifications are made publicly available or are exchanged between development teams.

Usability. The *usability* attributes characterise the ease to actually instantiate or execute a test specification. This explicitly does not include usability in terms of difficulty to maintain or reuse parts of the test specification which are covered by other characteristics.

Understandability is important since the test user must be able to understand whether a test specification is suitable for his needs. Documentation and description of the overall purpose of the test specification are key factors – also to find suitable test selections.

The *learnability* of a test specification pursues a similar target. To properly use a test suite, the user must understand how it is configured, what kind of parameters are involved, and how they affect test behaviour. Proper documentation or style guides have positive influence on this quality as well.

A test specification has a poor *operability* if it, e.g. lacks appropriate default values, or a lot of external, i.e. non-automatable, actions are required in the actual test execution. Such factors make it hard to setup a test suite for execution or they make execution time-consuming due to a limited automation degree.

A new test-specific subcharacteristic in *usability* is *test evaluability*. The test specification must make sure that the provided test results are detailed enough for a thorough analysis. An important factor is the degree of detail richness in test log messages.

Lastly, *attractiveness* is not relevant for test specifications. *Attractiveness* may play a role for test execution environments and tools, but for plain test specifications, there simply is no user interface involved that could be liked or not.

Efficiency. The *efficiency* characteristic relates to the capability of a test specification to provide acceptable performance in terms of speed and resource usage. The ISO/IEC 9126 subcharacteristics *time behaviour* and *resource utilisation* apply without change.

Maintainability. *Maintainability* of test specifications is important when test developers are faced with changing or expanding a test specification. It characterises the capability of a test specification to be modified for error correction, improvement, or adaption to changes in the environment or requirements. The *analysability*, *changeability*, and *stability* subcharacteristics from ISO/IEC 9126 are applicable to test specifications as well. The *testability* subcharacteristics does not play any role for test specifications.

The *analysability* aspect is concerned with the degree to which a test specification can be diagnosed for deficiencies. For example, test specifications should be well structured to allow code reviews. Test architecture, style guides, documentation, and generally well structured code are elements that have influence in the quality of this property.

The *changeability* subcharacteristic describes the capability of the test specification to enable necessary modifications to be implemented. E.g. badly structured code or a test architecture that is not expandable may have negative impact on this quality aspect.

Depending on the test specification language used, unexpected side effects due to a modification have negative impact on the *stability* aspect.

Portability. *Portability* in the context of test specification does only play a very limited role since test specifications are not yet instantiated. Therefore, *installability* (ease of installation in a specified environment), *co-existence* (with other test products in a common environment), and *replaceability* (capability of the product to be replaced by another one for the same purpose) are too concrete. However, *adaptability* is relevant since test specifications should be capable to be adapted to different SUTs or environments. For example, hardcoded SUT addresses (e.g. IP addresses or port numbers) or access data (e.g. user names) in the specification make it hard to adapt the specification for other SUTs.

Reusability. Although *reusability* is not part of ISO/IEC 9126, we consider this aspect to be particularly important for test specifications since it matters when test suites for different test types are specified. For example, the test behaviour of a performance or stress test specification may differ from a functional test, but the test data, such as predefined messages, can be reused between those test suites. It is noteworthy that the subcharacteristics correlate with the *maintainability* aspects to some degree.

The *coupling* degree is arguably the most important subcharacteristic in the context of reuse. Coupling can occur inbetween test behaviour, inbetween test data, and between test behaviour and test data. For example, if there is a function call within a test case, the test

case is coupled to this function. To make test specifications reusable, the ultimate goal is loose coupling and strong cohesion.

The *flexibility* of a test specification is characterised by the length of a specification subpart and its customiseability regarding unpredictable usage. For example, if fixed values appear in a part of a test specification, a parametrisation likely increases its reusability.

Finally, parts of a specification can only be reused if there is a good understanding of the reusable parts (*comprehensibility* subcharacteristic). Good documentation, comments, and style guides are necessary to achieve this goal.

4 An Instantiated Test Specification Quality Model for TTCN-3

Our quality model for test specifications (see Section 3) is kept abstract to support the application to different test specification technologies like, e.g. TTCN-3 [ETS05a] or *UML 2.0 Testing Profile* (U2TP) [OMG05]. The instantiation of the test specification quality model requires a set of metrics for each subcharacteristic that adequately capture the different aspects in numbers. There are various ways to obtain these numbers: static analysis, dynamic analysis on the specification level, but also results from manual reviews of specification documents. The latter may include the comparison of different kinds of test specification documents to assess the degree of consistency between them or coverage of specifications.

Due to our involvement in the standardisation, we chose TTCN-3 as test specification language to demonstrate the instantiation of our quality model. The instantiation depends on a variety of different aspects like application specific properties, customer-specific requirements, or weaknesses of test developers. Hence, our set of metrics should not be misconceived as a fixed set that cannot be changed. Rather, they represent a variable excerpt of what measurements may be suitable for each subcharacteristic.

A well known methodology to find appropriate metrics is the *Goal Question Metric* (GQM) approach [BW84] which we used to obtain suitable TTCN-3 specific metrics. In the following, we provide example metrics for three main characteristics: *test effectivity*, *maintainability*, and *reusability*.

Main Characteristic: Test Effectivity

Subcharacteristic 1: *test coverage*

- metric 1.1: *test purpose coverage* := $\frac{\text{number of test purposes covered by TTCN-3 test cases}}{\text{overall number of test purposes}}$, i.e. the number of test purposes covered by test cases specified in a TTCN-3 test suite is compared to the number of test purposes contained in a corresponding test purpose specification.
- metric 1.2: *system model coverage* := $\frac{\text{test coverage of system model}}{\text{possible coverage of system model}}$, where several different coverage criteria like path coverage, branch coverage, etc. are applicable. This metric determines the test coverage with respect to a model of the SUT.

Subcharacteristic 2: *test correctness*

- metric 2.1: *test verdict completeness* := $\frac{\text{number of paths in TTCN-3 test cases setting a test verdict}}{\text{overall number of paths in TTCN-3 test cases}}$, i.e. it is assessed whether all paths of the test cases do set a test verdict.
- metric 2.2: *test termination* := $\frac{\text{number of paths in TTCN-3 test cases terminating correctly}}{\text{overall number of paths in TTCN-3 test cases}}$, i.e. it is assessed whether all paths of the test cases terminate correctly.

Subcharacteristic 3: *fault-revealing capability*

- metric 3.1: *transmissibility of receiving templates* :=

$$1 - \frac{\text{number of wildcard-only-covered elements in type definitions used by receiving templates}}{\text{overall number of elements in type definitions used by receiving templates}}$$

  SUT responses might never be properly evaluated when the corresponding receiving templates are too transmissible due to wildcards. This metric measures to which degree the elements of received data are at least covered once by a non-wildcard expected value.
- metric 3.2: *effect coverage* := $\frac{\text{number of effects tested by a TTCN-3 test suite}}{\text{overall number of effects possible in the system specification}}$. This metric uses cause-effect analysis to determine the degree to which each effect is at least tested once.

Main Characteristic: Maintainability

Subcharacteristic 1: *analysability*

- metric 1.1: *complexity violation* :=

$$1 - \frac{\text{number of behavioural entities violating upper bound of complexity}}{\text{overall number of behavioural entities}}$$

  This metric measures the number of TTCN-3 testcases, functions, and altsteps which violate a defined boundary value of a complexity measure in comparison to the overall number of testcases, functions, and altsteps. Several complexity measures may be used, e.g. McCabe's cyclomatic number [McC76, ZNG+06b], nesting level, call-depth, or number of statements.

Subcharacteristic 2: *changeability*

- metric 2.1: *code duplication* := $1 - \frac{\text{entities containing duplicated code}}{\text{overall number of entities}}$. Since changes to duplicated code requires changing all locations of duplication, this metric determines the portion of duplicated code in terms of, e.g. *Lines of Code* (LOC) or statements.
- metric 2.2: *maximum number of references violation* :=

$$1 - \frac{\text{number of entities which are referenced more times than an upper bound allows}}{\text{overall number of entities}}$$

  This metric determines how often an entity is referenced and penalises the violation of an upper boundary value. When applying changes to entities which are referenced very often, a developer needs to check for every reference whether a change may have unwanted side effects or requires follow-up changes.

Subcharacteristic 3: *stability*

- metric 3.1: *global variable and timer usage* :=

$$1 - \frac{\text{number of component variables and timers referenced by more than one behaviour}}{\text{overall number of component variables and timers}}$$

  Global variables promote side effects. In TTCN-3, component variables and timers are global to all behaviour running on the same component. This metric measures the number of all component variables and timers referenced by more than one function, testcase, or altstep and relates them to the overall number of component variables and timers.

- metric 3.2: *parameter reassignment* $:= 1 - \frac{\text{number of out and inout parameters}}{\text{overall number of parameters}}$. Any modification of parameters which are passed into a testcase, function, or altstep as *out* or *inout* parameter leads to a side effect. Hence, this metric measures the potential of side effects by relating the number of out and inout parameters to the overall number of parameters.

Main Characteristic: Reusability

Subcharacteristic 1: *coupling*

- metric 1.1: *coupling to other modules* $:= \frac{\text{number of modules importing from other modules}}{\text{overall number of modules}}$. The reusability of the modules of a test suite depends on how tightly each module is coupled to other modules. Hence, this metric counts the number of modules coupled to other modules and relates this to the overall number of modules. In TTCN-3, coupling between modules is introduced by the *import* construct.

- metric 1.2: *coupling to overspecialised components* :=

$$\frac{\text{number of behavioural entities unnecessarily running on a specialised component}}{\text{overall number of behavioural entities running on components}}$$

  Reusability is reduced if a function, testcase, or altstep running on a component would run on a parent component as well, but is bound to a more specialised component. Hence, this metric relates the number of such cases to the overall number of functions, testcases, and altsteps which are coupled to components in general.

Subcharacteristic 2: *flexibility*

- metric 2.1: *shortness* $:= \frac{\text{number of behavioural entities violating size limit}}{\text{overall number of behavioural entities}}$. The shorter an entity is, the higher the probability is that it is flexible enough to be reused in a different context. Hence, this metric measures the number of testcases, functions, and altsteps whose LOC or number of statements violate a defined boundary value.

- metric 2.2: *parametrisation* $:= \frac{\text{number of formal parameters}}{\text{number of formal parameters} + \text{number of hardcoded values}}$. Reuse is hindered by hardcoded values and promoted by parametrisation. Hence, this metric values parametrisation and penalises hardcoded values.

Subcharacteristic 3: *comprehensibility*

- metric 3.1: *comments* $:= \frac{\text{number of commented entities}}{\text{overall number of entities}}$, i.e. the number of entities, e.g. testcases, functions, or altsteps, whose interface is properly documented in comparison to the overall number of considered entities.

- metric 3.2: *groupedness* $:= 1 - \frac{\text{number of ungrouped elements}}{\text{overall number of elements}}$. In TTCN-3 grouping is a means to structure the elements of a module. Hence, this metric calculates the degree of structuredness by penalising unstructured elements.

5 Example

As an example for the usage of our test specification quality model instantiation for TTCN-3, we applied it to several versions of a TTCN-3 test suite for testing the conformance of implementations of the SIP protocol. The different versions of the test suites are based on a TTCN-3 SIP test suite standardised by ETSI [ETS05b].

From the previously described quality metrics, we have so far automated the calculation of those related to maintainability. Table 1 shows some results of the calculated metrics for different versions of the SIP test suite (we designed the quality metrics to yield a value between 0, i.e. considered quality aspect not fulfilled at all, and 1, i.e. considered quality aspect fulfilled to 100%). To give an impression of the evolution of the sizes of the SIP test suite, we provide as well some absolute values of size metrics. From these, it can be seen that between versions 2.x and 3.x the number of testcases has been increased. The further size metrics are used as input for the subsequent quality metrics.

Metric	**SIP v2.20**	**SIP v2.24**	**SIP v3.01**	**SIP v3.06**
Testcases	1068	1068	1412	1412
Behavioural entities	1961	1971	2360	2369
Violations of max. cyclomatic complexity	27	30	51	51
Violations of max. number of statements	449	460	677	681
Overall alt branches	1900	1958	2482	2534
Duplicate alt branches	1435	1471	1849	1879
Definitions	2499	2526	3369	3419
Violation of max. references to definitions	119	111	133	134
Component variables and timers	63	65	66	66
Component variables and timers with side effects	53	55	56	56
Formal parameters	3175	3224	5062	5084
Formal parameter with side effects	1237	1244	1617	1628
Analysability:				
complexity violation wrt. cyclomatic complexity	0.99	0.98	0.98	0.98
complexity violation wrt. number of statements	0.77	0.77	0.71	0.71
Changeability:				
code duplication wrt. alt branches	0.25	0.25	0.26	0.26
maximum number of references violation	0.95	0.96	0.96	0.96
Stability:				
global variable and timer usage	0.16	0.15	0.15	0.15
parameter reassignment	0.61	0.61	0.68	0.68

Table 1: Measurements of Maintainability Characteristic

For assessing the analysability subcharacteristic, the violation of behavioural complexity bounds in terms of cyclomatic complexity and number of statements has been measured. For the cyclomatic complexity, the upper boundary has been chosen to be 10, for maximum number of statements, the boundary is 20. For the changeability subcharacteristic, code duplication has been measured with respect to duplicate branches in alternatives. 75% of all alt branches are duplicated, hence the obtained quality is very low (0.25–0.26). The number of references up to which changeability is considered as good has been set to 50. The stability subcharacteristic has been evaluated based on the usage of component variables and timers as global variables. This occurs quite frequently in the SIP test suite, thus the corresponding quality is low. Furthermore, possible side effects due to reassignment of inout and out parameters have been measured. The corresponding values (0.61–0.68) can be considered as barely acceptable. The maintainability compliance has not been measured, since no maintainability guidelines were defined when creating the SIP test suites.

Most efforts for the newer versions have been spent on adding new test cases to increase the coverage and were thus additions rather than refactorings [ZNG+06a]. However, they had minimal impact concerning the quality aspects measured. This can be interpreted positively considering that additions or changes can also lead to software ageing. However, some measurements, for example the high number of duplicate alt branches, indicate that there is room for improvement regarding maintenance.

6 Summary and Outlook

In this paper, we presented a quality model for test specifications. Our model is an adaptation of the ISO/IEC 9126 quality model to the domain of test development. We instantiated our model for TTCN-3 test specifications and presented measurements for different versions of the standardised SIP test suite to demonstrate the application of our approach.

Due to the domain of test specification, our model currently only covers internal quality aspects. We started to investigate a generalisation of our model which also includes external quality aspects, e.g. performance aspects and properties related to test campaigns.

A subset of the metrics presented in this paper has already been implemented in our tools [TRe07, Tes07]. We plan to implement further metrics and support for the quality assessment based on user-specific variants of our quality model. The latter may include the possibility to define user-specific profiles allowing the selection of relevant characteristics, subcharacteristics, and metrics as well as the specification of individual threshold values for metrics and the definition of evaluation schemes to combine measurements for different metrics to general quality verdicts. Furthermore, we are investigating means to evaluate whether chosen metrics are reasonable and independent, i.e. orthogonal to each other.

In addition to these activities, we started to work on further case studies (e.g. IPv6 [ZNG+06b]) and to investigate the combined usage of metrics and refactoring for a continuous quality assessment and quality improvement of test specifications. For the future, we plan to instantiate our quality model for tests specified by means of the UML 2.0 Testing Profile [OMG05].

References

[BBK+78] Boehm, B.; Brown, J.; Kaspar, J.; Lipow, M.; MacLead, C.; Merrit, M.: Characteristics of Software Quality. North Holland, 1978.

[BW84] Basili, V. R.; Weiss, D. M.: A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering, SE-10(6):728–738, 1984.

[ETS05a] ETSI Standard ES 201 873-1 V3.1.1 (2005-06): The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France, 2005.

[ETS05b] ETSI Technical Specification TS 102 027-3: SIP ATS & PIXIT; Part 3: Abstract Test Suite (ATS) and partial Protocol Implementation eXtra Information for Testing (PIXIT). European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France, 2005.

[ISO01] ISO/IEC Standard No. 14598: Information technology – Software product evaluation; Parts 1–6. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), Geneva, Switzerland, 1999-2001.

[ISO04] ISO/IEC Standard No. 9126: Software engineering – Product quality; Parts 1–4. International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC), Geneva, Switzerland, 2001-2004.

[McC76] McCabe, T. J.: A Complexity Measure. IEEE Transactions of Software Engineering, 2(4):308–320, 1976.

[MRW77] McCall, J.; Richards, P.; Walters, G.: Factors in Software Quality. Technical Report RADC TR-77-369, US Rome Air Development Center, 1977.

[Mye79] Myers, G.: The Art of Software Testing. Wiley, 1979.

[OMG05] OMG. UML Testing Profile (Version 1.0 formal/05-07-07). Object Management Group (OMG), 2005.

[Sne04] Sneed, H. M.: Measuring the Effectiveness of Software Testing. In (Beydeda, S.; Gruhn, V.; Mayer, J.; Reussner, R.; Schweiggert, F., eds.): Proceedings of SOQUA 2004 and TECOS 2004, volume 58 of Lecture Notes in Informatics (LNI). Gesellschaft für Informatik, 2004.

[Tes07] TestingTechnologies: TTworkbench. http://www.testingtech.de/products, 2007. Last visited: 1 February 2007.

[TRe07] TRex. http://www.trex.informatik.uni-goettingen.de, 2007. Last visited: 1 February 2007.

[VS06] Vega, D.-E.; Schieferdecker, I.: Towards Quality of TTCN-3 Tests. In: Proceedings of SAM'06: Fifth Workshop on System Analysis and Modelling, May 31–June 2 2006, University of Kaiserslautern, Germany, 2006.

[ZNG+06a] Zeiss, B.; Neukirchen, H.; Grabowski, J.; Evans, D.; Baker, P.: Refactoring and Metrics for TTCN-3 Test Suites. In (Gotzhein, R.; Reed, R., eds.): System Analysis and Modeling: Language Profiles, volume 4320 of Lecture Notes in Computer Science. Springer, 2006.

[ZNG+06b] Zeiss, B.; Neukirchen, H.; Grabowski, J.; Evans, D.; Baker, P.: TRex – An Open-Source Tool for Quality Assurance of TTCN-3 Test Suites. In (ASQF e.V., ed.): Software Quality in Service-Oriented Architectures – Proceedings of CONQUEST 2006. dpunkt.Verlag, 2006.