INTERNATIONAL STANDARD

ISO/IEC/ IEEE 29119-5

First edition
2016-11-15

Software and systems engineering — Software testing —

Part 5: Keyword-Driven Testing

Ingénierie du logiciel et des systèmes — Essais du logiciel —

Partie 5: Essais axés sur des mots-clés

Reference number
ISO/IEC/IEEE 29119-5:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

IEC Central Office 3,
rue de Varembé CH-
1211 Geneva 20
Switzerland
E-mail inmail@iec.ch
Web www.iec.ch

Institute of Electrical and Electronics Engineers, Inc
3 Park Avenue, New York
NY 10016-5997, USA

stds.ipr@ieee.org
www.ieee.org

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of ISO/IEC JTC 1 is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is called to the possibility that implementation of this standard may require the use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from ISO or the IEEE Standards Association.

ISO/IEC/IEEE 29119-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in cooperation with the Software & Systems Engineering Standards Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

ISO/IEC/IEEE 29119 consists of the following parts, under the general title *Software and systems engineering — Software testing:*

— *Part 1: Concepts and definitions*

— *Part 2: Test processes*

— *Part 3: Test documentation*

— *Part 4: Test techniques*

— *Part 5: Keyword-Driven Testing*

Introduction

The purpose of the ISO/IEC/IEEE 29119 series of software testing standards is to define an internationally-agreed set of standards for software testing that can be used by any organization when managing or performing any form of software testing.

This part of ISO/IEC/IEEE 29119 defines a unified approach for describing test cases in a modular way, which assists with the creation of items like keyword-driven test specifications and test automation frameworks. The term "keyword" refers to the elements which are, once defined, used to compose test cases, such as with building blocks. ISO/IEC/IEEE 29119-5 will explain the main concepts and application of Keyword-Driven Testing. It will also define attributes of frameworks designed to support Keyword-Driven Testing.

The concepts and definitions relating to software testing defined in ISO/IEC/IEEE 29119-1 are also applicable to ISO/IEC/IEEE 29119-5.

The test process model on which the Keyword-Driven Testing framework is based is defined in ISO/IEC/IEEE 29119-2 Test Processes. It comprises test process descriptions that define the software testing processes at the organizational level, test management level and dynamic test level. Supporting informative diagrams describing the processes are also provided in ISO/IEC/IEEE 29119-2. However, ISO/IEC/IEEE 29119-5 describes a specific implementation of the test design and implementation processes of ISO/IEC/IEEE 29119-2; in particular TD4 (Derive Test Cases), TD5 (Assemble Test Sets) and TD6 (Derive Test Procedures) as here applied to Keyword-Driven Testing.

The templates and examples of test documentation as defined in ISO/IEC/IEEE 29119-3 are also applicable to ISO/IEC/IEEE 29119-5.

Software test design techniques that can be used during test design are defined in ISO/IEC/IEEE 29119-4 Test Techniques. The application of ISO/IEC/IEEE 29119-4 is assumed when designing test cases that are then described by keywords according to ISO/IEC/IEEE 29119-5.

This part of ISO/IEC/IEEE 29119-5 has the following structure:

— terms and definitions can be found in clause 4

— an introduction to Keyword-Driven Testing is given in clause 5

— the application of Keyword-Driven Testing is explained in clause 6

— frameworks for Keyword-Driven Testing are described in clause 7

— data interchange is covered in clause 8

— Annex A states naming conventions for keywords

— Annex B names benefits that can be achieved with Keyword-Driven Testing

— Annex C gives advice on how interested parties wanting to use Keyword-Driven Testing can start

— Annex D describes roles that can be used in Keyword-Driven Testing

— Annex E contains examples of basic keywords that can be used to create test cases

— Annex F contains examples for keyword test cases

Software and Systems Engineering — Software Testing — Part 5: Keyword-Driven Testing

1 Scope

This part of ISO/IEC/IEEE 29119 defines an efficient and consistent solution for Keyword-Driven Testing by:

— giving an introduction to Keyword-Driven Testing;

— providing a reference approach to implement Keyword-Driven Testing;

— defining requirements on frameworks for Keyword-Driven Testing to enable testers to share their work items, such as test cases, test data, keywords, or complete test specifications;

— defining requirements for tools that support Keyword-Driven Testing. These requirements could apply to any tool that supports the Keyword-Driven approach (e.g., test automation, test design and test management tools);

— defining interfaces and a common data exchange format to ensure that tools from different vendors can exchange their data (e.g. test cases, test data and test results);

— defining levels of hierarchical keywords, and advising use of hierarchical keywords. This includes describing specific types of keywords (e.g. keywords for navigation or for checking a value) and when to use "flat" structured keywords;

— providing an initial list of example generic technical (low-level) keywords, such as "inputData" or "checkValue". These keywords can be used to specify test cases on a technical level, and may be combined to create business-level keywords as required.

NOTE This standard is applicable to all those who want to create keyword-driven test specifications, create corresponding frameworks, or build test automation based on keywords.

2 Conformance

2.1 Intended usage

The requirements in ISO/IEC/IEEE 29119-5 are contained in Clause 7 and in Annex A. ISO/IEC/IEEE 29119-5 provides requirements on frameworks supporting the application of Keyword-Driven Testing. It is recognized that particular projects or organizations may not need to use all of the components defined in this standard. Therefore, implementation of ISO/IEC/IEEE 29119-5 typically involves selecting a set of components or parts of components suitable for the organization or project. There are two ways that an organization can claim to conform to the provisions of this standard.

The organization or individual shall assert whether full or tailored conformance to this standard is claimed.

2.2 Full conformance

Full conformance is achieved by demonstrating that all of the Keyword-Driven Testing requirements (i.e. shall statements) defined in ISO/IEC/IEEE 29119-5 have been satisfied.

2.3 Tailored conformance

When ISO/IEC/IEEE 29119-5 is used for implementing components of frameworks that do not qualify for full conformance, the subset of components for which tailored conformance is claimed should be recorded. Tailored conformance is achieved by demonstrating that all of the requirements (i.e. shall statements) for the recorded subset of components have been satisfied.

Where tailoring occurs, the justification shall be provided, either directly or by reference, whenever a requirement defined in clauses 7 and Annex A of ISO/IEC/IEEE 29119-5 is not followed. All tailoring decisions shall be recorded with their rationale, including the consideration of any applicable risks. Tailoring decisions shall be agreed to by the relevant stakeholders.

EXAMPLE Tool vendors may in their portfolio provide only part of a keyword-driven test framework, and thus decide not to implement requirements that are covered by complementary tools (e.g. a vendor only provides an execution engine, but no keyword driven editor – then the execution engine can still be conforming with the standard).

3 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document, including any amendments, applies.

ISO/IEC/IEEE 24765, *Systems and software engineering – Vocabulary*

ISO/IEC/IEEE 29119-1, *Software and systems engineering – Software Testing – Part 1: Concepts and Definitions*

ISO/IEC/IEEE 29119-2, *Software and systems engineering – Software Testing – Part 2: Test Processes*

ISO/IEC/IEEE 29119-3, *Software and systems engineering – Software Testing – Part 3: Test Documentation*

ISO/IEC/IEEE 29119-4, *Software and systems engineering – Software Testing – Part 4: Test Techniques*

Other standards useful for the implementation and interpretation of this standard are listed in the bibliography.

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC/IEEE 24765 and the following apply.

NOTE Use of the terminology of ISO/IEC/IEEE 29119-5 is for ease of reference and is not mandatory for conformance with ISO/IEC/IEEE 29119-5. The following terms and definitions are provided to assist with the understanding and readability of ISO/IEC/IEEE 29119-5. Only terms critical to the understanding of ISO/IEC/IEEE 29119-5 are included. This clause is not intended to provide a complete list of testing terms. The Systems and Software Engineering vocabulary ISO/IEC/IEEE 24765 can be referenced for terms not defined in this standard. ISO/IEC/IEEE 29119-1 can be referenced for terms related to software testing in general. ISO/IEC/IEEE 29119-5 only defines terms specific to Keyword-Driven Testing.

4.1
domain layer
highest level of abstraction for the test item

Note 1 to entry: Keywords on this level are chosen in a way that is familiar to domain experts.

4.2
high-level keyword
keyword that covers complex activities that may be composed from other keywords and is used by domain experts to assemble keyword test cases

4.3
keyword
one or more words used as a reference to a specific set of actions intended to be performed during the execution of one or more test cases

Note 1 to entry: The actions include interactions with the User Interface during the test, verification, and specific actions to set up a test scenario.

Note 2 to entry: Keywords are named using at least one verb.

Note 3 to entry: Composite keywords can be constructed based on other keywords.

4.4
keyword dictionary
keyword library
repository containing a set of keywords reflecting the language and level of abstraction used to write test cases

4.5
Keyword-Driven Testing
testing using test cases composed from keywords

4.6
Keyword-Driven Testing framework
test framework covering the functional capabilities of a keyword-driven editor, decomposer, data sequencer, manual test assistant, tool bridge, data and script repositories, a keyword library and the test execution environment

4.7
keyword execution code
implementation of a keyword that is intended to be executed by a test execution engine

4.8
keyword test case
sequence of keywords and the required values for their associated parameters (as applicable) that are composed to describe the actions of a test case

4.9
low-level keyword
keyword that covers only one or very few simple actions and is not composed from other keywords

4.10
manual testing
humans performing tests by entering information into a test item and verifying the results

Note 1 to entry: Automated testing uses tools, robots, and other test execution engines to perform tests. Manual testing does not use these items.

4.11
test execution engine
tool implemented in software and sometimes in hardware that can manipulate the test item to execute test cases

Note 1 to entry: A typical test execution engine includes unit test tool frameworks, stimulation-command systems, capture and playback tools or hardware robots along with the software to control them.

4.12
test framework
environment that facilitates testing

4.13
test interface
interface to the test item used to stimulate the test item, to get responses (e.g. actual results), or both

Note 1 to entry: The GUI, API or SOA interfaces are typical test interfaces.

Note 2 to entry: Stimulating the test item can involve passing data into it via computer interfaces or attached hardware.

Note 3 to entry: Getting responses includes getting information from the test item under test or associated hardware.

4.14
test interface layer
lowest level of abstraction for keywords, which interacts with the test item directly and encapsulates the atomic (lowest level) interactions at the test interface

5 Introduction to Keyword-Driven Testing

5.1 Overview

Keyword-Driven Testing is a test case specification approach that is commonly used to support test automation and the development of test automation frameworks. However, it can also be used if no automation approach is planned or established.

In principle, Keyword-Driven Testing can be applied at all testing levels (e.g. component testing, system testing) and for various types of testing (e.g. functional testing, reliability testing). Keyword-Driven Testing benefits include the following:

— ease of use

— understandability

— maintainability

— test information reuse

— support of test automation

— potential cost and schedule savings

The fundamental idea in Keyword-Driven Testing is to provide a set of "building blocks", referred to as keywords, that can be used to create manual or automated test cases without requiring detailed knowledge of programming or test tool expertise. The ultimate goal is to provide a basic, unambiguous set of keywords comprehensive enough so that most, if not all, required test cases can be entirely composed of these keywords. The vocabulary included in these dictionaries or libraries of keywords is, therefore, a reflection of the language and level of abstraction used to write the test cases, and not of any standard computer programming language.

For test automation, each keyword needs to be implemented in software.

NOTE 1 When keywords are not used, test cases are usually written using natural language or written in a computer programming language. Compared with natural language, keywords have the advantage of being less ambiguous and more precise. Compared with a computer programming language, when keywords are well defined and structured, they have the advantage of being understandable by many people who do not have software engineering skills.

A keyword is usually defined at the following two levels:

— At a low level, each keyword is associated with a detailed set of one or more actions that describe the exact steps that are to be performed.

— At a high level, a meaningful name is used to identify the keyword. This keyword may require a set of input parameters, which would also belong to this level in the structure. The keyword and the parameters together comprise a high-level description of the actions associated with a test case.

Thus, instead of describing each individual action in test cases, tests can be defined at a higher level of abstraction using keywords. Higher levels of abstraction can be achieved by using composite keywords, which are comprised of other keywords to describe associated actions.

An example of the benefits obtained from both manual and automated keyword-driven test cases is enhanced maintainability. Consider a case where the precise set of actions to carry out a commonly repeated operation has changed. The modularity introduced by keywords allows modification of only the actions for the changed operation in the relevant lower-level keyword, leaving the test cases untouched. Without modularity, it may be necessary to modify each occurrence of this operation in all of the test cases.

Modularization has helped popularize this approach. If test automation is required, a framework can be created to interpret manually created keyword test cases as executable test automation scripts. This is achieved by implementing test automation code for each keyword (e.g. keyword execution code).

NOTE 2 Testing tools can be used to support Keyword-Driven Testing, but the available tools may be limited in their capability to support all the concepts described in this standard.

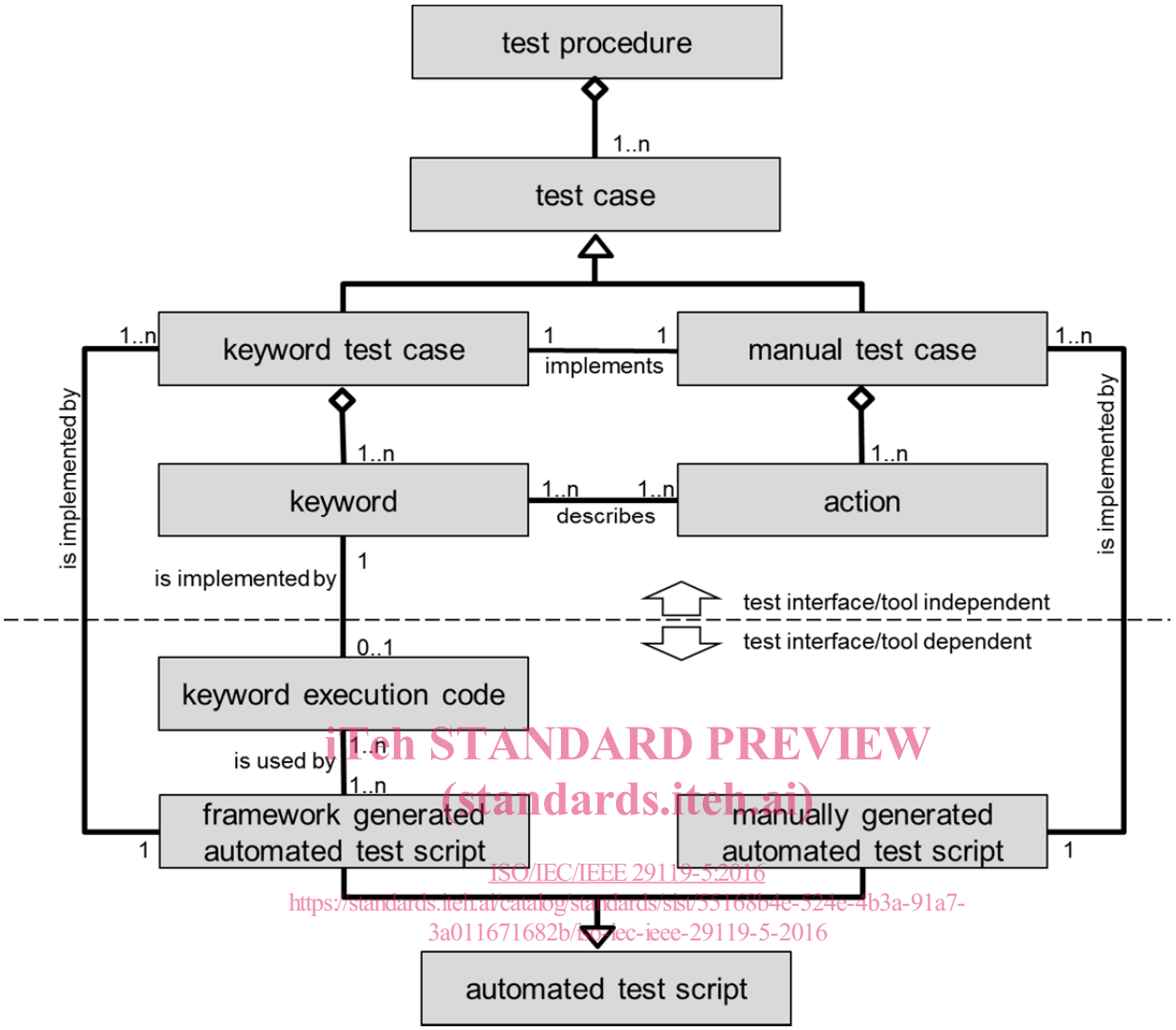

Figure 1 — Relationships between Keyword-Driven Testing entities

A test procedure can have multiple test cases in it, and a test case can, in turn, be part of different test procedures (Figure 1). Test cases can be either manual test cases or keyword test cases. A keyword test case implements a manual test case.

A keyword test case is typically composed of a sequenced series of keywords. Keywords should be chosen to be modular and generic so that they can be reused in many test cases. Keywords can also be used more than once in the same test case. A test case is composed from test actions. Keywords represent test actions.

NOTE 3 It is possible to map several keywords to a single action. It is also possible to define keywords in a way that each keyword represents one action. In these cases, a one-to-one relationship exists between actions and keywords. However, a test designer can decide to structure keywords in a different way (e.g. use more than one keyword to implement an action, or to combine two or more actions into one keyword). This relationship is not a 1:1 relationship in Figure 1.

Test automation is an option that can be chosen when implementing Keyword-Driven Testing, but a manual approach is also possible. If keyword test cases are automated, each keyword is implemented by keyword execution code. Keyword execution code is specific to the chosen tool or test execution engine, and will additionally depend on the test interface. For the manual approach, the action described by a keyword is executed manually, so there is no keyword execution code. That is why in Figure 1 the relationship between keyword and keyword execution code is 1 to 0..1.

Test automation is typically highly technical and tool dependent since it depends on the test interface and on the capabilities of the available tools. In general, keywords can be independent of the test interface (e.g. user interface) and the tools used to execute the test cases.

In this context, automated test scripts may either be generated automatically by a framework or developed manually by a test automation specialist. Automated test scripts are typically developed by testers with programming experience.

NOTE 4 When developing automated test scripts, it is beneficial to align the structure (e.g. levels) of the keywords with the implementation of the automated test scripts.

If a keyword test case or a set of keyword test cases is automated, the framework for Keyword-Driven Testing generates the automated test script based on the keyword execution code.

NOTE 5 A framework for Keyword-Driven Testing does not necessarily "generate" code. The required code can also be prepared by testers and be executed by the framework.

5.2 Layers in Keyword-Driven Testing

5.2.1 Overview

Keywords can represent actions at different abstraction levels. For example, one keyword can refer to a very complex set of activities, like the creation of a contract, which includes a lot of steps, while another keyword can refer to a very simple action, like pressing a button on a graphical user interface. The first keyword is close to the business and end user domain, while the second is closer to the test interface. Keywords that are written at a similar level of detail, and have a similar relationship to the stakeholder's view, are said to belong to the same abstraction layer.

Keyword-Driven Testing can be organized by using one or more layers. Typical layers are the end user domain layer and the test interface layer.

While many implementations of Keyword-Driven Testing will comprise two or three abstraction layers, in some cases it may be necessary to structure keywords in more layers.

The topmost layer is the most abstract layer, which is generally aligned with the wording of the application's users. In practice, the topmost layer is usually the domain layer. However, in some situations the domain layer may not be required, and another, more abstract layer is used (e.g. if the test cases are supposed to span several different end user domains, a meta domain layer can be introduced).

The lowest layer is the most detailed layer. It is commonly aligned with the names of test interface elements (e.g. "selectMenuItem"). In practice, this layer is usually, but not always, the test interface layer (e.g. as sometimes a test interface layer is not required, or for specific reasons, even more detailed layers may be used).

Most Keyword-Driven Test systems will have more than one layer due to factors such as having understandable keyword test cases, maintainability and division of work relying on a multi-layer structure. If only one layer is implemented, it will commonly be either at a low level, which affects the readability of the keyword test cases, or at a high level, which can result in more keyword execution code.

In Figure 2, an example is given, showing how test cases for two different test interfaces can be structured by using two layers of keywords.

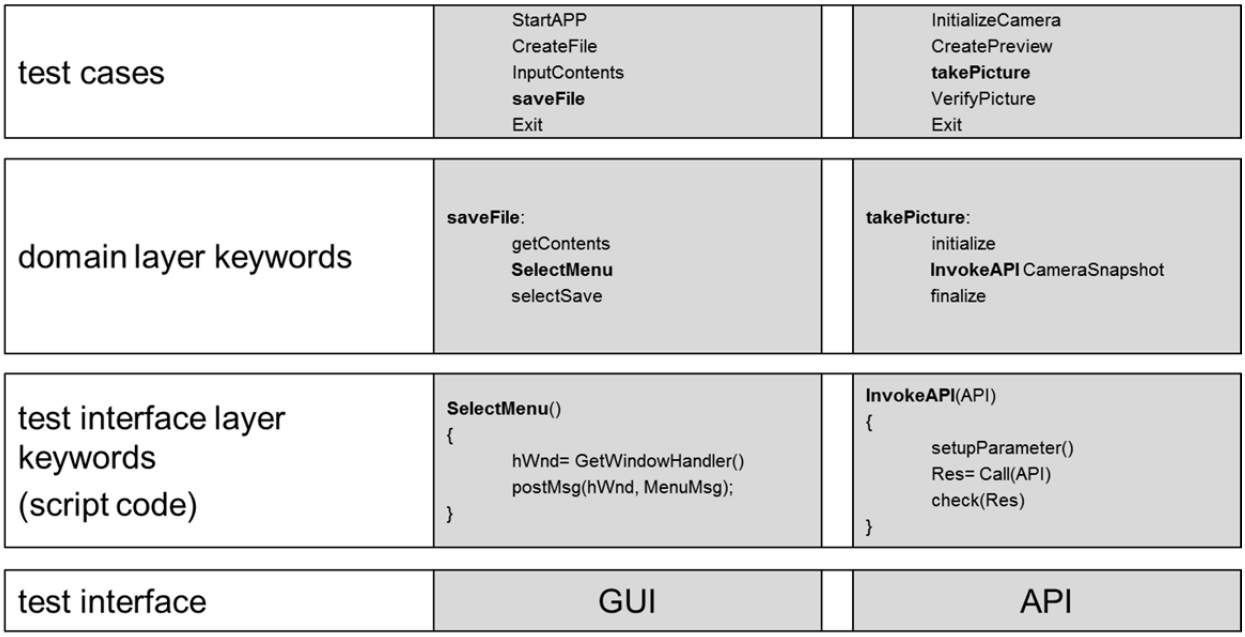

Figure 2 — Example for defining test cases by keywords at several layers

EXAMPLE In Figure 2, two test cases are shown which are designed using a domain layer and a test interface layer. One of the examples sketches a test case for a GUI application, the other for a camera API. In both examples, the implementation of the test cases in respect to test automation is done on the test interface layer. From top to bottom, the example shows a test case for each test item, shows how one of the used composite keywords on the domain layer for both test items could be structured, and gives an idea of how the implementation of one of the basic keywords on the test interface layer for each test item could look.

5.2.2 Domain layer

Keywords in the domain layer correspond to business or domain related activities and reflect the terminology used by domain experts. Because of this, it can be easier for testers at the domain or business level to create test cases.

Keywords developed for the domain layer are generally implementation-independent; that is, the keywords define tests that work regardless of the technology used to implement the test item.

EXAMPLE 1 Consider a keyword test developed to test a word processing application. Domain layer keywords correspond to the activities that are part of the “business of word processing”:

StartApplication <app_name>

ClearBuffer

EnterText “Hello World!”

ReplaceText “Hello”, “Goodbye”, “ALL_OCCURRENCES”

VerifyText “Goodbye World!”

StopApplication

This test is valid for any text editor application that provides a global replace function, (e.g. Notepad, MSWord, Notepad++, GED, EMACS, etc.).

EXAMPLE 2 Frequently used domain layer keywords are "Login" and "CreateAccount"

Tests constructed using domain layer keywords are relatively immune to changes in the implementation of the test item, and can prevent expensive rework over the lifetime of the test item.

NOTE Extensive changes to the application, (e.g. changes in the workflow) can require test cases to be reworked.

5.2.3 Test interface layer

Keywords at the test interface layer refer to a specific type of test interface, (e.g. the graphical user interface (GUI)). The actions needed to address the test items can usually be easily identified. The total number of keywords is typically smaller than at the domain layer, since the test interface is limited.

EXAMPLE 1 A GUI can be used as the test interface. As the GUI controls (along with the associated actions) are mapped to a fixed set of keywords, a small number of keywords is needed. In the same case the domain-related keywords can be very versatile, and may need to be extended according to the needs of the tester, which leads to a much bigger number of keywords.

If automation is desired, the keyword execution code for keywords at the test interface layer is often simpler. However, for a keyword test case composed from keywords at the test interface layer, it may be difficult to see how the interface layer keywords are related to the business domain.

Interface layer keywords usually reflect the underlying implementation technology for the interaction with the test item. For example, keywords such as MenuSelect and PressButton reflect a GUI operation. Using the example above, they would not be applicable to text editors using a command line interface, such as vi, because they correspond to window-based operations.

EXAMPLE 2 Consider a test interface that is a graphical user interface. Keywords are chosen to cover single actions such as "Click" or "Select". These keywords are applied to different elements like lists, grids, or images, and the specific element can be selected by using the keyword with a parameter (See 5.4 Keywords and Data). Some combinations of actions and elements can be excluded.

5.2.4 Multiple layers

To combine the advantages of several layers (e.g. domain layer and test interface layer), a framework is required, which can help manage hierarchical keywords (see section 7 for details about testing frameworks). This way a high-level keyword at the business level (e.g. domain layer keyword), can be built from several lower level keywords at a more technical level (e.g. test interface layer keywords).

Figure 3 illustrates how multiple layers can be used in Keyword-Driven Testing.

In complex settings, three or more layers of keywords are necessary. In the figure, additional intermediate layers are represented by three dots "…".

Using multiple layers requires composite keywords (see 5.3.2 Composite keywords).