TECHNICAL REPORT

ISO/IEC TR 29119-6

First edition
2021-07

Software and systems engineering — Software testing —

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

Ingénierie du logiciel et des systèmes — Essais du logiciel —

Partie 6: Lignes directrices pour l'utilisation de l'ISO/IEC/IEEE 29119 (toutes les parties) dans les projets agiles

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 ..... **v**

Introduction ..... **vi**

1 Scope **1**

2 Normative references **1**

3 Terms and definitions **1**

4 Concepts **1**

4.1 Agile practices and artefacts 1

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

4.2.1 Overview 2

4.2.2 Acceptance criteria 3

4.2.3 Acceptance test-driven development (ATDD) 3

4.2.4 Amplify learning 3

4.2.5 Backlog management 3

4.2.6 Behaviour-driven development (BDD) 4

4.2.7 Build integrity in 5

4.2.8 Burn-down and burn-up charts 5

4.2.9 Co-located teams 6

4.2.10 Collective code ownership 6

4.2.11 Continuous delivery and deployment 6

4.2.12 Continuous integration and continuous testing 7

4.2.13 Cross-functional team 7

4.2.14 Daily stand-up 8

4.2.15 Definition of done 8

4.2.16 Definition of ready 9

4.2.17 Eliminate waste 10

4.2.18 Empowered team 11

4.2.19 Emergent design 11

4.2.20 Epic 11

4.2.21 Fast user feedback 11

4.2.22 Feature-driven development (FDD) 12

4.2.23 Feature toggle 12

4.2.24 Frequent interaction with product owner 12

4.2.25 Increment 12

4.2.26 Informal defect management 12

4.2.27 Iteration backlog 13

4.2.28 Iteration goal 13

4.2.29 Iteration planning 13

4.2.30 Iteration review 13

4.2.31 Iteration zero 14

4.2.32 Just in time 14

4.2.33 Limit work in progress 14

4.2.34 Mood chart 14

4.2.35 Occasional test iterations 15

4.2.36 Pair programming 15

4.2.37 Parallel test iterations 15

4.2.38 Planning poker 15

4.2.39 Product backlog 16

4.2.40 Product owner 16

4.2.41 Refactoring 16

4.2.42 Relative estimation 16

4.2.43 Release planning 17

4.2.44 Retrospective meeting 17

4.2.45 Scrum master 17

4.2.46 Self-organizing teams 17
4.2.47 Short iterations 17
4.2.48 Simplicity 18
4.2.49 Story mapping 18
4.2.50 Story testing 18
4.2.51 Sustainable pace 18
4.2.52 Task board 18
4.2.53 Team charter 18
4.2.54 Team room 18
4.2.55 Team-based estimation 19
4.2.56 Technical debt 19
4.2.57 Test-driven development (TDD) 19
4.2.58 Timebox 20
4.2.59 Transparency 20
4.2.60 User story 20
4.2.61 User stories – INVEST mnemonic 20
4.2.62 User story format – role/feature/rationale 20
4.2.63 Velocity 21

Annex A (informative) **Mapping of The Scrum Guide to ISO/IEC/IEEE 29119-2 test processes** **22**

Annex B (informative) **Mapping of ISO/IEC/IEEE 29119-2 (test processes) to agile practices and techniques covered under Clause 4** **24**

Annex C (informative) **Example mapping of typical agile test artefacts to ISO/IEC/IEEE 29119-3 test documentation** **37**

Annex D (informative) **Example agile test artefact** **39**

Bibliography **45**

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 terminology databases for use in standardization at the following addresses:

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

— IEC Electropedia: available at https://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.

Table 12 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Definition of ready	6.2 Organizational test process	The definition of ready is required to include the requirement that acceptance criteria are included in user story. This quality related requirement can be stated in the test policy, and the approach to ensuring acceptance criteria are suitable would be explained in the organizational test practices.
	7.2.4.6 Design test strategy > task c)	Entry criteria would be defined in the test plan, via the design test strategy activity, task c).
	7.3.4.3 Monitor	The ability for a user story to meet 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 entry criteria in the definition of ready 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 ready can be met, This can include changes to testing, the test plan, test data, test environment, staffing and/or changes in other areas, such as business analysis.
	7.3.4.4 Control > task d)	Entry criteria can be updated during monitoring and control to treat newly identified risks, using the control activity, task d).
Confirming readiness	7.3.4.4 Control > task g)	Confirming that entry criteria have been met for each user story would be carried out during each iteration, using the control activity, task g).

4.2.17 Eliminate waste

The elimination of waste is a key concept of lean, in which waste is defined as any activity, document or defect that consumes time, resources or space without adding value to a product or service.

Table 13 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Eliminate waste	4.1.3 Tailored conformance	The standard enables the elimination of waste via the tailoring clause. Tailored conformance allows organizations to only implement the specific test practices and to produce the level test documentation that is required to meet their needs.
	6 Organizational test process	The requirement to create lean test documentation would be stated in the test policy. The approach to achieving this would be described in detail in the organizational test practices.
	7.2.4.4 Identify and analyse risks	The standard enables the elimination of waste via the identification of risks and the assignment of an exposure level to each risk, under the Identify and Analyse Risk activity. This enables a potential reduction in test effort, as described in ISO/IEC/IEEE 29119-2:—, 7.2.4.5.
	7.2.4.5 Identify risk treatment approaches	The standard enables the elimination of waste by allocating risk treatments/mitigations according to the exposure level assigned to each risk, thereby supporting a reduction in test effort based on risk. This is achieved via the identify risk treatment approaches activity, and would occur during release and iteration planning.

Table 13 *(continued)*

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
	8 Dynamic test processes: 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)	Waste is further eliminated by prioritising all test assets that are used during test execution, including the test model, test coverage items, test cases and test procedures.

4.2.18 Empowered team

Empowering the team is a key concept of lean and agile, in which the agile team is encouraged to make decisions that, in the past, would typically have been made by managers.

EXAMPLE Process improvement is an example of something the team can be empowered to do.

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. Thus, the concept of an empowered team does not map directly onto the standard.

4.2.19 Emergent design

Emergent design involves agile teams delivering functionality whilst system and software design evolves over time. This often requires refactoring to resolve any design issues that emerge. Refactoring usually requires regression testing to ensure code changes have not introduced defects. The concepts of emergent design and refactoring do not map directly to any specific process of ISO/IEC/IEEE 29119-2.

4.2.20 Epic

An epic is a large user story that can be decomposed into a collection of smaller related user stories. Epics are usually too large to be comfortably implemented in a single iteration.

The standard does not place any requirements on the format that system requirements are specified in. Therefore, any requirement format is permitted. However, the concept of epics does not map directly onto ISO/IEC/IEEE 29119-2.

4.2.21 Fast user feedback

One of the benefits of agile is the ability to gain frequent customer and user feedback on whether a system meets their requirements. This is often sought via user acceptance testing during each iteration and via the iteration review meeting, to verify that each new feature meets the acceptance criteria and requirements of each user story.

Table 14 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Fast user feedback	All clauses	Fast user feedback via acceptance testing per iteration relates to all clauses, since the requirement to conduct acceptance testing during each iteration can be specified in the test policy, the approach described in the organizational test plan and included in the test plan. Acceptance testing is then monitored, controlled and reported on under the test management processes and carried out in a specific test environment often with specific data that are required to be requested, set up and maintained as needed. Acceptance tests are then designed and executed and with test incidents (defects) raised as necessary.

4.2.22 Feature-driven development (FDD)

Feature-driven development is an iterative and incremental system development approach in which feature implementation is driven by the level of client value in each feature, with frequent release of working software to the customer.

Although ISO/IEC/IEEE 29119-2 is designed to be implemented under any life cycle, including iterative methodologies, the concept of FDD does not map directly onto the standard.

4.2.23 Feature toggle

Feature toggles make continuous delivery less disruptive by allowing features to be turned on and off as required, for specific customers or for all customers. Feature toggles are often used when conducting "A/B testing" of new system features, to determine if a chosen customer or set of customers like the new feature. There is no specific concept in ISO/IEC/IEEE 29119-2 that maps to this concept.

4.2.24 Frequent interaction with product owner

Frequent interaction with the product owner provides the agile team with new and updated user stories and regular feedback on their product, improving the team's ability to meet the product owner's vision.

ISO/IEC/IEEE 29119-2 does not place any requirements on the frequency with which stakeholders are consulted or on how often testing occurs. Therefore, frequent iteration with the product owner is allowed. However, this concept does not map directly onto the standard.

4.2.25 Increment

In agile, products are produced iteratively, with each iteration resulting in a potentially releasable "increment" of the system. In many cases, increments are released after several iterations rather than each iteration. Increments are also referred to as "product increments."

ISO/IEC/IEEE 29119-2 is designed to be implemented under any life cycle model, including iteratively against product increments. This includes all aspects of test planning, test case design, test execution and reporting. However, the concept of increments does not map directly onto the standard.

4.2.26 Informal defect management

In agile, defects are often raised informally, such as by adding a comment to a story card, speaking directly with developers and demonstrating defects for them or by emailing defect evidence to developers when teams are not co-located.

Table 15 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Informal defect management	8.5 Test incident reporting process	The standard supports the decision of resolving an incident without formally reporting the incident.

4.2.27 Iteration backlog

An iteration backlog is the set of backlog items (e.g. user stories) that are selected for inclusion in a given iteration. Iteration backlogs are also referred to as "sprint backlogs" and "sprint catalogues."

ISO/IEC/IEEE 29119-2 is designed to be implemented under any life cycle model, including iteratively against user stories in an iteration backlog. However, the concept of an iteration backlog does not map directly onto the standard.

4.2.28 Iteration goal

An iteration goal is a summary of the business and technical goals that an agile team agrees to achieve within an iteration. An iteration goal is typically defined during iteration planning. Iteration goals are also called "sprint goals."

EXAMPLE If the stories selected for implementation in a given iteration include the development of features that enable online credit card payments, then an iteration goal can be "Enable online credit card payments."

Testing can be conducted under ISO/IEC/IEEE 29119-2 to meet a given iteration goal. However, this concept does not map directly onto the standard.

4.2.29 Iteration planning

The purpose of iteration planning is to decide which user stories will be implemented during an iteration and to assign and schedule tasks to complete that work. Each story is typically discussed at the iteration planning meeting, so that all team members have the opportunity to understand how the stories will be implemented and tested. Iteration planning also usually includes team agreement on effort estimates (e.g. via planning poker) to confirm that the team can complete the user stories in the given iteration. Iteration planning is also known as “sprint planning.”

Table 16 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Iteration planning	7.2 Test strategy and planning process	Iteration planning meetings need to include test planning, which would be carried out under the test strategy and planning process. This would include identification of risks, types and levels of testing required and estimates for each user story.

4.2.30 Iteration review

During an iteration review, the agile team reviews what was "done" during the iteration, compares what was achieved against the iteration goal and collaborates on what to do in the next iteration. The agile team can also demonstrate the new product features to stakeholders and review the timeline, budget and longer-term release plans. An iteration review is also called a "sprint review."

Table 17 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Iteration review	8.4.4.2 Execute test procedures > task a)	At the iteration review meeting, agile teams often re-execute tests that have passed to demonstrate correct product behaviour to users. This would be carried out under the execute test procedures activity, task a).

4.2.31 Iteration zero

When used, iteration zero is the first iteration of any project and can include a variety of set-up activities such as agreeing on the definitions of ready and done, setting up tools and environments and creating the initial backlog to ensure enough user stories are ready for the first iteration. Iteration zero is also referred to as "sprint zero."

Table 18 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Iteration zero	7.2 Test strategy and planning process	Iteration zero includes the establishment of test plans via the test strategy and planning process.
	7.3.4.2 Set-up	Iteration zero includes the establishment of test measures, changing risks and monitoring activities such as status reporting via the test controls set-up activity.
	8.3 Test environment and data management process	Iteration zero includes the set-up of the test environment and any required data under the test environment and data management process.

4.2.32 Just in time

The concept of "just in time" is a key concept in lean that aims to increase efficiency and decrease waste by developing product features only when they are needed. "Decide as late as possible" and "last responsible moment" are other examples of the just-in-time concept.

ISO/IEC/IEEE 29119-2 is designed to be implemented under any life cycle model, including iterative models. This allows each individual test activity and task to be implemented only when it is needed. However, this concept does not map directly onto the standard.

4.2.33 Limit work in progress

Limiting work in progress (WIP) is a core feature of Kanban. Limiting the number of simultaneous tasks being worked on by the team reduces the amount of time wasted in context switching between tasks. It also reduces bottlenecks in the production process by forcing predecessor tasks to be completed sooner. Limiting work in progress is also referred to as “limiting work in process”.

EXAMPLE An example of limiting work in progress is minimized lead times (also known as "lead time reduction").

ISO/IEC/IEEE 29119-2 is designed to be implemented under any life cycle model, including iterative models. This allows each individual test activity and task to be implemented when it is needed. However, this concept does not map directly onto the standard.

4.2.34 Mood chart

The purpose of a mood chart is to track and enable assessment of the mood of the agile team. It is also referred to as a "niko-niko". There is no corresponding concept for a mood chart in ISO/IEC/IEEE 29119-2.

4.2.35 Occasional test iterations

Ideally, all development and testing for each user story takes place within the same iteration. However, sometimes not all testing can be completed within the iteration. For instance, when the test environment is not available, or users are not available for acceptance testing, in which case, such testing may be delayed and performed during an "occasional test iteration."

Table 19 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Occasional test iterations	All clauses	If an occasional test iteration is performed, then all test processes in the standard can be used in the iteration, and the requirement to conduct this approach across the organization can be addressed in the test policy and organizational test practices.

4.2.36 Pair programming

Pair programming is a practice from extreme programming (XP) that typically involves one developer writing source code or configuring a system while a second developer observes their work to look for errors and to determine whether the work completed is accurate and meets its requirements.

There is no corresponding concept for pair programming in ISO/IEC/IEEE 29119-2. However, the concept of this type of informal review is supported by ISO/IEC 20246 work product reviews.

4.2.37 Parallel test iterations

In a parallel test iteration, development and testing are carried out by two separate teams. Once a development iteration is completed, the testing of the deliverables is performed in a testing iteration. In this way, two parallel streams of development and testing iterations are performed.

NOTE 1 Parallel test iterations are also referred to as testing in “iteration plus one,” “sprint plus one” or “sprint N + 1.”

NOTE 2 It is usually preferred that all development and testing is completed in the same iteration.

Table 20 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Parallel test iterations	All clauses	If a parallel test iteration is performed, then all relevant test processes in the standard can be used in the iteration.

4.2.38 Planning poker

Planning poker is a team-based estimation approach whereby relative estimates of the effort required to develop and test each user story are set via consensus. The approach uses poker cards that are typically based on a Fibonacci sequence (i.e. 1, 2, 3, 5, 8, 13).

Table 21 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Planning poker	7.2.4.6 Design test strategy > task f)	Initial test estimates for each user story and test activity would be identified during iteration zero, via the design test strategy activity, task f).
	7.2.4.8 Record test plan > task a)	Refined test estimates would then be provided during planning poker, using the record test plan activity, task a).

4.2.39 Product backlog

A product backlog is an evolving list of requirements that need to be "done" for a given product. The most common type of product backlog item is a user story, though other types of backlog items include functional, non-functional, technical and testing requirements. Product backlogs are also referred to as "backlogs".

Product owners typically prioritize product backlogs according to the value that each user story presents to the customer, though they can also consider project or product risks identified during test planning. Prioritization is usually carried out with input from the agile team.

The concept of a product backlog does not map directly to ISO/IEC/IEEE 29119-2.

4.2.40 Product owner

The product owner is an individual who represents the customer's views on the project. They are required to be an individual and not a group so that decisions can be made quickly. They are responsible for the product backlog. They are the primary interface for product requirements. This concept does not map to any specific process of ISO/IEC/IEEE 29119-2.

4.2.41 Refactoring

The purpose of refactoring is to reengineer existing design, source code or tests to improve non-functional system attributes, without changing the system's external functional behaviour. Refactoring usually requires regression testing, to ensure code changes have not introduced defects. The concept of refactoring does not map directly to any specific process of ISO/IEC/IEEE 29119-2.

4.2.42 Relative estimation

Relative user story estimation is based on determining the comparative levels of estimated effort to design, implement and test each user story, compared to that of other stories.

EXAMPLE 1 Story points are one approach to relative estimation. The numbers of a Fibonacci sequence (1, 2, 3, 5, 8, 13, 21, 34, 55, ...) are sometimes used as story point values.

EXAMPLE Relative sizes such as small, medium, large and extra-large can be used in place of story points.

Table 22 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Relative estimation	7.2.4.5 Design test strategy > task f)	Initial test estimates for each user story can be set during iteration zero for user stories existing at that time, via the design test strategy activity, task f).
	7.2.4.7 Record test plan > task a)	Test estimates can be added and refined during each iteration (e.g. during planning poker) via the record test plan activity, task a).

4.2.43 Release planning

The purpose of release planning is to identify a set of epics and features that will be implemented and released at the end of a given series of iterations. This often involves identifying a theme for a given release and selecting or creating user stories that fit under that theme. It is a common approach in situations where less frequent releases are desired.

Table 23 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Release planning	7.2 Test strategy and planning process	Release planning needs to include test planning, which would be carried out under the Test strategy and planning process.

4.2.44 Retrospective meeting

During retrospectives, the agile team usually identify what went well during each iteration, what can be improved and actions to implement the improvements. Retrospectives enable continuous improvement. Retrospectives are also referred to as “iteration retrospectives” or "sprint retrospectives."

EXAMPLE This can include improvements to agile processes, procedures, tools, methodologies and culture.

Table 24 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Retrospective meeting	7.4.4.4 Identify lessons learned	The identification of improvements for testing would be carried out under the identify lessons learned activity during each retrospective.

4.2.45 Scrum master

A Scrum master facilitates the processes and meetings of the agile team. They also remove obstacles identified during daily stand-ups and can act as an interface between the team and the product owner.

ISO/IEC/IEEE 29119-2 does not place requirements on which role (e.g. developer, tester) can implement each process in the standard. The Scrum master can implement any process, activity or task within the standard. However, this concept does not map directly onto the standard.

4.2.46 Self-organizing teams

Self-organizing teams decide how best to accomplish their work by determining which team members are best suited to complete each task.

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. However, this concept does not map directly onto the standard.

4.2.47 Short iterations

In agile, systems are designed, developed, tested, accepted and released in a series of short iterations (usually 2 to 4 weeks per iteration). This enables frequent change, adaptation and delivery of the product based on evolving user and system requirements. Each iteration creates a potentially releasable "increment" of the final system. Iterations are also referred to as "sprints."

ISO/IEC/IEEE 29119-2 can be implemented iteratively, which allows testers in agile teams to respond to changing testing requirements. However, the concept of short iterations does not map directly onto the standard.

4.2.48 Simplicity

Agile teams strive for the simplest design possible while still providing the required behaviour, as this produces more maintainable and testable systems. It is a fundamental principle of XP and agile. This concept does not map directly to any specific concept defined in ISO/IEC/IEEE 29119-2.

4.2.49 Story mapping

Story mapping is an approach to organizing stories into releases (usually under epics that are chosen for inclusion in the release).

ISO/IEC/IEEE 29119-2 places no requirements on the way in which requirements are grouped for development and testing. However, this concept does not map directly onto the standard.

4.2.50 Story testing

The aim of story testing is to ensure acceptance criteria have been met (in preparation for user acceptance testing). Story testing is typically carried out by testers after unit and integration testing.

ISO/IEC/IEEE 29119-2 is designed to be implemented under any type of testing, including story testing. However, this concept does not map directly onto the standard.

4.2.51 Sustainable pace

Sustainable pace is ensuring the agile team is not overworked in order to achieve deadlines, at the expense of their ability to meet future deadlines. It is an XP practice and is one of the agile principles (referred to as the "40-hour working week" in XP).

ISO/IEC/IEEE 29119-2 does not place any requirements on when each testing process, activity and task is performed. Thus, this is permitted. However, this concept does not map directly onto the standard.

4.2.52 Task board

A task board provides a visual representation of an agile team's progress within an iteration. Task boards are also referred to as "Kanban boards" and "Scrum boards".

Table 25 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Task board	7.3.4.3 Monitor	Test progress against each user story would be monitored under the monitor activity. This information would be used to update the task board.
	7.3.4.4 Control	Any issues with progress made during testing would be handled through the control process. Control directives can be communicated during daily stand-up.

4.2.53 Team charter

A team charter specifies the agile team's goals/project aim, scope, approach, success criteria, risks, constraints and indicators of success. The charter is designed and agreed by the agile team and is usually displayed in a visible location, such as next to the team's task board. A team charter is also called a "project charter". This concept does not map to ISO/IEC/IEEE 29119-2.

4.2.54 Team room

It is ideal when agile team members (e.g. product owner, business analyst, developers, testers) can be situated in the same location. This is sometimes referred to as the “team room”.

ISO/IEC/IEEE 29119-2 places no requirements on the location at which testers complete their work. Thus, this is permitted by the standard. However, this concept does not map directly onto the standard.

4.2.55 Team-based estimation

All members of an agile team typically participate in user story estimation (e.g. via planning poker). Estimates can include effort for design, development and testing.

Table 26 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Team-based estimation	7.2.4.6 Design test strategy > task f)	The standard 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 estimation tasks. Initial test estimates are usually developed during iteration zero, via the design test strategy activity, task f).
	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).

4.2.56 Technical debt

Technical debt can take several forms, such as difficult-to-maintain design, code or tests and can also include unresolved defects. Technical debt typically occurs when meeting deadlines becomes more important than achieving long-term quality goals. Technical debt can occur accidentally (e.g. when agile team members make mistakes) or deliberately (e.g. to meet deadlines). This concept does not map directly to ISO/IEC/IEEE 29119-2.

4.2.57 Test-driven development (TDD)

Test-driven development (TDD) is a software development approach in which automated tests are designed for each new feature, before the code for that feature is written. Each test will therefore fail in the first instance, with the goal of passing once code is written to pass each test. In each case, a minimal amount of code is written to pass each test. The test is then re-run and the code is fixed (if required) until it passes the test.

Table 27 — Mapping to ISO/IEC/IEEE 29119-2

Agile concept	ISO/IEC/IEEE 29119-2:— clause	Mapping details
Test-driven development	All clauses	When conducting TDD, all clauses of ISO/IEC/IEEE 29119-2 apply. The requirement to conduct TDD can be included in the test policy, and the chosen approach to TDD would be described in detail in the organizational test practices. The requirement to use TDD 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.

4.2.58 Timebox

Many activities in agile are timeboxed, including most agile meeting (e.g. daily stand-up, iteration planning, retrospective) and iterations (e.g. 2-week iterations).

ISO/IEC/IEEE 29119-2 does not place any requirements on how long each clause is implemented. Thus, timeboxed implementation is permitted. However, this concept does not map directly onto the standard.

4.2.59 Transparency

Transparency in agile is being open about everything on the agile project. This can include making the agile team's progress easily visible or can mean that the product owner shares information on planned future releases with the agile team. This concept does not map to any specific clause of ISO/IEC/IEEE 29119-2.

4.2.60 User story

User stories are the most common type of backlog item in agile projects, with each story specifying a small, independent feature that a user requires in a system. Each user story typically includes a rationale of why a given set of users require the feature and which users it will benefit. User stories can also be used to specific non-functional, technical and testing requirements. User stories are also referred to as "backlog items" and "product backlog items" (PBI).

ISO/IEC/IEEE 29119-2 can be implemented against any form of requirements, including user stories. This, the concept of user stories is supported by the standard. However, this concept does not map directly onto the standard.

4.2.61 User stories – INVEST mnemonic

INVEST is a mnemonic that is used to review each user story, to ensure each one is ready to enter a given iteration. INVEST stands for:

a) Independent – each user story needs to be self-contained (i.e. not dependent on any other user story) so that each can be designed, developed, tested and accepted within a given iteration.

b) Negotiable – user stories are not designed to be written contracts. They need room for discussion and further refinement within an iteration.

c) Valuable – each user story needs to deliver value to its stakeholders.

d) Estimable – each user story needs to be able to be estimated, including design, development, testing and acceptance effort.

e) Small – each user story needs to be small enough so that it can be “done” within an iteration.

f) Testable – each user story needs to include enough information for it to be tested and accepted (e.g. it needs to include acceptance criteria).

This concept does not map directly onto the test processes defined in ISO/IEC/IEEE 29119-2.

4.2.62 User story format – role/feature/rationale

User stories are often specified using the format of "role/feature/rationale".

EXAMPLE 1 As a <role>, I want to <feature>, so that <rationale>.

EXAMPLE 2 As a mortgage broker, I want to create customer accounts, so that I can record the details of each new customer.

ISO/IEC/IEEE 29119-2 does not place any requirements on the format of the test basis (i.e. requirements). Therefore, user stories specified in role/feature/rationale format are permitted by the standard. However, this concept does not map directly onto the standard.

4.2.63 Velocity

Velocity specifies the number of units of work (e.g. story points) completed within an iteration. It is used to predict the amount of work that can be completed within future iterations. Velocity is not suitable for comparing the amount of work completed by different teams, as the approach to measuring velocity varies between teams.

Testers contribute to velocity by verifying that acceptance criteria have been met during testing. Testers also monitor the number of stories that have passed testing, where each completed story that passes contributes to team velocity. ISO/IEC/IEEE 29119-2 is designed to be utilised under any lifecycle model, including those that monitor team velocity. However, this concept does not map directly onto the standard.

Annex A
(informative)

Mapping of The Scrum Guide to ISO/IEC/IEEE 29119-2 test processes

A.1 Scrum life cycle

Figure A.1 illustrates the key elements of a typical Scrum life cycle.

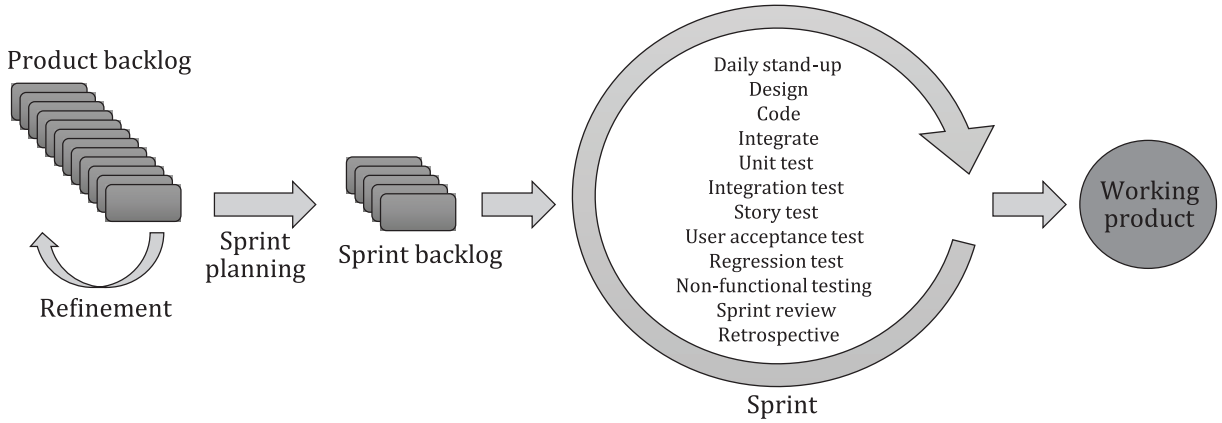


Figure A.1 — Typical Scrum life cycle

A.2 Mapping of The Scrum Guide to ISO/IEC/IEEE 29119-2 test processes

Table A.1 provides a mapping of The Scrum Guide to ISO/IEC/IEEE 29119-2.

Table A.1 — Mapping of The Scrum Guide to ISO/IEC/IEEE 29119-2

Scrum	ISO/IEC/IEEE 29119-2:— clause	Corresponding clause in this Document
Daily Scrum	7.3.4.3 Monitor 7.3.4.4 Control 7.3.4.5 Report	4.2.14 Daily stand-up
Definition of done	7.2.4.6 Design test strategy > task c) Confirming done: 7.3.4.4 Control > task f)	4.2.15 Definition of done
Development team	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.18 Empowered team
Increment	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.25 Increment
Product backlog	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.39 Product backlog
Product owner	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.40 Product owner

Table A.1 *(continued)*

Scrum	ISO/IEC/IEEE 29119-2:— clause	Corresponding clause in this Document
Refinement	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.30 Iteration review
Scrum master	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.45 Scrum master
Sprint	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.47 Short iterations
Sprint backlog	This concept does not map to any specific process of ISO/IEC/IEEE 29119-2	4.2.27 Iteration backlog
Sprint planning	7.2 Test strategy and planning process	4.2.29 Iteration planning 4.2.43 Release planning
Sprint retrospective	7.4.4.3 Identify lessons learned	4.2.44 Retrospective meeting
Sprint review	8.4.4.1 Execute test procedures > task a)	4.2.30 Iteration review

Annex B
(informative)

Mapping of ISO/IEC/IEEE 29119-2 (test processes) to agile practices and techniques covered under Clause 4

The purpose of Table B.1 is to map each activity in of ISO/IEC/IEEE 29119-2 to the agile practices and artefacts listed in Clause 4. The conformance clause of ISO/IEC/IEEE 29119-2 is also mapped in Table B.1.

Table B.1 — Mapping of ISO/IEC/IEEE 29119-2 clauses to agile practices and artefacts covered under Clause 4

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
4 Conformance	4.1.3 Tailored conformance	4.2.17 Eliminate waste
6 Organizational test process	6.2.4.2 Develop organizational test specification	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	6.2.4.3 Monitor and control use of organizational test specification	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	6.2.4.4 Update organizational test specification	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
7 Test management processes	7.2 Test strategy and planning process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous deployment and delivery 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development
	7.2.4.2 Understand context	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.2.4.3 Organize test plan development	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)
	7.2.4.4 Identify and analyse risk	4.2.3 Acceptance test-driven development (ATTD) 4.2.5 Backlog management 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.2.4.5 Identify risk treatment approaches	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)
	7.2.4.6 Design test strategy	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.38 Planning poker 4.2.42 Relative estimation 4.2.43 Release planning 4.2.55 Team-based estimation 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.2.4.6 Determine Staffing and Scheduling	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)
	7.2.4.7 Record Test Plan	4.2.3 Acceptance test-driven development (ATTD) 4.2.5 Backlog management 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.38 Planning poker 4.2.42 Relative estimation 4.2.43 Release planning 4.2.55 Team-based estimation 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.2.4.8 Gain Consensus on Test Plan	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)
	7.2.4.9 Communicate Test Plan and Make Available	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.29 Iteration planning 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.43 Release planning 4.2.57 Test-driven development (TDD)
	7.3 Test Monitoring and Control Process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.3.4.1 Set-Up	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.15 Definition of done 4.2.21 Fast user feedback 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	7.3.4.3 Monitor	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.8 Burn-down and burn-up charts 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.14 Daily stand-up 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.52 Task board 4.2.57 Test-driven development (TDD)
	7.3.4.4 Control	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.14 Daily stand-up 4.2.15 Definition of done 4.2.16 Definition of ready 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.52 Task board 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.3.4.5 Report	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.8 Burn-down and burn-up charts 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.14 Daily stand-up 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	7.4 Test completion process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	7.4.4.2 Archive test assets	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	7.4.4.3 Clean up test environment	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	7.4.4.4 Identify lessons learned	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.44 Retrospective meeting 4.2.57 Test-driven development (TDD)
	7.4.4.5 Report test completion	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
8 Dynamic test processes	8.2 Test design and implementation process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.2.4.2 Create test model	4.2.2 Acceptance criteria 4.2.3 Acceptance test-driven development (ATTD) 4.2.5 Backlog management 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	8.2.4.3 Identify test coverage items	4.2.3 Acceptance test-driven development (ATTD) 4.2.5 Backlog management 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.2.4.4 Derive test cases	4.2.3 Acceptance test-driven development (ATTD) 4.2.5 Backlog management 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.2.4.5 Create test procedures	4.2.3 Acceptance test-driven development (ATTD) 4.2.5 Backlog management 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.17 Eliminate waste 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	8.3 Test environment and data management process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.3.4.2 Establish test environment	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.3.4.3 Prepare test data	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.3.4.4 Maintain test environment	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	8.3.4.5 Maintain test data	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.31 Iteration zero 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.4 Test execution process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.4.4.2 Execute test procedures	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.30 Iteration review 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.4.4.3 Compare test results	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Table B.1 *(continued)*

ISO/IEC/IEEE 29119-2:— clause		Agile practices and artefacts covered in Clause 4
Clause	**Activity**	
	8.4.4.4 Record test execution	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.5 Test incident reporting process	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.26 Informal defect management 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.5.4.2 Analyse test results	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.26 Informal defect management 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)
	8.5.4.3 Create/update incident report	4.2.3 Acceptance test-driven development (ATTD) 4.2.6 Behaviour-driven development (BDD) 4.2.7 Build integrity in 4.2.11 Continuous delivery and deployment 4.2.12 Continuous integration and continuous testing 4.2.21 Fast user feedback 4.2.26 Informal defect management 4.2.35 Occasional test iterations 4.2.37 Parallel test iterations 4.2.57 Test-driven development (TDD)

Annex C
(informative)

Example mapping of typical agile test artefacts to ISO/IEC/IEEE 29119-3 test documentation

Table C.1 provides an example mapping of typical test artefacts produced by agile teams to the documentation templates defined in ISO/IEC/IEEE 29119-3. The table includes artefacts that are not exclusively used for testing, but that are commonly used during agile testing.

To enable lightweight test documentation, ISO/IEC/IEEE 29119-3:—, Clause 4 permits tailoring, allowing organizations to claim conformance to only those clauses that apply to their life cycle. This includes allowing document headings to be renamed, reordered, combined or removed. The standard also permits storage of test artefacts in tools, reducing maintenance overheads.

Table C.1 — Example mapping of typical agile test artefacts to ISO/IEC/IEEE 29119-3

Agile test artefact	ISO/IEC/IEEE 29119-3:— clause	Mapping details
Test policy	6.2 Test policy	The policies for agile testing that are standard across all areas of an organization are often specified in the test policy, and the approach to testing is described in detail in the organizational test practices. This reduces the amount of detail that is needed in each subsequent set of test documentation, reducing the length of test plans, test strategies and test reporting artefacts, as each can simply refer to the agreed terms stated in the organizational test documentation.
Organizational test practices	6.3 Organizational test practices	
Test plan	7.2 Test plan	Test plans in agile are typically lean (e.g. a wiki page, mind map, whiteboard photo) and are often derived during release and iteration planning. Provided the information defined in ISO/IEC/IEEE 29119-3:—, 7.2 is covered somewhere in the test plan, then such lightweight test plans are fully supported by the standard. If any specific section of the test plan in ISO/IEC/IEEE 29119-3:—, 7.2 cannot be met, then users of the standard can tailor out those clauses via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Test strategy	7.2.7 Test strategy	Test strategies in agile are typically lean (e.g. a wiki page, mind map, whiteboard photo), are often derived during release planning and iteration planning, and are often created separately to the test plan. Provided the information defined in ISO/IEC/IEEE 29119-3:—, 7.2.7 is covered somewhere in the test strategy, then such lightweight test strategies are fully supported by the standard. If any specific section of the test strategy described in ISO/IEC/IEEE 29119-3:—, 7.2.7 cannot be met, then users of the standard can tailor out those clauses via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Exploratory tests	8 Dynamic test processes documentation	Exploratory testing is supported by the standard, by allowing ISO/IEC/IEEE 29119-3:—, Clause 8 to be tailored to the level of detail and evidence required during exploratory testing. For instance, during an exploratory testing session, the test models that would otherwise be captured in a test design specification can be captured as test ideas on a session sheet. If any specific section of the test artefacts described in ISO/IEC/IEEE 29119-3:—, Clause 8 cannot be met, then users of the standard can tailor out those clauses via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.

Table C.1 *(continued)*

Agile test artefact	ISO/IEC/IEEE 29119-3:— clause	Mapping details
Regression test	8 Dynamic test processes documentation	Regression test cases in agile are typically automated. This is supported by the standard, by implementing automated test cases that cover the requirements specified in ISO/IEC/IEEE 29119-3:—, Clause 8, which supports electronic creation, storage and execution of tests. If any specific section of the artefacts described in ISO/IEC/IEEE 29119-3:—, Clause 8 cannot be met by the test automation framework, then users of the standard can tailor out those clauses via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Automated test	8 Dynamic test processes documentation	Automated tests are supported by the standard, by implementing automated tests that cover the requirements specified in ISO/IEC/IEEE 29119-3:—, Clause 8, which supports electronic creation, storage and execution of tests. If any specific section of the artefacts described in ISO/IEC/IEEE 29119-3:—, Clause 8 cannot be met, then users of the standard can tailor out those clauses via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Test	8 Dynamic test processes documentation	In agile, tests are often documented informally (e.g. checklists, test ideas, mind maps, session sheets). During test design and execution in agile, testers usually still have to go through the test design process of identifying test models (e.g. test scenarios or test ideas), building test cases (e.g. selecting inputs and identifying expected outputs for each test) and executing test cases in a given order (i.e. test procedures). However, they seldom create detailed test documentation. This is supported by tailoring out those clauses of the standard that are not required, via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Defect	8.11 Test incident report	In agile, defects are often discussed directly with developers or captured as comments on a user story. This is supported by tailoring out those clauses of the standard that are not required, via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Test progress report	7.3 Test status report	In agile, progress is often reported verbally, at daily stand-up. via email (e.g. for distributed teams) or by making test progress visible in test management tools or build management tools. This is supported by tailoring out those clauses of the standard that are not required, via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.
Burn-down and burn-up chart	7.3 Test status report	The metrics collected during test status reporting can be used as input to burn-down and burn-up charts. This particularly includes which stories have passed testing.
Test completion report	7.4 Test completion Report	Agile test completion reports are usually lightweight and focus on metrics that demonstrate that products are ready for release. They can also be reported directly from test management automation or build tools. This is supported by tailoring out those clauses of the standard that are not required, via the guidance provided in ISO/IEC/IEEE 29119-3:—, Clause 4.

Annex D
(informative)

Example agile test artefact

D.1 Overview

This annex provides examples of agile test artefacts, with information on how they meet the requirements of ISO/IEC/IEEE 29119-3.

D.2 Test plan

Figure D.1 is an example test plan for a financial system, that is designed as a mind map. It achieves tailored conformance to ISO/IEC/IEEE 29119-3; although it meets all the requirements for test plans, it does not include 'common information elements' (e.g. unique identifier, issuing organization, approval authority), which are applicable to all documents in ISO/IEC/IEEE 29119-3.

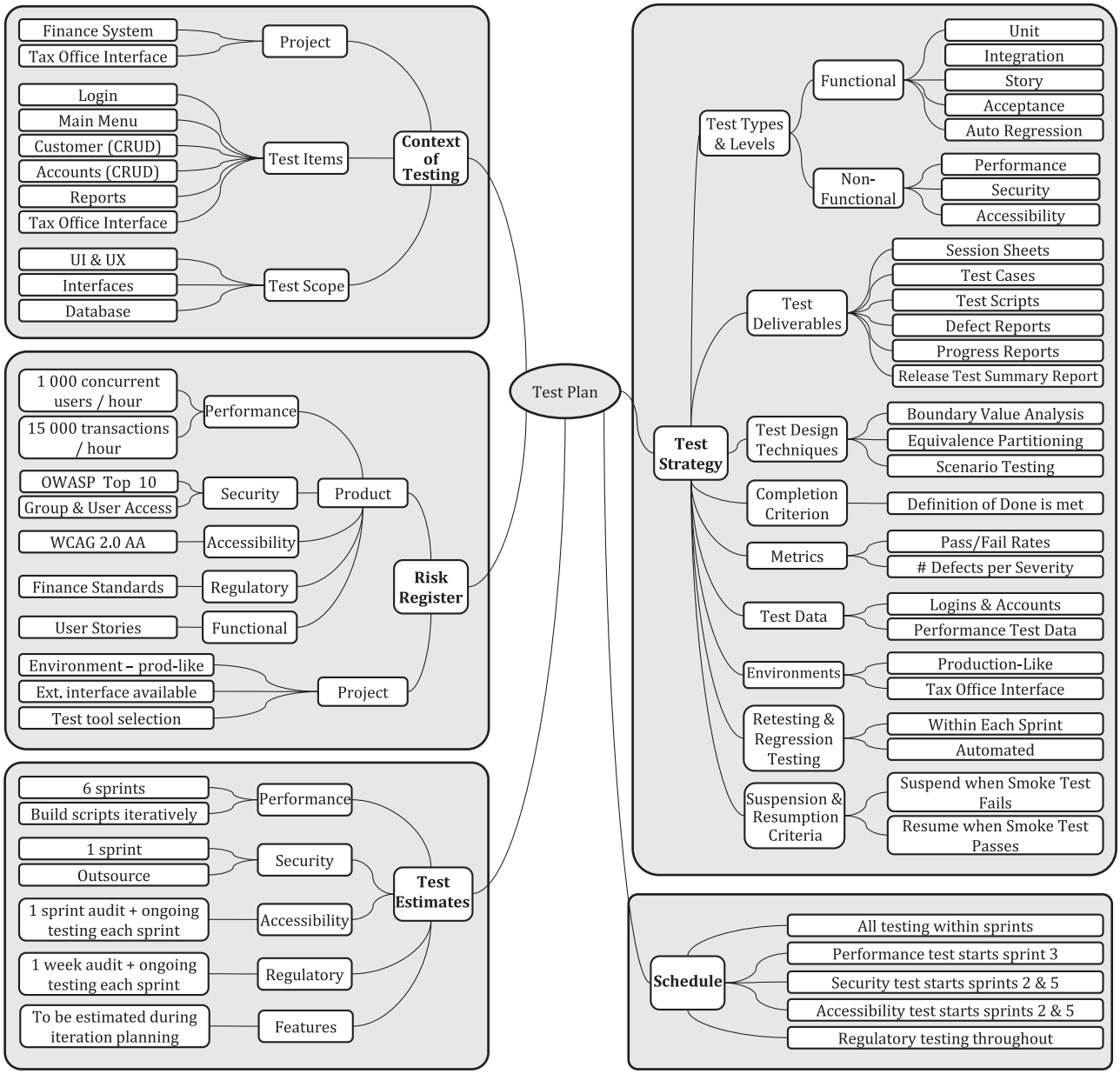


Figure D.1 — Example test plan

D.3 Session sheet

Figure D.2 is an example session sheet for a customer details screen. The session sheet achieves tailored conformance to ISO/IEC/IEEE 29119-3, as the specific test models and test cases for each test idea have not been specified, along with expected and actual test results and it does not include 'common information elements.'

Tester Name	A. B. Tester	Backlog Item #	BI_3467
Sprint #	4	Date 10/5/2018	
Session Duration	30 minutes	Build # 12.6.4	
Charter Explore	the Customer Details screen's data validation & error handling for UI input fields, using boundary value analysis. All input fields on the screen are in scope.		
Test Ideas and Notes	a) Test input fields with boundary value analysis 1) Name: allows 150 alphas & symbols – tested 149, 150, 151 2) Address Line 1: allows 100 alphanumeric & symbols – tested 99, 100, 101 3) Address Line 2: allows 100 alphanumeric & symbols – tested 99, 100, 101 4) Suburb: pick from list, matches postcode – tested first and last and attempted selection of just before first and just after last 5) Postcode: matches suburb – tested first and last of each valid postcode range and just before first and last of each range 6) State: pick from list – tested first and last and attempted selection of just before first and just after last 7) Contact number: allows 6 to 12 numbers with optional leading "+" – tested with 5, 6, 7, 11, 12, and 13 numbers, all with & without leading "+"		
Defects			

Each boundary is a part of the test model

Each boundary value is a test coverage item

Figure D.2 — Example session sheet

D.4 Defect report

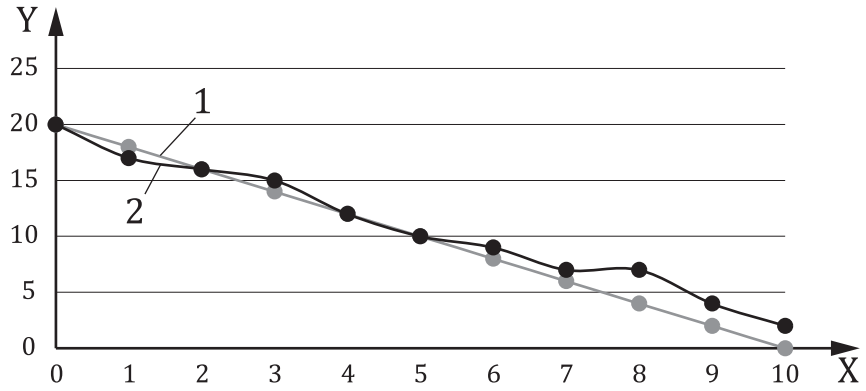
In agile, defects are often documented as comments on a story card, which is designed to reduce defect management overheads. This is illustrated in the example in Figure D.3. This achieves tailored conformance to ISO/IEC/IEEE 29119-3; although it meets all the requirements for incident reports, it does not include 'common information elements.'

Title: Customer address
Story: As an account manager I want to enter a customer's address on the Customer Details screen So that we can send the customer correspondence by mail
Acceptance Criteria 1: Given a valid / positive address is entered When the user enters all other details correctly and clicks Save Then the address is saved to the database **Acceptance Criteria 2:** Given an invalid / negative address is entered When the user enters all other details correctly and clicks Save Then the system will display an error message indicating that the address is invalid **Acceptance Criteria 3:** Given a valid / positive address is entered with a non-matching suburb/postcode combination When the user enters all other details correctly and clicks Save Then the system will display an error message indicating that suburb does not match postcode
Estimate: 4 points
Priority: High
Status: In Development
Comments: [A. B. Tester 18/3/2018 2:25pm]: Acceptance Criteria 3 failed testing. When entering suburb "Melbourne" with postcode "2000" (which is not a valid suburb/postcode combination), the system accepted the values and saved them to the database, without displaying an error message. This defect is high severity, priority 2, due to the high risk involved in saving incorrect customer addresses, as this could lead to customers not receiving required correspondence. Defect must be fixed before story can be set to Done. Story card returned to In Development, as this defect must be fixed.

Figure D.3 — Example of an informal defect report produced for an agile project

D.5 Test progress report

Agile teams often track test progress via a burn down chart, which tracks the number of user stories that have met the Definition of done within an iteration, communicating team productivity.



Key

X day number
Y no. of user stories
1 planned
2 actual

Figure D.4 — Example burndown chart

Testers in agile teams also commonly report test progress verbally at daily stand-up. However, in situations where stakeholders outside the agile team require written progress updates (e.g. to update them on progress towards meeting performance or security testing requirements, or to provide them with information on new or changed risks), testers can report test progress via a lightweight document, such as a regular email or via a wiki. Figure D.5 provides an example of a lightweight test progress report, which achieves tailored conformance to ISO/IEC/IEEE 29119-3, as it does not include a specific report against the test plan for the sprint or 'common information elements'.

Test Progress – Sprint 2 – Day 7

This sprint:

- # stories done (passed acceptance testing): 13
- # stories passed system testing: 14
- Total # stories failed system testing: 5
- Current # stories failed system testing: 1
- Total # stories tested today: 4

Risks & Issues:

- Test environment unavailable for 2 hours today due to connectivity issue with third-party interfaces (3 hours total environment downtime this sprint)
- Performance testing commencement delayed due to resource availability issue. Now planned for commencement on Friday (total delay = 9 days).
- Security issue identified on Login screen – returned to developers for fixing

Blockers & Workarounds:

- Performance tester not starting until Friday. Requesting support from development team to start 1-day pilot of likely performance test tool, to allow performance test planning to progress.

Figure D.5 — Example daily test progress report for an agile project

D.6 Test completion report

Lightweight test completion reports can be required in agile when testing outcomes have to be reported to external stakeholders (e.g. the project board). Figure D.6 provides an example of a

lightweight test completion report for a release on an agile project. It achieves tailored conformance to ISO/IEC/IEEE 29119-3; although it meets all the requirements for test completion reports, it does not include 'common information elements'.

Test Completion Report – Team A

Release #: 1

Sprints: 1 to 4

Testing completed:

- All user stories were developed and tested within the sprint
- Test types: functional, performance, security, data migration, accessibility, usability

Deviations from planned testing:

- Performance Testing engagement commenced 9 days later than hoped. Completed testing on time by allocating developers to support performance testing tasks.

Test completion evaluation:

- As all user stories were considered "Done" within each sprint, with all priority 1 and 2 tests passed, and with the product is considered to be meeting user requirements and ready for release.

Factors that blocked progress:

- Procurement process to appoint Performance Tester was delayed due to staff illness
- Total test environment downtime of 14 hours experienced due to connectivity issues with third-party interface environments and due to deployment windows needing to be moved into business hours.

Test measures:

- Provided in daily status reports – see agile team Wiki

Residual risks:

- There are no known quality or testing risks present that could impact on the release

Test deliverables:

- All testing deliverables are stored on the agile team Wiki under Testing link

Reusable test assets

- All test assets (test cases) are referenced from the agile team Wiki under Testing link

Lessons learned

- All lessons learned and actions to improve that were identified during testing were raised at fortnightly team retrospectives. See agile team Wiki > Retrospectives

Figure D.6 — Example test completion report for an agile project

Bibliography

[1] ISO/IEC 20246, *Software and systems engineering — Work product reviews*

[2] ISO/IEC/IEEE 29119-1, *Software and systems engineering — Software testing — Part 1: Concepts and definitions*

[3] ISO/IEC/IEEE 29119-2:—[1)], *Software and systems engineering — Software testing — Part 2: Test processes*

[4] ISO/IEC/IEEE 29119-3:—[2)], *Software and systems engineering — Software testing — Part 3: Test documentation*

[5] ISO/IEC/IEEE 29119-4, *Software and systems engineering — Software testing — Part 4: Test techniques*

[6] Schwaber K., Sutherland J., The Scrum Guide – The Definitive Guide to Scrum: The Rules of the Game. November 2017.

1) Under preparation.

2) Under preparation.

ICS 35.080

Price based on 45 pages