TECHNICAL REPORT

ISO/IEC TR 29119-6

First edition

Software and systems engineering — Software testing —

Part 6:
Guidelines for the use of ISO/IEC/IEEE 29119 (all parts) in agile projects

PROOF/ÉPREUVE

Reference number
ISO/IEC TR 29119-6:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

A list of all parts in the ISO/IEC/IEEE 29119 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

The purpose of ISO/IEC/IEEE 29119 (all parts) is to define an internationally agreed set of standards for software testing that can be used by any organization when performing any form of software testing.

This document facilitates understanding of how ISO/IEC/IEEE 29119 (all parts) applies to agile life cycles.

ISO/IEC/IEEE 29119-1 introduces software testing concepts and vocabulary. This document uses the concepts and vocabulary of ISO/IEC/IEEE 29119-1.

ISO/IEC/IEEE 29119-2 comprises test process descriptions that define the software testing processes at the organizational level, test management level and dynamic test levels. It supports dynamic testing, functional and non-functional testing, manual and automated testing and scripted and unscripted testing, and can be utilized within any lifecycle model, including agile lifecycles and methodologies.

ISO/IEC/IEEE 29119-3 defines software test documentation. The requirements specified for templates and examples of test documentation defined in ISO/IEC/IEEE 29119-3 can be met in standard or tailored agile lifecycles and methodologies.

ISO/IEC/IEEE 29119-4 defines test design techniques, which can be utilized in any lifecycle, including agile.

ISO/IEC/IEEE 29119-5 addresses the use of keywords to support automated testing.

This document provides a mapping of agile concepts to ISO/IEC/IEEE 29119-2. It also explains how ISO/IEC/IEEE 29119-2 can be adopted under specific agile methodologies and demonstrates how the test documentation templates defined in ISO/IEC/IEEE 29119-3 can be implemented in agile lifecycles.

Clause 4 maps agile practices and artefacts to corresponding clauses of ISO/IEC/IEEE 29119-2. Annex A provides a mapping from The Scrum Guide[6] to ISO/IEC/IEEE 29119-2 clauses. Annex B provides a mapping from all clauses of ISO/IEC/IEEE 29119-2 to the agile practices and artefacts covered under Clause 4. Annex C provides an example mapping of typical test artefacts used in agile to ISO/IEC/IEEE 29119-3. Annex D provides examples of agile test artefacts and explains how they comply with ISO/IEC/IEEE 29119-3.

Software and systems engineering — Software testing —

Part 6: Guidelines for the use of ISO/IEC/IEEE 29119 (all parts) in agile projects

1 Scope

This document provides guidance for the application of ISO/IEC/IEEE 29119 (all parts) in agile life cycles. This document is intended for (and not limited to) testers, test managers, business analysts, product owners, Scrum masters and developers involved in agile projects. The mappings provided in this document are designed to benefit any team or organization that is either moving away from traditional/waterfall life cycles and into agile or vice versa as well as new organizations that are commencing agile as their chosen life cycle. It is designed to be understandable regardless of the reader's familiarity with ISO/IEC/IEEE 29119 (all parts).

2 Normative references

There are no normative references in this document.

3 Terms and definitions

No terms and definitions are listed in this document.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

NOTE For terms and definitions in the field of systems and software engineering, see ISO/IEC/IEEE 24765, which is published periodically as a "snapshot" of the SEVOCAB (Systems and software Engineering Vocabulary) database and is publicly accessible at www.computer.org/sevocab.

4 Concepts

4.1 Agile practices and artefacts

This document explains how ISO/IEC/IEEE 29119 (all parts) can be adopted for testing in products, projects, teams or organizations that have adopted agile methodologies (referred to in this document as "agile testing"). The aim is to assist users of the processes and documentation templates defined in ISO/IEC/IEEE 29119-2 and ISO/IEC/IEEE 29119-3 in agile life cycles.

Agile is an approach to software and systems development whereby requirements and systems evolve over time via the collaboration and communication of self-organizing cross-functional teams, with regular feedback from end-users, supporting a rapid and flexible response to requirement change. Example agile methodologies include Scrum, SAFe and eXtreme Programming (XP), within which a wide variety of agile practices and artefacts exist. The agile practices and artefacts listed in Table 1 are covered in this document. These agile practices and artefacts include many that are utilized during testing. Some of these practices and artefacts might not be part of a specific agile methodology such

as Scrum. This document explains how each practice and artefact maps to the concepts covered in ISO/IEC/IEEE 29119-2 and ISO/IEC/IEEE 29119-3.

Table 1 — Agile practices and artefacts covered in this document

Acceptance criteria	Feature toggle	Retrospective meeting (a.k.a. Sprint retrospective)
Acceptance test-driven development	Frequent interaction with product owner	Scrum master
Amplify learning	Increment (a.k.a. product increment)	Self-organizing teams
Backlog review (a.k.a. backlog maintenance, backlog refinement)	Informal defect management	Short iterations
Behaviour-driven development (a.k.a. specification by example)	Iteration backlog (a.k.a. sprint backlog, sprint catalogue, iteration backlog)	Simplicity
Build integrity in	Iteration goal (a.k.a. sprint goal)	Story card
Burn-down and burn-up chart	Iteration planning (a.k.a. sprint planning) and release planning (a.k.a. phase planning, stage planning)	Story mapping
Co-located teams	Iteration review (a.k.a. demo, sprint review, iteration review)	Story testing
Collective code ownership	Iteration zero (a.k.a. sprint zero)	Sustainable pace (a.k.a. 40-hour week)
Continuous delivery and deployment	Just in time (a.k.a. decide as late as possible)	Task board (a.k.a. Scrum board, Kanban board)
Continuous integration and continuous testing	Limit work in progress (WIP)	Team charter (a.k.a. project charter)
Continuous process improvement	Mood chart (a.k.a. niko-niko)	Team room
Cross-functional team	Occasional test iterations	Team-based estimation
Daily stand-up (a.k.a. daily Scrum, frequent stand-up)	Pair programming	Technical debt
Definition of done	Parallel test iterations	Test-driven development
Definition of ready	Planning poker	Timebox (a.k.a. timeboxed iterations)
Eliminate waste	Product backlog (a.k.a. backlog)	Transparency (a.k.a. visibility of project progress)
Empowered team	Product owner	User stories – INVEST mnemonic
Emergent design	Refactoring	User story
Epic	Relative estimation	User story format – role/feature/rationale
Fast user feedback		Velocity
Feature-driven development		

Explanations of how these agile practices and artefacts can be used in the test processes of ISO/IEC/IEEE 29119-2 are provided in 4.2. An example mapping of each concept to Scrum is provided in Annex A. A mapping of Scrum to ISO/IEC/IEEE 29119-2 is provided in Annex B.

Some of the same test documentation artefacts that are created as in traditional projects are also often created in agile, such as test policy, organizational test practices and test plan, though typically with significantly less detail and sometimes in an alternative format. An example is provided in Annex C demonstrating how test documentation that is often used in agile maps to ISO/IEC/IEEE 29119-3.

4.2 Mapping of agile practices to ISO/IEC/IEEE 29119-2 test processes

4.2.1 Overview

This subclause demonstrates how the agile practices and artefacts listed in Table 1 map to the test processes defined in ISO/IEC/IEEE 29119-2. Each mapping is presented in Tables 2 to 27.

4.2.2 Acceptance criteria

Acceptance criteria are the conditions that a user story needs to satisfy for it to be considered "done" (e.g. accepted by the product owner, customer, user or other stakeholders).

Table 2 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Acceptance criteria	8.2.4.2 Create test model	Acceptance criteria would be created as a test model that lists atomic requirements to be met before a user story can be accepted as “done.”

4.2.3 Acceptance test-driven development (ATDD)

Acceptance test-driven development (ATDD) is a test-first approach whereby an agile team creates acceptance tests, to verify that acceptance criteria of each user story are met, before the code that passes those tests is written. ATDD is a form of test-driven development (TDD) at the acceptance testing level.

Table 3 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping Details
Acceptance test-driven development	All clauses	When conducting ATDD, all clauses of ISO/IEC/IEEE 29119-2 apply. The requirement to conduct ATDD can be included in the test policy, and the chosen approach to ATDD would be described in detail in the organizational test practices. The requirement to use ATDD on a given product would be mentioned in the test plan for that product. Tests can be designed and executed under the test design and execution process, and the environment and test data requirements would be covered under the environment and data management process. Testing can be monitored, controlled and reported on under the monitor and control activity. Defects detected during testing can be raised under the incident reporting process. Once testing is complete, assets can be archived, and the outcomes of testing reported, under the test completion process.

4.2.4 Amplify learning

Amplify learning is a concept of creating a team environment that encourages learning through various approaches including through trial and error (i.e. accepting and learning from failure), highlighting the fact that systems development is a continual learning process. Although this can be supported by implementing ISO/IEC/IEEE 29119-2 iteratively (which is supported by the standard), there is no specific concept in the standard that maps to this.

4.2.5 Backlog management

It is common for all members of the agile team to regularly review user stories, including testers. This can include reviewing acceptance criteria to ensure they are complete and testable. Backlog reviews are also referred to as "backlog maintenance" and "backlog refinement." During backlog management, it is common for testers to review the backlog from a testing perspective, to estimate testing effort for each user story and to prioritise testing efforts on each user story.

Table 4 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Backlog management	8.2.4.2 Create test model	Acceptance criteria would be created as a test model that lists atomic requirements to be met before a user story can be accepted as "done."
	7.2.4.4 Identify and analyse risks	Project and product risks can be considered during user story prioritization, would be identified via the identify and analyse risks activity.
	7.2.4.8 Record test plan > task a)	Test estimates are refined during each iteration (e.g. during iteration planning or backlog refinement sessions) via the record test plan activity, task a).
	8.2 Test design and implementation process: — 8.2.4.2 Create test model > task e); — 8.2.4.3 Identify test coverage items > task b); — 8.2.4.4 Derive test cases > task b); — 8.2.4.5 Create test procedures > task c)	The priority of each user story would be used to prioritize test design, via the various test design activities in the dynamic test processes.

4.2.6 Behaviour-driven development (BDD)

Behaviour-driven development (BDD) is a form of test-driven development, where development and testing are based on the design of acceptance criteria that are written in a "given/when/then" format.

EXAMPLE An example of a given/when/then statement is:

— given I am at the Login screen;

— when I enter a valid username and matching password;

— then I am logged into the system.

By writing tests that describe the behaviour of each feature and its underlying requirements before the code is written, it assists teams with designing better quality systems. The approach is also designed to encourage greater communication and collaboration between developers, testers and business analysts/representatives.

In BDD, an agile team write one or more given/when/then statements that elaborate a user story, storing them as acceptance criteria on the story. A developer then writes unit tests that cover each of the acceptance criteria, before writing code that passes each unit test. Such unit tests are typically automated, with specialist tools available to support it. This approach is similar to TDD, except that the unit tests are designed at the acceptance level, as they are designed to demonstrate that the new code passes the acceptance criteria on the story.

Behaviour-driven development is also known as specification by example.

Table 5 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Behaviour-driven development	All clauses	When conducting BDD, all clauses of ISO/IEC/IEEE 29119-2 apply. The requirement to conduct BDD can be included in the test policy, and the chosen approach to BDD would be described in detail in the organizational test practices. The requirement to use BDD on a given product would be mentioned in the test plan for that product. Tests can be designed and executed under the test design and execution process, and the environment and test data requirements would be covered under the environment and data management process. Testing can be monitored, controlled and reported on under the monitor and control activity. Defects detected during testing can be raised under the incident reporting process. Once testing is complete, assets can be archived, and the outcomes of testing reported, under the test completion process.

4.2.7 Build integrity in

Building integrity in is a concept from lean software development, which includes:

— conceptual integrity: how well the system's components work together as a whole;

— perceived integrity: how the system is perceived by stakeholders outside the agile team, which can be improved by including customers and users in user acceptance testing.

Table 6 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Build integrity in – conceptual integrity	7.2 Test strategy and planning process	Integration and system testing are used to verify how well a system's components work together as a whole. Test planning for integration testing and system testing would be carried out under the test strategy and planning process.
	7.3 Test monitoring and control process	Progress towards meeting each test plan would be monitored under the test monitoring and control process.
	8 Dynamic test processes	Test design and execution for integration testing and system testing would be carried out under the dynamic test processes.
	7.4 Test completion process	The outcomes of testing would be evaluated and reported under the test completion process.
Build integrity in – perceived integrity	7.2 Test strategy and planning process	Agile teams typically verify how a system is perceived by external stakeholders via user acceptance testing. Test planning for user acceptance testing would be carried out under the test strategy and planning process.
	7.3 Test monitoring and control process	Progress towards meeting the test plan would be monitored under the test monitoring and control process.
	8 Dynamic test processes	Test design and execution for user acceptance testing would be carried out under the dynamic test processes.
	7.4 Test completion process	The outcomes of testing would be evaluated and reported under the test completion process.

4.2.8 Burn-down and burn-up charts

Burn-down and burn-up charts are used to track and measure progress in agile projects and can be used for releases or iterations. They illustrate the number of user stories "done" in an iteration or

release at a given point in time. Since the "definition of done" usually requires all tests to pass, burn-down and burn-up charts provide information on test progress.

Table 7 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Burn-down and burn-up charts	7.3.4.3 Monitor > task a)	Testers contribute to burn-down and burn-up charts by executing test cases that verify whether each user story's acceptance criteria have been met. This daily test execution progress would be tracked via the monitor (TMC2) activity, task a, which focuses specifically on collecting metrics to track progress towards completion.
	7.3.4.5 Report	The status of testing, including progress towards testing on each user story, would be reported via the report activity.

4.2.9 Co-located teams

In co-located agile teams, all members of the teamwork at the same physical location, enhancing team communication and trust. ISO/IEC/IEEE 29119-2 does not place any requirements on the location of team members, thus this concept does not map directly to the standard.

4.2.10 Collective code ownership

The concept of collective code ownership is that all agile team members take collective responsibility for the product (i.e. system), including testing and quality. This also means that any team member can be responsible for making code changes, even if they did not originally write that part of the code.

ISO/IEC/IEEE 29119-2 does not place any requirements on which role (e.g. developer, tester) can implement each process in the standard. Any agile team member can implement any process, activity or task within the standard, including test management, test design, test execution and reporting. Thus, the concept of collective code ownership does not map directly to the standard.

4.2.11 Continuous delivery and deployment

Continuous delivery is the process of building systems that can be delivered to production as often as required (e.g. daily, weekly, fortnightly). Continuous delivery typically involves low levels of human intervention in the build and deployment process.

Continuous deployment is fully automated deployment with no human intervention, with products automatically released to test or production environments, sometimes on a frequent basis.

Continuous delivery and deployment typically require automated testing, to confirm that systems are functioning correctly after build and before deployment, where tests that fail prevent new changes from being deployed to production.

Table 8 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Continuous delivery and deployment	All clauses	When conducting continuous delivery and deployment, automated testing is typically required, to confirm that all systems are behaving correctly after build and after deployment. For automated testing, all clauses of ISO/IEC/IEEE 29119-2 apply. The requirement to conduct automated testing can be included in the test policy, and the chosen approach to automated testing would be described in detail in the organizational test practices. The requirement to use automated testing on a given product would be mentioned in the test plan for that product. Tests can be designed and executed under the test design and execution process, and the environment and test data requirements would be covered under the environment and data management process. Testing can be monitored, controlled and reported on under the monitor and control activity. Defects detected during testing can be raised under the incident reporting process. Once testing is complete, assets can be archived, and the outcomes of testing reported, under the test completion process.

4.2.12 Continuous integration and continuous testing

Continuous integration is a development practice whereby code is integrated into a build whenever it is checked into a shared code repository.

In continuous testing, each time a new build is produced an automated test suite (normally for unit and regression testing) is executed against the build, to ensure the latest code changes have not introduced any new defects into the system.

Table 9 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Continuous integration and continuous testing	All clauses	When conducting continuous testing, automated testing is required, to confirm that all systems are behaving correctly after build and after deployment. For automated testing, all clauses of ISO/IEC/IEEE 29119-2 apply. For example, the requirement to conduct automated testing can be included in the test policy, and the chosen approach to automated testing would be described in detail in the organizational test practices. The requirement to use automated testing on a given product would be mentioned in the test plan for that product. Tests can be designed and executed under the test design and execution process, and the environment and test data requirements would be covered under the environment and data management process. Testing can be monitored, controlled and reported on under the monitor and control activity. Defects detected during testing can be raised under the incident reporting process. Once testing is complete, assets can be archived, and the outcomes of testing reported, under the test completion process.

4.2.13 Cross-functional team

In cross-functional teams, the team has all the skills and capabilities required to deliver the product. ISO/IEC/IEEE 29119-2 does not place any requirements on which role (e.g. developer, tester) can implement each process in the standard. Any agile team member can implement any process, activity

or task within the standard. Therefore, any team member can implement any test process or activity, including test design, test execution and incident reporting. Thus, this concept does not map directly onto the standard.

4.2.14 Daily stand-up

A daily stand-up is a key means of communication in agile projects. A common format is a timeboxed meeting of 5 min to 15 min, during which each agile team member explains what they did yesterday and what they will do today to meet the team's iteration goal (see 4.2.28), along with any impediments ("blockers") that can prevent them from meeting the goal. Daily stand-ups are also referred to as "daily Scrum".

Table 10 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Daily stand-up	7.3.4.3 Monitor	Testers monitor progress towards meeting iteration goals by monitoring test progress against the iteration test plan, which is carried out under the monitor activity.
	7.3.4.4 Control	Any control directives identified as a result of monitoring would be handled under the control activity. This can be via verbal communication at daily stand-up.
	7.3.4.5 Report	Test progress is often reported at daily stand-up.

4.2.15 Definition of done

The "definition of done" is a set of criteria to be met before a user story can be declared "done" (i.e. closed). It typically means that no further development or testing is required before the feature developed for that user story can be deployed to production. The definition of done typically includes test completion criteria, which need to be met before the testing of each user story can be deemed complete. All team members need to share an agreed understanding of the definition of done.

EXAMPLE The definition of done can include the requirement that all user acceptance tests pass.

Table 11 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Definition of done	6.2 Organizational test process	The definition of done is required to include the requirement that acceptance tests pass before each user story is considered "done." Passing other types and levels of testing can also be mentioned in the definition of done. This requirement can be stated in the test policy, and the approach to each type and level of testing would be explained in the organizational test practices.
	7.2 Test strategy and planning process	The requirement for each user story passing specific types and levels of testing, as part of the definition of done, would be specified in the test plan, such as test completion criteria and exit criteria.
	7.3.4.3 Monitor	Test progress against the definition of done would be monitored under the monitor activity.
	7.3.4.4 Control > task c)	During the monitoring activity, if it looks likely that test completion criteria in the definition of done will not be met for a given user story, then the control activity, task c), would be used to implement any necessary actions to ensure the definition of done can be met, This can include changes to the testing, the test plan, test data, test environment, staffing and/or changes in other areas, such as development.
	7.3.4.4 Control > task d)	Test completion criteria can be updated during monitoring and control to treat newly identified risks, using the control activity, task d).
Confirming done	7.3.4.4 Control > task g)	Confirming that test completion criteria have been met for each user story would be carried out during each iteration, using the control activity, task g).

4.2.16 Definition of ready

Before a user story can enter an iteration, it should be reviewed against a series of "readiness" criteria to determine whether it can (in theory) be developed, tested and delivered within an iteration. Readiness criteria typically include entry criteria for determining whether development and testing is ready to commence.

EXAMPLE User stories are often deemed unfit for inclusion in an iteration if they do not include acceptance criteria.