Context-specific Quality Evaluation of Test Cases

Ivan Jovanovikj, Vishwak Narasimhan, Gregor Engels and Stefan Sauer

1. Software Innovation Lab, Paderborn University, Zukunftsmale 1, Paderborn, Germany
2. Department of Computer Science, Paderborn University, Warburger Str. 100, Paderborn, Germany

Keywords: Quality Plan, GQM, Metrics, Quality Model.

Abstract: Software systems are continuously changed during maintenance and evolution. To ensure their quality, they have to be tested. But before starting testing a software system, the quality of the test cases themselves has to be evaluated. Due to the changes of the software system, they might become obsolete or even erroneous. Furthermore, test cases created in an industrial setting are extensive and at some point of time, they might have become difficult to understand, unmanageable and inefficient. Therefore, by evaluating their quality, we can better understand, control and eventually improve the quality of test cases. We present the Test Case Quality Plan (TCQP) approach, which is based on the GQM (Goal-Question-Metric) approach and enables a systematic and efficient development of quality plans. They serve as a guideline for the quality evaluation of test cases, and emphasize the context of use of test cases as a major factor of influence for the whole quality evaluation. The TCQP approach has been applied and evaluated in an industrial case study.

1. INTRODUCTION

Nowadays, software systems rapidly evolve, i.e., they are continuously changed. Along the process of changing the system, its quality has to be ensured. Software testing is one of the principal and commonly performed activity in software quality assurance.

Similarly, quality assurance is needed for test cases as well. But, it is a challenging task, due to the fact that the test cases are also constantly changed (Deursen et al., 2001; Guerra and Fernandes, 2007; Meszaros, 2007), e.g., due to system changes (Fowler and Beck, 1999). These changes involve updating, deleting, and creating new test cases. Consequently, some test cases might no longer reflect the updated behavior, i.e., they might become obsolete or even erroneous. Furthermore, test cases created in an industrial setting are extensive and at some point of time, the test cases might become difficult to understand, unmanageable and inefficient (Sneed, 2003). Therefore, by evaluating their quality, we can better understand, control and eventually improve the quality of test cases.

Currently, the work regarding quality evaluation of test cases is predominantly split into two areas. In the first area, the focus is on providing a set of internationally standardized and recognized test metrics to quantify different quality aspects of test cases (Sneed, 2004; Kaner, 2003; Bowes et al., 2017). In the other one, the focus is on providing a quality model based on existing quality models (e.g., ISO/IEC 9126 (ISO/IEC, 2001)) for the domain of test cases (Zeiss et al., 2007). However, there is no systematic approach which considers the context of use of the test cases, which is crucial for obtaining a suitable insight in the quality of the test cases, and which relies on standards like the ISO/IEC standard. Similarly as in (Voigt et al., 2008), we derive and discuss the following set of requirements that should be addressed by the solution:

Requirement 1 - Common Quality Understanding: The solution should use definitions for qualities of test cases for a consistent and common quality understanding. Among the stakeholders, every team member should have the same understanding of quality related to the test case domain. False interpretations can lead to misunderstandings and incorrect results.

Requirement 2 - Context Characterization: The solution should be able to provide a minimum set of context factors. The quality of test cases strongly depends on the context of use, in which they are created, managed, and applied. This means that different factors like the available artifacts, the environment of the test, test case type (code-based or natural language-based), etc. should be considered.

Jovanovikj, I., Narasimhan, V., Engels, G. and Sauer, S.
Context-specific Quality Evaluation of Test Cases.
DOI: 10.5220/0006724405940601
In Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2018), pages 594-601
Copyright © 2018 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved
Requirement 3 - Definition of Measurements:
The solution should distinguish between objective and subjective measurements. The evaluation of test cases requires measurements to ensure the attainment of numerical quality goals. For this reason, metrics are introduced to quantify different quality aspects of test cases or software artifacts in general.

Requirement 4 - Systematic Approach: The solution must be a systematic process that guides the stakeholders based on the context factors. Existing evaluation approaches define a set of measurements that apply to test cases. But, according to the context and information needs, this set of measurements might be reduced or extended. Hence, a systematic process is required that guides the stakeholders based on the test case’s context factors.

We address the previously defined requirements by developing the Test Case Quality Plan (TCQP) approach. The solution provides a systematic process which considers the context information and integrates a standardized quality model. It builds upon Model Quality Plan (MQP) (Voigt and Engels, 2008), an approach specialized for the domain of software models, and the new ISO/IEC 25010 quality standard (ISO/IEC, 2011). Our solution provides a practical guideline for the stakeholders (developers, testers and managers) for planning, implementing and using goal-oriented measurement for gaining an insight into the quality of test cases. Thus, the quality of test cases for a constantly changed software can be systematically and continuously evaluated.

The paper is structured as follows: Section 2 gives a brief overview of the related work. Then, in Section 3, we introduce the Test Case Quality Plan (TCQP) approach. In Section 4, we present our industrial case study. Section 5 presents the tooling and at the end, Section 6 concludes the work and gives an outlook on future work.

2 RELATED WORK

The work in test case quality assessment area is currently split in two main areas: approaches that identify and provide a set or a catalogs of metrics and approaches that provide a general and a standardized quality model which is applicable in any setting. The existing work which is done regarding the metrics, is predominantly considering test effectiveness (Staats et al., 2011; Mockus et al., 2009; Gopinath et al., 2014; Ellims et al., 2006; Chernak, 2001). However, there are other quality aspects for test cases besides effectiveness.

The introduction of metrics to quantify different quality aspects of test cases is considered in (Sneed, 2003). The intention of Sneed (Sneed, 2003) is to provide a set of internationally standardized and recognized test metrics from which the software development teams can select to plan their test projects and evaluate their test operation. However, these metrics do not consider the context of use of test cases. Moreover, a common definition for the quality characteristics is also missing.

Bowes et al. (Bowes et al., 2017), provide a list of 15 testing principles that represent the basis of testing goals and best practices and how they can be quantified as indicators for test case quality. However, their main focus is not on relation to an existing quality standard and consideration of the context of use.

Kaner argues in (Kaner, 2003) that it depends on the purpose how good given test cases are. According to Kaner, test cases can be "good" in different ways but it is impossible that is good at all of them. As test cases are created according different styles in different domains, a good test case in one domain is different from a good test in another domain. Thereby, Kaner clearly emphasize the importance of context of use of test cases. But, neither a relation to a quality standard, nor a systematic quality assessment approach is presented.

In general, quality models divide the term quality into its essential quality characteristics. Each of these characteristics can be subdivided into more detailed quality sub-characteristics and finally into quality attributes. In (Zeiss et al., 2007), an adaptation of the ISO/IEC 9126 quality model (ISO/IEC, 2001) to test specifications is presented. The definition for most of the characteristics are generously re-interpreted from the ISO/IEC 9126 (ISO/IEC, 2001) and applied for test specification. However, quality models do not document their assumptions about the context and it remains unclear to which degree a quality model is applicable to a given set of test cases. Moreover, the ISO/IEC 9126 (ISO/IEC, 2001) was replaced by the ISO/IEC 25010 (ISO/IEC, 2011) standard in 2011.

A well-known methodology to find appropriate metrics for an explicitly stated purpose is the GQM approach (Basili et al., 1994). GQM considers the characterization of context factors for the organization and development projects. The Model Quality Plan (MQP) approach (Voigt and Engels, 2008) specializes GQM to describe a procedure for building a quality plan for quality assessment of software models, both static and dynamic models. It combines the advantages of both GQM and quality models (Voigt et al., 2008). However, MQP does not explicitly focus on test cases and does not provide all required context factors.
3 TEST CASE QUALITY PLAN

In this section, we introduce our approach for quality evaluation of test cases, called Test Case Quality Plan (TCQP). TCQP builds upon the MQP approach (Voigt and Engels, 2008) which is relevant to the domain of software models. To apply conceptually the MQP approach in the domain of test cases, we adopted MQP by providing a new meta-model relevant to the domain of test cases. The TCQP approach consists of a top-down process, called TCQP Process (Figure 1), and a related meta-model, called TCQP meta-model (Figure 2).

The TCQP process serves as a guideline for establishing a quality plan for the quality evaluation of test cases. The TCQP meta-model contains all relevant information with respect to the quality plan. The contents of the TCQP meta-model are structured into packages which are closely linked to a respective phase of the TCQP process. In the following, we give an overview of the TCQP process.

Firstly, the Characterization of Context phase involves identifying the context information specific to a given set of test cases. Context factors are essential elements that may affect the outcome of the evaluation. Test cases are usually derived from requirement specifications, directly from the structure of a component or system or they can be also based on tester’s experience and intuition. Identifying the relevant test case context factors which include the environment, domain, and the associated artifacts, assists in selecting suitable measures for evaluating the test cases (Pfaller et al., 2008). The test case relevant context factors are defined by the Context Description Meta-model.

Secondly, any successful evaluation is performed towards an explicitly stated purpose. In the Identification of Information Needs phase, the quality goals for the evaluation of test cases are documented. By documenting the information needs, an insight necessary to manage the objectives, goals, risks and problems related to a specific quality goal of the test cases is identified and documented. As identifying the information needs is a creative process and requires a significant human resource, the context factors determined in the previous phase are used as an additional input. The Goal-Question-Metric (Basili et al., 1994) approach is used to select the insights mentioned above targeting a specific quality goal. More specifically, we utilize the goal template (Briand et al., 1996) to document the goal dimensions (object of study, purpose, quality focus, viewpoint, context) of our goals. Further, the documented goals are refined into questions. The goals and questions are specified through interviews and structured brainstorming sessions with the stakeholders. The corresponding meta-model for this step is the Information Need Meta Model and an excerpt of it is shown in Figure 3.

Third, the quality goals and their related quality focus had to be described in common terms, so that everyone who is involved in the evaluation has the same perception of the term quality. In the Definition of a Common Understanding phase, we utilize the Quality Model for Test Specification (Zeiss et al., 2007) for establishing a common quality understanding. With this general quality understanding team members would not understand quality characteristics like Usability or Test Effectivity differently. Further, the goals defined with a quality focus are mapped to quality characteristics. For the corresponding questions, quality attributes are identified and documented. As the quality model for test specifications presented in (Zeiss et al., 2007) relies on the ISO/IEC 9126 (ISO/IEC, 2001) quality standard, which was replaced in 2011 by the new ISO/IEC
have compared the differences and extended the quality model for test specifications (Zeiss et al., 2007). Doing so, this thus making it compatible with the new ISO/IEC 25010 quality standard. For example, Test Confidence was added as a new quality sub-characteristic for the Test Effectivity quality characteristic and Consistency is a new sub-characteristic for the quality characteristic Usability. Furthermore, Efficiency was renamed to Performance Efficiency. Last but not least, Security is now a separate characteristic and represents the degree to which the specified test cases have information and data that are protected and are only available depending upon the levels of authorization.

Fourth, in the Definition of Measurement phase, suitable measures are documented for the quality attributes identified in the previous phase. The measures are specified according to the ISO/IEC 15939 standard (ISO/IEC, 2002). This standard provides the Measurement Information Model (MIM) that helps in determining what has to be defined during measurement planning and evaluation (ISO/IEC, 2002). The standard distinguishes: base measures, derived measures, and indicators. The description of a measure is structured as follows: A Name and an Acronym are used to refer to a measure. The Measurement Method specifies how to compute the measure. We differentiate the two measurement types: subjective and objective. A measurement is subjective, if the quantification involves human judgment and objective, if it may be quantified automatically. The Scale and its Scale Type define possible measurement values. Base measures are computed functionally independent of other measures. In contrast, derived measures are computed as functions of two or more values of base measures. Indicators are the calculation of combining one or more base or derived measures with an associated decision criteria determined through interviews.

The TCQP meta-model, as shown in Figure 2, groups the classes in four different packages. Each of the packages contains classes specific for each of the previously introduced steps of the TCQP process. The classes from different packages are strongly interrelated with each other, thus enabling seamless transition between the process steps. For example, the imported class QualityAttribute shows transition between the last two steps. The excerpt of the Information Need Meta Model shown in Figure 3, shows the interrelation of the Information Need Meta-model with the Context Description Meta-model as well as the Quality Meta-model. In the step Characterization of Context, the ContextFactors, TestCaseType and TestItem are documented. These two context factors are used in the Identification of Information Need step for the analysis of the object of study TestSuite. Goals and Questions are used to document the information needs which are further described with respect to a quality focus. Once documented, they are used in the Definition of Quality Understanding.

The excerpt shown in Figure 3 shows just part of the meta-model. We mentioned just two ContextFactors, but there are some other relevant context factors for the domain of test cases. In the following section, we provide a concrete example, where in a tabular form we provide some additional context factors,
4 INDUSTRIAL CASE STUDY

In this section, we present a case study where we have applied and evaluated our approach. In an industrial context, due to a change of a test case management tool, test cases had to be migrated. However, the test cases created in the project were voluminous and were, unmanageable. Over time, some of these test cases became hard to understand and inefficient to execute. Hence, the team decided to evaluate and also continuously monitor the quality of the existing test cases. The main goal was to evaluate quality of natural-language test cases. Beside this requirement, the solution should also provide a mechanism that gives quality improvement suggestions for the problematic test case. Last but not least, a tooling which would support and ideally automate the evaluation process was required. To address these requirements, we have applied our TCQP approach. Due to space constraints, we use tables for the representation of the TCQP models, i.e., the concrete syntax, instead of using abstract syntax, the object diagrams.

Characterization of Context: We started with the characterization of the context according to the context factors specified in Context Description Meta-model. Using this meta-model, one can describe in which context the test cases are measured. The context factors include the environment, the domain, and the associated artifacts and they assist later in selecting suitable measures for the test cases. Table 1 shows the analyzed context factors along with the respective values. For example, the Test Object ContextFactor, contains the information about what is tested, whereas Test Case Type contains the information about the type of the test cases, either natural language-based or code-based test cases.

Table 1: Context Description Model.

<table>
<thead>
<tr>
<th>Context Factor</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Phase</td>
<td>System Requirement Specification</td>
</tr>
<tr>
<td>Test Phase</td>
<td>System Testing</td>
</tr>
<tr>
<td>Test Object</td>
<td>Automated Teller Machine</td>
</tr>
<tr>
<td>Test Item</td>
<td>Printer Module</td>
</tr>
<tr>
<td>Test Suite</td>
<td>Printer Functional Tests</td>
</tr>
<tr>
<td>Test Case Type</td>
<td>Natural Language (English)</td>
</tr>
<tr>
<td>Software Development Artif</td>
<td>System Design Document</td>
</tr>
</tbody>
</table>

Identification of Information Needs: Then, through series of interviews with two quality managers and few testers, we have specified the goal as follows: Analyze the test suite for the purpose of evaluation with respect to Usability and the context factors defined in the previous phase. Further, this goal was refined by questions in a brainstorming session with testers. For example, one specific question was regarding the completeness of the test cases, i.e., ”Has every test case its test target specified?”. At the end, each question is related to an appropriate quality attribute (Table 2). The above mentioned question was related to the quality attribute Specified Test Target.

Table 2: Questions and Quality Attributes.

<table>
<thead>
<tr>
<th>Questions</th>
<th>Quality Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has every test case its test target specified?</td>
<td>Specified Test Target</td>
</tr>
<tr>
<td>Has every test case its procedure specified?</td>
<td>Specified Procedure</td>
</tr>
<tr>
<td>Has every test case its expected results specified?</td>
<td>Specified Expected Results</td>
</tr>
<tr>
<td>Are there test cases without German Characters?</td>
<td>German Character</td>
</tr>
<tr>
<td>Are there test cases without conditional logic?</td>
<td>Conditional Logic Tests</td>
</tr>
<tr>
<td>Are there test cases without ambiguous words?</td>
<td>Ambiguous Tests</td>
</tr>
</tbody>
</table>

Common Quality Understanding: In the third step, the goal’s quality focus Usability is further refined using quality characteristics and sub-characteristics defined in the extended quality model for test specifications consistent with the ISO/IEC 25010 (ISO/IEC, 2011) quality model, thus enabling common quality understanding.

Table 3: Definition of Quality Characteristics of Test Specifications (Zeiss et al., 2007).

<table>
<thead>
<tr>
<th>Quality</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usability</td>
<td>The degree to which the specified test cases could be used with ease.</td>
</tr>
<tr>
<td>Understandability</td>
<td>The degree to which the test cases are understandable for a particular need. Documentation and description of the overall purpose of the test specification are important factors and guides in finding the suitable test cases.</td>
</tr>
<tr>
<td>Maintainability</td>
<td>The degree to which the specified test cases could be modified with ease due to changes in the software.</td>
</tr>
<tr>
<td>Analysability</td>
<td>The degree to which the test cases can be diagnosed for deficiencies. For example, test cases should be well structured to allow code reviews.</td>
</tr>
<tr>
<td>Reusability</td>
<td>The degree to which the specified test cases could be reused for different test types.</td>
</tr>
<tr>
<td>Comprehensibility</td>
<td>The degree to which the test cases are unambiguous and might not contain any conditional logic the specified test cases could be reused for different test types.</td>
</tr>
<tr>
<td>Test Effectivity</td>
<td>The degree to which the specified test cases fulfill a given test purpose.</td>
</tr>
<tr>
<td>Fault-Revealing Capability</td>
<td>The capability of the test cases to reveal faults (e.g., mutation coverage as an indicator).</td>
</tr>
</tbody>
</table>

The light gray rows in Table 3 represent the definitions of the selected quality characteristics: Usability, Maintainability, Reusability, and Test Effectivity, whereas the white rows are the associated quality sub-characteristic. For example, for the quality charac-
teristic Usability, Understandability is the associated sub-characteristic.

At the end of this step, each selected quality characteristic is related via corresponding sub-characteristics to the quality attributes defined in the previous step. For example, the quality characteristic Usability is related to the Understandability, which is further related to the quality attribute German Character.

Definition of Measurements: In the last step, measurements that quantify each quality attribute are defined and documented. We differentiate between two types of measurements, objective and subjective measurements. The measurement in Table 4 is an example of an objective measurement and it represents the number of test cases in a test suite. The formalization of the measurement done with the help of the Object Constraint Language (OCL)\(^1\) is necessary in order to enable automatic computation.

Table 4: Description of Objective Base Measure.

<table>
<thead>
<tr>
<th>Base Measure</th>
<th>Number of Test Cases in a Test Suite (NTCts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (Acronym)</td>
<td>Number of Test Cases in a Test Suite (NTCts)</td>
</tr>
<tr>
<td>Informal Definition</td>
<td>Count the number of test cases in a test suite</td>
</tr>
<tr>
<td>Formal Definition (OCL)</td>
<td>context Class::NTCts():Integer = fn:count(testProcedure[fn:matches(.,&quot;^())]))</td>
</tr>
<tr>
<td>Type of Measurement</td>
<td>Objective</td>
</tr>
<tr>
<td>Scale (Type of scale)</td>
<td>Integer from Zero to Infinity (Absolute)</td>
</tr>
</tbody>
</table>

The measurement in Table 5 on the other hand, is an example of subjective measurement. It is subjective as a human judgment is required to check if a test case is really ambiguous or not.

Table 5: Description of Subjective Base Measure (Hauptmann et al., 2013).

<table>
<thead>
<tr>
<th>Base Measure</th>
<th>Number of Ambiguous Test Cases in a Test Suite (NATCts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (Acronym)</td>
<td>Number of Ambiguous Test Cases in a Test Suite (NATCts)</td>
</tr>
<tr>
<td>Informal Definition</td>
<td>Count the number of test case procedure having ambiguous words like similar, better, similarly, worse, having in mind, take into account, take into consideration, clear, easy, strong, good, bad, useful, significant, adequate, fast, recent</td>
</tr>
<tr>
<td>Formal Definition (OCL)</td>
<td>context Class::NATCts():Integer = fn:count(testProcedure[fn:matches(.,&quot;^())]))</td>
</tr>
<tr>
<td>Type of Measurement</td>
<td>Subjective</td>
</tr>
<tr>
<td>Scale (Type of scale)</td>
<td>Integer from Zero to Infinity (Absolute)</td>
</tr>
</tbody>
</table>

The definition of measurements concludes the creation of the quality plan for the particular context and it is ready to be applied and evaluate the quality of the natural language-based system test cases.

5 TOOL IMPLEMENTATION

In this section, we present Test Case Quality Evaluator (TCQEval), a tool that we have developed for continuous monitoring of test case quality.

This tool supports the quality evaluation process by providing an environment for setup and execution of an already developed quality plan. After a quality plan is executed, the results are displayed on a dashboard as shown in Figure 4.

The tool provides insight in the quality of the test cases on three different levels of granularity: a general quality of all test cases, quality of a specific test suite and quality of a single test case. This enables the user to easily locate a test case or a group of test cases that need quality improvement, i.e., to locate those test cases with a lower score regarding specific characteristic or sub-characteristic.

The user interface of the tool, as shown in Figure 4, is split in several tabs: the first one, the Overview tab, gives a general overview of the test case quality and each of the following tabs gives an overview of a particular quality characteristic, e.g., Usability, Maintainability etc.

The Overview tab, as it names suggest, gives an general overview of the test case quality regarding all characteristics. On the left-hand side, using charts, the quality level of each quality characteristics and sub-characteristics is displayed. This graphical representation should ease the result interpretation. The Quality Assessment Summary section gives more precise information by providing the measured values of quality attributes for each respective quality characteristic and sub-characteristic. The Insights section, informs the user regarding various KPIs, defined by the quality managers in the measurement definition phase, by using different spotlight indicators like red, yellow, or green colors. This provides at-a-glance view of a particular measure’s performance. The Execution Summary informs the user about the current executions of the test cases.

For more detailed information about the quality of each separate test case regarding specific quality characteristic, e.g., usability, the respective tab should be visited, in this case the Usability tab. The test cases with some quality issues, e.g., missing test target, procedure or expected result, are properly marked. This indicates to the user what should be done to improve the quality of the test cases.

Seen from technological perspective, the tool is
a Windows Presentation Foundation (WPF)\textsuperscript{2} application. The technology selection was influenced by the industrial partner, as they had already a Microsoft technology stack in place and the existing test cases were stored in a Microsoft Access database.

6 CONCLUSION AND FUTURE WORK

In this paper, we have introduced our approach for quality evaluation of test cases called Test Case Quality Plan (TCQP) which enables a systematic and efficient development of quality plans. We have also presented an industrial case study where we have demonstrated how a quality plan can be systematically created for evaluating usability of test cases with an appropriate tool support.

TCQP builds upon Model Quality Plan and consists of a process and a corresponding meta-model. The process has the role of guiding in developing a test case quality plan, whereas the meta-model defines how a quality plan may look like. To make the basic idea applicable to the domain of test cases, we have adopted this approach to the domain of test cases by introducing a suitable meta-model. Firstly, we have introduced a new set of context factors and new questions for the information needs part. Then, we have adapted the ISO/IEC 25010 quality model to the domain of test cases and integrated it. Furthermore, we have derived appropriate measurements and metrics for test cases. Last but not least, a tool, called TCQE-val, that supports the evaluation process was implemented. Therefrom, all four requirements specified at the beginning were addressed by our work.

Regarding future work, we have defined several research directions. Firstly, the minimum set of context factors should be revisited and eventually extended. There could be other context factors that might influence the quality evaluation in a particular context. Secondly, the completeness of the quality model should be also considered. By using our approach, one can systematically extend the quality model with additional quality attributes that could be used to measure any type of test cases. The case study discussed in this paper was done in a particular in a particular context in which system test cases specified in a natural language were evaluated. However, our evaluation method provides a general solution for the development of quality plans for any test level (unit, integration and acceptance level) and not just for system level. An eventual application of our method for quality evaluation of unit test cases (e.g., jUnit test cases) would eventually mean an extension regarding the quality attributes and derivation of a proper measurements. Hence, when initially applying our approach, an initial effort due to the documentation of the required information must be expected. However, we believe that by developing a quality plan for a specific context, this initial effort will pay off by reuse for the same or a similar context. For this reason, a rule-based mechanism which will enable reuse of existing quality plans is foreseen. The basic idea is to provide rules which describe a particular context and which should help by selecting and reusing an existing qual-

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{TCQEval_Dashboard.png}
\caption{TCQEval Dashboard.}
\end{figure}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Quality Characteristics & Quality Subcharacteristics & Quality Attributes \\
\hline
Overview & Quality Maintenance & Test Effectiveness \\
\hline
Quality Characteristics & Quality Subcharacteristics & Quality Attributes \\
\hline
Execution & Test Cases & Insights \\
\hline
Quality Assessment & Quality Attributes & # Test Cases Percentage \\
\hline
Quality Characteristics & Quality Subcharacteristics & Quality Attributes \\
\hline
Insights & Quality Assessment & Execution Summary \\
\hline
Figure 4: TCQEval Dashboard.
\end{tabular}
\end{table}

\textsuperscript{2}https://docs.microsoft.com/en-us/dotnet/framework/wpf/
ity plan. By doing so, one can significantly reduce the initial effort needed to create a quality plan.

REFERENCES


