INTERNATIONAL STANDARD

ISO/IEC/ IEEE 26515

First edition
2011-12-01

Corrected version
2012-03-15

Systems and software engineering — Developing user documentation in an agile environment

Ingénierie du logiciel et des systèmes — Développement de la documentation de l'utilisateur dans un environnement agile

Reference number
ISO/IEC/IEEE 26515:2012(E)

Contents

Page

Foreword

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

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

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

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

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

ISO/IEC/IEEE 26515 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in cooperation with the Systems and Software Engineering Committee of the IEEE Computer Society, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

In this corrected version, the cover pages, front matter, page headers and footers have been corrected to reflect that ISO/IEC/IEEE 26515 is a joint development project under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

Introduction

Anyone who uses application software needs accurate information about how the software will help the users accomplish a task. The documentation may be the first tangible item that the user sees, and so influences the first impressions the users have of the product. If the information is supplied in a convenient form and is easy to find and understand, the users can quickly become proficient at using the product. Hence, well designed documentation not only assists the users and helps to reduce the cost of training and support, but also enhances the reputation of the product, its producer, and its suppliers.

Projects that implement agile development focus on providing rapid and frequent deliveries of high value software. These methods often involve detailed planning only for the short term, and the implementation of processes in parallel, rather than planning for an entire project in distinct phases.

Although agile development methods often advocate less life cycle documentation, the users of a software product still expect and require quality user documentation to be provided with these software products. Although the end results of the user documentation process are the same, the methods to get there may be very different in an agile environment.

Agile development methods may lead to the production of less user documentation, but the user documentation developed must be sufficient to meet the needs and requirements of the users. If the deliverables of user documentation and associated life cycle documentation are agreed in a contractual relationship between an acquirer and a supplier, then the deliverables that are produced are dictated by the terms of the contract. In these circumstances, the user and life cycle documentation deliverables that are agreed upon will depend on the demands of the acquiring organization regardless of the types of development methodologies used to produce them.

Technical writers and other personnel involved in the production of user documentation should understand the agile development processes used by their organization, and use the most effective agile development methods to produce relevant and useful user documentation.

Because of the nature of agile development methods, the traditional means of developing the end user documentation (both print and onscreen) as described in the current ISO/IEC 2651*n* family of International Standards are not entirely applicable.

This International Standard was developed to assist users of ISO/IEC 15288:2008 (IEEE Std 15288:2008), *Systems and software engineering — System life cycle processes*, or ISO/IEC 12207:2008 (IEEE Std 12207-2008), *Systems and software engineering — Software life cycle processes*, and ISO/IEC 26514, S*ystems and software engineering — Requirements for designers and developers of user documentation* (also available as IEEE Std 26514-2010, *IEEE Standard for Adoption of ISO/IEC 26514:2008, Systems and Software Engineering — Requirements for Designers and Developers of User Documentation*) and others in the ISO/IEC 2651*n* family of International Standards. It provides requirements and guidance to technical writers and related roles on how to adapt the processes described in the ISO/IEC 2651*n* family of International Standards to develop quality user documentation.

This International Standard is independent of the agile development methods and tools that are used to produce the software.

This International Standard will conform to ISO/IEC 12207:2008 (IEEE Std 12207:2008) as an implementation of the user documentation part of 6.1: Documentation. The primary references for this International Standard are ISO/IEC 26514:2008 and ISO/IEC 26513:2009.

Systems and software engineering — Developing user documentation in an agile environment

1 Scope

This clause presents the scope, purpose, organization, and candidate uses of this International Standard.

This International Standard supports the interest of technical writers and associated roles responsible for producing user documentation for software and systems developed within an agile environment. This International Standard takes a process standard approach to specify the way in which user documentation can be developed in agile development projects.

This International Standard provides requirements on information management and documentation processes appropriate for software projects that are using agile development methods.

Clause 5 covers the overall requirements for documentation in the software life cycle.

Clause 6 describes how the information development lead or project manager may plan and manage the user documentation development team in an agile environment.

Clause 7 covers the relationship between the user documentation process and life cycle documentation process in agile development.

This International Standard is intended neither to encourage nor to discourage the use of any particular agile development tools or methods.

This International Standard provides guidance on processes appropriate for developers of user documentation in software and systems projects that are using agile development methodologies. It will not be limited to the development phase of the life cycle of user documentation, but includes activities throughout the user documentation life cycle.

This International Standard is intended for use in all organizations that are using agile development, or are considering implementing their projects using these techniques. It is assumed that users of this International Standard have experience or general knowledge of traditional user documentation processes.

2 Conformance

This International Standard may be used as a conformance or a guidance document for projects and organizations claiming conformance to ISO/IEC 15288:2008 (IEEE Std 15288-2008), *Systems and software engineering — System life cycle processes* and/or ISO/IEC 12207:2008 (IEEE Std 12207-2008), *Systems and software engineering — Software life cycle processes*.

2.1 Application of conformance

Throughout this International Standard, “shall” is used to express a provision that is binding, “should” to express a recommendation among other possibilities, and “may” to indicate a course of action permissible within the limits of this International Standard.

Use of the nomenclature of this International Standard for the features of agile methodology or the parts of user documentation (that is, scrum, sprint, chapters, topics, pages, screens, windows, etc.) is not required to claim conformance.

Conformance to this International Standard may only be claimed by an organization if all of the requirements in this International Standard can be met by the organization. When conformance is claimed for a multi-supplier program, it may be the case that no individual supplier may claim conformance because no single contract calls for all the required activities. Nevertheless, the program, as a whole, may claim conformance if each of the required activities are performed by an identified party.

This International Standard may be included or referenced in contracts or similar agreements when the parties (called the acquirer and the supplier) agree that the supplier shall deliver user documentation services in accordance with this International Standard. It may also be adopted as an in-house standard by a project or organization that decides to develop documentation in accordance with this International Standard.

Organizations, projects, or multi-supplier programs intending to claim tailored conformance should consult ISO/IEC 12207/IEEE Std 12207:2008, Annex A, Tailoring Process.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC/IEEE 24765:2010, *Systems and software engineering — Vocabulary*

4 Terms and definitions

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

4.1
agile development
software development approach based on iterative development, frequent inspection and adaptation, and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continuous stakeholder feedback

4.2
agile environment
organization or team implementing agile development methods and approaches

4.3
audience
category of users sharing the same or similar characteristics and needs (for example, purpose in using the documentation, tasks, education level, abilities, training, and experience) that determine the content, structure, and use of the intended documentation

NOTE There may be a number of different audiences for a software product's documentation (for example, management, data entry, maintenance).

4.4
burndown chart
document that records project status, usually showing tasks completed against total number of tasks

4.5
documentation
information that explains how to use a software product

NOTE In this International Standard, documentation is used to mean user documentation.

4.6
feature
functional or non-functional distinguishing characteristic of a system, usually an enhancement to an existing system

4.7
iteration
repetition of a process or activity

4.8
persona
archetypical user of a system, based on research into real users of a system

4.9
scrum
iterative project management framework used in agile development, in which a team agrees on development items from a requirements backlog and produces them within a short duration of a few weeks

4.10
scrum master
person who facilitates the scrum process within a team or project

4.11
scrum meeting
brief daily project status meeting or other planning meeting in agile development methodologies

NOTE The scrum meeting is usually chaired by the scrum master.

4.12
scrum report
report that documents the daily activities of a scrum team, recording any problems or issues to be dealt with

4.13
scrum team
members of an agile development team working together under the scrum process, usually led by the scrum master and project owner

4.14
sprint
short time frame, in which a set of software features is developed, leading to a working product that can be demonstrated to stakeholders

NOTE In some organizations, a sprint is known as an iteration.

4.15
use case
description of the behavioural requirements of a system and its interaction with a user

4.16
user story
simple narrative illustrating the user goals that a software function will satisfy

4.17
writer
person designing or developing user documentation

5 User documentation processes in an agile environment

5.1 Relationship between user and life cycle documentation processes

Teams using agile development shall perform the following activities:

a) identify documents to be produced by the process or project;

b) specify the content and purpose of all documents and plan and schedule their development and production;

c) identify the standards to be applied for development of documents;

d) develop and publish documents in accordance with identified standards and in accordance with nominated plans;

e) maintain documents in accordance with specified criteria.

The following information items are used in documentation developed using both traditional and agile development methods:

- description;
- plan;
- policy;
- procedure;
- report;
- request;
- specification.

NOTE 1 ISO/IEC/IEEE 15289:2011 defines the purpose and generic content of these information items.

The software implementation processes are the same between projects using traditional and agile development methods, but some or all of these stages may be repeated in each sprint. In some projects the software detailed design process may be shortened or sparsely documented. Code development may proceed through the development of working prototypes, rather than a detailed specification of the design being created and approved.

NOTE 2 Annex A contains a brief overview of agile development practices which the information developer may encounter.

Software Implementation Processes

1. Software Implementation
2. Software Requirements Analysis
3. Software Architectural Design
4. Software Detailed Design
5. Software Construction
6. Software Integration
7. Software Qualification Testing

Figure 1: Software Implementation Process

NOTE 3 ISO/IEC 12207:2008 (IEEE Std 12207-2008), *Systems and software engineering — Software life cycle processes* defines the software life cycle processes for a project.

5.2 Life cycle software documentation in an agile environment

Designing, developing, and testing user documentation is greatly assisted by the presence of life-cycle documentation such as a documentation plan, system design document, system test plan, release records, and problem reports. Other formal documentation specific to the user documentation development process may be produced, such as style guides and organizational procedures for content management and documentation reviews.

ISO/IEC 15289:2011, *Systems and software engineering — Content of life cycle information products (documentation)* provides recommended contents for the production of required documents throughout the systems/software life cycle.

In projects using agile development, any life-cycle documentation that is included in the project is likely to be less detailed, and possibly less formal than in other types of development projects. Some documents, for example, detailed specifications and design documents may not be produced at all. Because of the focus on delivery of working software, not only are some of the documents that traditionally would be produced as a part of the software life-cycle process not being produced (or are significantly reduced in content), but some of the processes may be skipped altogether. For example, the development team may proceed straight from producing a high level architectural design to software coding and testing, skipping the production of a detailed design.

Communication of the intent and behaviour of the user documentation may instead be provided by face to face communication, rather than through the use of detailed, formal documentation plans.

The documentation that is produced and the level of detail within each document are likely to be project specific. The level of detail may be influenced by the size of the team, the location of the team, requirements of acquirers, and other contractual agreements. More substantial documentation is needed if the team works in different time-zones or locations. Small, collocated teams may prefer minimal documentation and a reliance on face to face communication, whereas large, multi-location teams are likely to require more detailed documentation for communication purposes and future reference.

The types, level of detail, and timing of the production of the documentation will vary between projects. When the focus is on the delivery of working code, a development team may not have planned to include the resources needed to produce large quantities of documentation. Mechanisms are still required to ensure that the software product and associated user documentation match the user requirements; however these may be defined on the project.

Projects using agile development may benefit from the introduction of alternative content storage systems, such as wikis, that enable content to be captured quickly and cheaply. These allow information to be rapidly updated, and to be shared across the development team, including the technical writers, both locally and across multiple sites and time zones. However, wikis tend to collect chronological information without structuring the content for usability.

5.3 Life cycle documentation in agile development

Life cycle documentation should be produced in projects using agile development to communicate the processes, requirements, and deliverables required of the teams working on the project. These documents may contain less detail than their counterparts in other software development methods. The life cycle documents produced by projects using agile development are named the same as in other software projects, but the amount of detail or specific contents may differ.

The life cycle documentation items may not be formal or highly detailed documentation, but they are still useful in developing the user documentation. These documentation items should be produced by projects using agile development to assist both the production of software and user documentation that meet requirements:

- project plans;
- sprint plans;
- requirements documents, (expressed in user stories, scenarios);
- high-level design proposals, (may not be needed for agile development);
- test plans, (test procedure);
- risk statements, (risk register);
- user stories;
- use cases;
- descriptions of personas;
- burndown charts;
- task lists;
- scrum reports;
- end of sprint lessons learned reports.

6 Management of information development in an agile environment

6.1 Documentation management considerations for agile development

Agile development is an iterative and incremental approach to software development performed in a highly collaborative manner by “self-organizing” teams. There are many specific agile development methods. Most promote development sprints, teamwork, collaboration, and process adaptability throughout the life cycle of the project. Agile development methods frequently discourage the creation of detailed engineering support documentation and detailed technical specifications. This means that technical writers often do not have source documentation from which to extrapolate feature details.

Developers using agile development often prefer to work as “self-directing” teams, rather than fulfilling directives of external management. Team members may fulfil several roles from day to day. Agile development teams may expect that the technical writers and other staff involved in the development of user documentation will perform other roles, such as software testing. Nevertheless, except in very small organizations, the agile development team members receive assignment to the teams, provision of resources, and approval of priorities from executive management.

The short time span of sprints means that participation of each team member is essential. In particular, unavailability of the information developer on the team may make it impossible for documentation to keep pace with software development. The information development lead or project manager should decide whether the technical writer will be replaced by another technical writer from the documentation team or pool, or whether one of the other members of the agile development team will perform the technical writer role. If another team member who is not a technical writer by profession covers the technical writer on the project, arrangements should be made to educate that team member in the skills required, and apply special attention to the peer review process.

In large organizations, technical writers may belong to a centralized documentation team and may be shared between multiple development organizations. The centralized documentation team may be responsible for the production of documentation that may be shared across products, such as overview and concept information, but is not tracked within the agile development teams.

6.2 Change management when moving to an agile development process

Information development leads or project managers should understand how agile development processes will be applied, and work with the designers and product managers to understand roles and responsibilities while using agile development processes.

Agile development is not suitable for use in all projects, and its suitability may differ for specific types of organizations and products.

The information development lead or project manager shall agree with the designers and product managers whether the documentation team will follow agile development methods, and integrate with the development organization, or whether they will continue to operate using other methods. For example, some projects may use agile development for the production of online help, but not printed documentation. The documentation team should be educated on agile development practices before their organization moves to agile development.

The information development lead or project manager or another appropriate member of the documentation team should be involved in defining exactly what the new processes will be for the organization, and negotiate how agile development will affect their processes and team members. The representative from the documentation team should then communicate the plans to the wider documentation team, and act as a facilitator in the first scrum meeting that plans the project.

Plans should be made to provide education on agile and iterative development to all the development teams on the project, including the documentation team.

6.3 Composition of agile development teams

Agile teams are usually multidisciplinary, and focus on individual feature development, so that a single team may be responsible for one or more features and be comprised of an architect, developers, testers, technical writers, user representatives, and other stakeholders. This differs from other software development environments, where teams are often pools of single disciplines such as separate teams for developers, functional testers, system testers, technical writers, and then other individuals such as usability practitioners and graphic designers.

In agile development teams, different individuals may be expected to perform multiple roles. For example, developers may also do testing, in addition to unit testing and peer review. Testers may help the developers to fix defects in the code. This role sharing may depend on the requirements of the project or the availability of resources. Members of the agile development team who contribute to the development of the user documentation should be trained in the organization's purpose, identified audience, structure, and scope for user documentation.

6.3.1 Communication in agile development teams

Effective communications on the agile development team are key to the success of an agile development project. Because the communication in agile development is face to face rather than through the use of detailed life cycle documentation, information developers should be a part of the agile development teams from the beginning of the sprint, as well as during the documentation of future changes for the project backlog.

6.3.2 Teams working in different locations

When team members are not co-located, sufficient methods for establishing effective and reliable communication shall be put into place. These may include the following communication methods:

- video conferencing for meetings;
- teleconferencing for meetings;
- web meetings;
- one-to-one communication between the technical writer and developer.

The following communication products may also be used to facilitate effective communication:

- design/feature documents for sharing information;
- use cases;
- user stories;
- collaborative document repositories, for example wikis and databases.

When members of the agile development team work in different time-zones, or do not speak the same language as the primary language of communication, special communications accommodations are needed. Temporary assignment to the main location for the agile development team for remote team members may help solve communication issues, and establish better communication and relationships between team members when the team members return to their remote locations.

A lack of specification documentation and detailed designed documents means that developers of user documentation may need to use informal interviews for collecting information.

NOTE 1 Annex B provides an example questionnaire for interviewing technical experts.

NOTE 2 More information about interviewing technical experts and other contacts may be found in ISO/IEC 26514:2008.

In projects where the detailed specification documentation is not produced, alternative methods of information collection from the development organization shall be agreed upon by the user information development lead or project manager. If the technical writers and other roles involved in the development of user documentation have difficulty getting the information they need to write the user documentation, or produce other materials associated with the user documentation, the issues should be raised with the user information development lead or project manager, and at the appropriate scrum meetings and project management meetings.

6.4 Management of information development across teams using agile development

In an agile development project there may be multiple teams using agile development working on different features of the software. A risk with writers working independently from each other within the agile development teams is that each team creates different user documentation products that do not fit together when the product is brought together as a whole. To mitigate this risk the documentation manager or project manager shall ensure that sufficient time is provided for analysis and design to be carried out for the user documentation design and development to specify the structure, content, format, and style to be used in the documentation. The documentation manager or project manager shall also specify the standards, conventions, and style guides for the writers to follow when they design and develop the user documentation.

6.5 Management of tasks across sprints

In a project using agile development, ideally the user documentation is developed in parallel and to the same schedule as the software. This enables software to be regularly released to customers with sufficient documentation. Providing the user documentation as early as possibly within a sprint provides time to review and test the documentation in parallel with the software. The benefits for the technical writer of developing the user documentation in parallel with the software are:

- early access to features under development;
- communication with the developers while the features are under development;
- use of the function at an early stage to find and report defects;
- potential contribution to the design and usability of features.

6.5.1 Planning the project as a whole

In an agile environment it may be difficult to produce detailed plans beyond a single iteration at a time. However the project management should produce a project plan that includes the sprints to be developed and any milestones such as dates to provide the software and user documentation for manufacturing and translation schedules.

Within each agile development team, a high level overview of the planned content of each iteration should be developed. These plans are likely to be subject to future change because the inclusion of new user requirements may come into the project for inclusion in future releases at any time.

Not all user documentation work will necessarily take place in a single iteration, for example, work related to an information architectural strategy may need to be completed over multiple sprints. Tasks that fit into the bigger picture need to be considered and planned in addition to tasks defined for the individual sprints. Some tasks may be done in special sprints, or may be done at other set points in a project’s life cycle that may have an impact on the user documentation, for example, acceptance test, translation, and accessibility testing. There may be sprints which focus on consolidation or infrastructure tasks which do not require concurrent user documentation development. These sprints may produce other deliverables, such as infrastructure improvements, which may involve the technical writers or other members of the documentation team, such as build members.

There may also be user documentation work that needs to be performed that is not associated with the software feature produced in an agile development team, for example, a user documentation backlog or introductory information. Some types of information can be difficult to develop in individual sprints, especially user documentation related to the project as whole. Examples of user documentation that relate to the project as a whole may be installation and configuration guides, tutorials, high-level concepts, reference, and troubleshooting documentation. This work shall be included in the planning by the user information development lead or project manager, and the user information development lead or project manager shall ensure that these tasks are resourced and tracked as a part of the project as a whole. Ideally these tasks are determined at the beginning of the project in the project planning stage, and are then allocated and drafted as early as possible in the cycle. It may be possible to schedule the development of these documentation items at the beginning of a project between the acquisition of requirements and the start of code development, sometimes known as iteration zero, or during iterations dedicated to testing or quality assurance. The documentation produced for these items may need to be reviewed at the end of the sprints to take account of any changes during the sprint.

The information development lead or project manager should reserve time before the end of a release cycle, for example, at least part of the last sprint, for production, and any final production edits. The information development lead or project manager should set a documentation freeze date in the schedule, and ensure that technical writers and other members of the agile development teams are aware of the freeze date and production schedule. Technical writers should commit only to what they may contain before the documentation freeze date and communicate that date to their development teams, prioritizing work with them if necessary.

The information development lead or project manager should determine how much user documentation is required and the minimum requirements for the inclusion of task, concept, reference, and troubleshooting information for the user documentation to be usable and ready to release to customers. The considerations of the minimum documentation requirements that must be complete before the product and user documentation can be delivered to customers should be included in the planning for the project as a whole.

6.5.2 Sizing and resourcing each sprint

Planning for the project requires planning the content for individual sprints. The user documentation produced with the product function in each sprint is likely to be mostly task orientated information, although it may also be possible to schedule feature-related concept and reference materials within the individual sprints. Planning for each sprint should include identifying tasks based upon the features to be completed within that sprint. Development of concept and reference materials may be easier to schedule at the beginning of a sprint, while task based information may be easier to develop when some coding or prototypes have been completed.

Planning for each sprint should include tasks for development, adequate review, and testing of the user documentation, as well as the effort for coding and testing the software. The technical writer, information development lead, or project manager should ensure that the details and sizings for the user documentation work are included in the plan for each sprint. The technical writer, information development lead, or project manager should also ensure that sufficient time is included in the sprint length and work item sizing to cover review and test of the user documentation. Time should be provided for testing of the documentation with the working software, but it may be appropriate to schedule this testing in a separate consolidation or user acceptance sprint. Time should also be included in the plans for updates to the documentation as a result of defects or problems found during review and test.

It may not be practical to provide complete and thorough user documentation for a software feature developed in a single iteration. Therefore planning the user documentation to accompany the software feature should focus on the most essential user documentation required to support the user, based on the intended audience. In some agile development teams, planning includes core and stretch items. Instructional information that is required for the user to be able to use the software shall be part of the core items, but supporting user documentation, such as overviews and concept documents may form part of the stretch items. The core items should be documented as a priority, and the technical writer shall ensure that any agreed items that are not completed are carried over into the next iteration.

6.5.2.1 Sizing information development work

The information development lead or project manager or technical writer should size the user documentation tasks based on estimates from previous development activities, considering the potential for reuse or adaptation of existing material

NOTE The complexity of the software development effort may not directly relate to the time and amount of detail required in the user documentation.

A size should be provided by the technical writer for each task that they are responsible for in the sprint. Depending on the sprint the technical writer should break down large tasks into smaller tasks. Tasks that are sized above 3 to 5 days may be difficult to adequately estimate in a sprint, and may result in slippage if the task cannot be completed in the sprint. For short sprints, for example, 1 or 2 weeks, tasks sized at over 1 day should be broken down into smaller tasks for the purpose of sizing.

The technical writer shall ensure that any work that is not completed within the current sprint is either carried on to the next iteration, or is removed from plan if it is determined to not be required. A functional item is not completed until:

- code writing has been completed;
- user documentation writing has been completed;
- both code and user documentation have been tested;
- any issues identified from testing have been resolved or a plan exists to resolve them.

6.5.2.2 Assigning information development resources

In an ideal agile environment, the user documentation work for each user story will be completed in the same sprint as the development and testing work. The information development lead or project manager shall list and size the user documentation tasks before the start of each sprint. If the estimates exceed the resources for the sprint, the technical writer should coordinate with the team to prioritize the tasks and, if necessary, schedule the overflow for the following iteration.

In some projects there is a single agile development team, and in others there may be multiple agile development teams. The size of the project using agile development and the relative size of the technical writing team should determine exactly how the technical writers interact with the agile development teams. The number of technical writers assigned to an agile development team should be decided by the user information development lead or project manager based upon the expected resource requirements for the project. This number may vary between sprints.

If a technical writer is shared across multiple agile development teams, a percentage of the writer's time should be allocated to each team, so that their availability and utilisation may be measured as a part of tracking in each team.

When committing resources to an agile development team, the information development lead or project manager should consider how much time the technical writers are likely to spend in meetings. These meetings include regular scrum meetings and also other meetings where they may elicit information for the production of the user documentation such as design meetings, demos, and test reviews. Schedules should be arranged so that the information developer can participate regularly in scrum meetings for each assigned team.

6.5.3 Handling last-minute documentation changes

The information developer should establish a process with the extended team for handling last-minute documentation changes that come in too late for that sprint. For example, the team agrees that changes received after a certain point in the sprint will be deferred to the next iteration. Or, if deferral is not an option, the team decides to contribute to writing the information that is required for the sprint.

6.6 Monitoring and analysing progress in an agile environment

Tracking progress is an important part of agile development, to ensure that the committed items are successfully completed in the sprint, and any potential problems or holdups are discovered and resolved as early and rapidly as possible.

6.6.1 Status meetings

Regular status meetings, known as stand-up meetings or scrum meetings, are common to many agile development methodologies. These meetings are usually designed to find out what tasks each member of the team has completed, what task they are going to work on next, and what problems they may have. These regular status meetings are intended to be short meetings, for example, for a maximum of 15 minutes depending on the size of the agile development team. The meeting leader, sometimes termed a scrum master, maintains scrum reports, and helps to solve any problems that are blocking progress of the sprint.

Regular status meetings should be held daily or as often as required. Virtual team meetings may also be held daily or as often as required for members who are not collocated.

The information development lead or project manager shall ensure that the members of the documentation team attend status meetings regularly. The information development lead or project manager shall also attend if identified as a stakeholder. The technical writers should report their status to the attendees of the meeting. If they are unable to attend they should arrange to provide their status and raise any issues through another means, such as through a team document repository, for example, a wiki, or via email direct to the meeting leader.

Sometimes these regular meetings also include demonstrations of the software code, or the user documentation that has been produced, and may be an effective way of communicating changes, or performing peer reviews or walkthroughs of both the software and the user documentation.

The information development lead or project manager should ensure that the members of the documentation team are getting the most value out of the status meetings. These meetings should be useful forums for communication and raising concerns, but should not be focused on in-depth discussions of technical implementation for example. If the meetings are frequently diverting away from the status of the project team, or not providing information that the technical writer requires, the information development lead or project manager should work with the meeting leader to focus the meetings.

6.6.2 Monitoring progress

In addition to the regular status meetings, tracking of a project using agile development is usually done through the listing of the status of the individual tasks for the sprint. These tasks include both information development tasks related to the development of new function, and also additional information development tasks, such as the product of concept and reference information. All of the user documentation tasks for the sprint should be listed, such as:

- topics to be developed;
- topics to be reviewed;
- topics to be tested;
- additional documentation to be developed;

EXAMPLE Error lists, troubleshooting, reference, concepts, and tutorials.

- additional documentation to be reviewed;
- additional documentation to be tested;
- translation or localisations packages;
- embedded assistance to be produced;
- infrastructure activities;
- reviews and tests.

Associated with these tasks should be the sizing estimate for the task, and the current status:

- not started;
- in progress;
- under review;
- completed.

It may be appropriate to record a value for percentage complete for each task, depending on the granularity of the task.

Team members should record the actual effort expended to allow improvements in sizing algorithms based on actual data.

6.6.3 Rework and changing requirements

The scrum master and information development lead or project manager should provide guidance to the technical writers and other members of the agile development teams on how to handle changing or new requirements. One of the key differences from the technical writers' point of view between agile development and traditional development methods is that they spend time in rework or rewriting of the documentation, when changes are made to the design as a result of problems discovered during development or new requirements. Although this cannot be completely mitigated, developing the user documentation in parallel with the code, and working directly with the developer on the specific details may help.

Consideration that rework may be required should be included in the documentation planning for the product and individual iterations. Providing the documentation for review in a different manner than the final format for delivery of the documentation, for example, as soft copies rather than printed materials, may assist in leaving costly decisions and production to the last minute, after the majority of changes required have already been made. The effort and expense of formatting and producing final user documentation should be deferred from interim reviews until the documentation content and the software are considered stable for delivery.

Storing user documentation plans and schedules in a place which makes them easy to update, without needing them to go through lengthy review cycles may help make it easier to include changes and new requirements to the plans.

6.7 Stakeholder involvement

Stakeholders are individuals or organizations who have an interest in the software development project. Stakeholders may be directly involved in the projects, or their interest may be affected as a result of the project. The project management team should be responsible for the identification of stakeholders, and determine their involvement with the project. Stakeholders directly involved in the project include project managers, architects, developers, testers, and documentation developers. Stakeholders who may or may not be directly involved in the project may be users, sales personnel, infrastructure and operational system administrators, and other personnel related to the development of user documentation such as the information architect, graphic designer, support personnel, or the user information development lead or project manager. Often user involvement is considered essential in approving sprint content from the backlog and approving each release.

In agile development, the involvement of stakeholders is essential to the success of the project. The individual stakeholders are directly involved with the development of the software and have the opportunity to influence the project and introduce new initiatives. Therefore, the roles and responsibilities of the stakeholders in the project shall be defined, and shall be communicated to the stakeholders.

The involvement of stakeholders is important for the user documentation process because it provides opportunities for the documentation team to get customer requirements, customer priorities, and customer feedback. Writers should have the opportunity to actively participate in meetings or other communications with stakeholders in order to clarify requirements and to receive feedback.

6.8 Improving the user documentation process in an agile environment

The iterative process in agile development provides the opportunity to frequently monitor and improve the processes used in the development organization. At the end of each sprint, the agile development teams including the user documentation development team members should discuss the successes and failures of the previous sprint in order to identify and implement process enhancements or improvements.

Best practices and successful behaviours should be shared across the documentation team. Any problems related to the production of user documentation or any problems encountered by the technical writers during the sprint should be highlighted. These problems and their remedies should be discussed by the agile development team and the information development lead or project manager. The information development lead or project manager should work with the agile development team lead or product managers to resolve the problems, and improve the process for the next iteration.

Any problems that occur within an agile development team during a sprint do not have to wait until the end of the sprint to attempt to improve them or prevent them happening. The information development lead or project manager should encourage the members of the documentation team to report any difficulties promptly, and seek assistance in solving them if required, either through the status meetings or directly to the manager.

6.8.1 Assessing customer satisfaction

Stakeholder and user involvement is a key part of the agile development process. Both working code and user documentation is usually expected at the end of each sprint. Customers may try the software and supporting user documentation and provide feedback.

The information development lead should work with the project manager and user representative to seek opportunities to acquire feedback from customers through methods such as review comments, beta programs, and usability testing.

Customer feedback may lead to new customer requirements to include in future iterations of the project. These customer requirements may include additions, changes or updates to the user documentation.

7 Developing user documentation in an agile environment

7.1 What agile development means for information development

In agile development it is important that the development of the user documentation is part of the same processes as the software product life cycle, and performed in conjunction with development of the software. This enables the software and the user documentation to be tested, distributed, and maintained together. In agile development, the software cannot be considered complete without the production and validation of the associated user documentation.

Agile development processes may impact the writers of user documentation in the following ways:

- allows for involvement early in the development process;
- allows for influencing the software design, particularly of the user interface;
- allows for close interaction with developers, testers, usability, and other development roles;
- minimize duplication of documentation across the development organization;
- develop only the documentation that the user wants and needs;
- receive user feedback on the software and the documentation early in the development cycle;
- receive user feedback continually enabling the work load to be spread throughout the development cycle;
- have access to stakeholders representing the user.

7.2 Product design and developing the user documentation

User stories, personas, and use cases should be developed in the requirements and high-level design stage of the project. These documents may help inform the technical writers and other members of the agile development team about the purpose, users, and externals of the software. The writers should be provided with access to these documents.

In order to minimize unnecessary work and duplication of effort, technical writers may be asked to produce or contribute to life cycle documentation. The user documentation may itself become the design specification for the product under development, where the design details the external design features applicable to the user, such as the user interface and steps to use it, and not the internal implementation of the software. Details of the external design and information about why and how the user should use the feature are required for development of code, testing, and production of the user documentation. This approach may have other effects, such as changing the order in which user documentation is produced or tested. For example, the user documentation may exist before the code is implemented, and both may be tested at the same time earlier in the cycle in conjunction with feature/system test, rather than waiting until the end.

In projects where no design specification documents are created, the user documentation itself may become the source reference for the design and purpose of the software.

There are a number of techniques that may be used to develop the user documentation without the use of design specifications, including:

- pair programming with the developer and technical writer;
- interview model with the technical writer asking specific questions;

NOTE An example set of questions is provided in Annex B.

- prototype or software walkthroughs;
- regular reviews and testing of the documentation produced for the iteration.

Many of these methods require direct communication between the technical writer and the developer, either face to face, or via virtual conferences. The user documentation may act as a repository for information acquired through direct communication, and the iterative approach to development should result in no need to return to user documentation produced in an earlier sprint, except where it is required to be updated for new features or defects in a later iteration. It may be useful to record some of the information exchanged for tracking purposes, or for changes in personnel. For example, meeting minutes may be taken, outstanding issues may be documented, and walkthroughs may be recorded, and should be kept in a content management system in accordance with the organization's standard procedures.

The primary way to ensure that the user documentation under development is up to date with the software design is to ensure regular communication between the technical writer and the other team members. The technical writer shall attend both regular scrum meetings and design meetings to ensure the writer is aware of any updates and to have the opportunity to express concerns, and ask for assistance if required. If the technical writer is remotely located from the scrum team, then arrangements should be made to allow the writer to participate. If the writer cannot attend a team meeting, a representative in the meeting shall raise concerns on behalf of the writer and ensure that the writer receives the information conveyed in the meetings.

The agile development team shall use a controlled source location for published designs and high level design documents. Comments and discussion around the design should be included with the published designs. Ideally the comments and discussion are integrated with the published designs through iterative updates to show how the discussions have changed the design. At the beginning of each new sprint, when an existing design is updated, the latest version of the design should be used as the base design for the new iteration.

Technical writers should be able to access both the current design documentation and the software prototypes to verify that the user documentation they are developing is correct for the sprint. Interviews with the developers and review of change logs of the design documention are also helpful.

7.3 Design and development of user documentation in an agile environment

User requirements should form the basis for all code and user documentation produced in an agile development project. Most of the user documentation produced will be associated with code, but there may be separate features that relate to specific user requirements that relate only to the user documentation.

User requirements should be documented, and the project plans and user document plans should make reference to these user requirements, so that new or changed features may be tracked back to the user requirements to which they are related.

The user requirements should be used to create design documents such as user stories or scenarios. User stories or scenarios should be created for tasks that are user documentation only, for example, an overhaul to the current information navigation, or the production of some new documentation guides. Design documents such as user stories and scenarios may help the agile development teams understand the value of these tasks for the users, and provide a mechanism for developing testing to validate the tasks, as well as providing guidance on how to complete the tasks.

NOTE ISO/IEC 26514:2008, *Systems and software engineering — Requirements for designers and developers of user documentation*, specifies the structure, content, and format for user documentation, and also provides informative guidance for user documentation style

7.3.1 Design techniques

A number of techniques are used in agile development to help design both software and user documentation based on user needs and requirements. These techniques may be used as part of a formal design phase at the beginning of a project between the acquisition of requirements and the start of code development, sometimes known as iteration zero. Alternatively these techniques may be used at the beginning of each sprint for the design of individual features.

7.3.1.1 Use cases

Use cases describe the sequence of interactions between actors and the system needed to deliver the service that satisfies the goal. Use cases also decide what the system will be used for before defining what the system is supposed to do. Use cases are a valuable source of information for a writer because they provide information about how a user is expected to use the system, what options the user has, what goals the user is trying to achieve, and what error conditions may occur.

A use case records:

- who (actor) does what (interaction) with the system;
- for what purpose (goal), without dealing with system internals.

A complete set of use cases specifies all the different ways to use the system, and therefore defines all functional behaviour required of the system. The system is treated as a black box; therefore, use cases describe the interactions with the system, including system responses, as seen from outside the system.

Use cases focus on the user's point of view and help ensure the feature provides value to the user. The use case is understandable by the different roles in the development team, as well as users and other stakeholders involved with the project.

The user documentation developed should support the user in completing the goals defined in the use cases to be developed for the iteration.

7.3.1.2 User roles

For any given product there is likely to be one or more user roles that will use that product. Each user role will have particular tasks they perform or goals that they want to achieve. They may also have a set of associated skills, and they may work in a particular environment.

User roles are useful sources of information for a writer because they provide information about the different types of users using the system. The user role descriptions may also provide information about what skills and requirements each role has, and this may inform the user documentation that is produced for each role.

Examples of user roles might be:

- web developer;
- systems administrator;
- technical support officer;
- lab assistant.

The user documentation developed should support the user roles that are appropriate for the iteration.

7.3.1.3 Personas

Personas are an extension of user roles in which the documentation developer creates a fictional character that represents a group of users. At a minimum each persona should include:

- character name;
- job title;
- experience;
- relevant skills;
- business goals.

In some instances a picture or image may lend additional relevance to the personas. The following information also adds substantial value to the persona:

- age;
- education;
- organization;
- major responsibilities;
- goals and tasks that the user wants to complete;
- the problem or business opportunity that the product solves for the user's organization;
- the user's physical and social environment;
- the software and hardware platforms the user is using;
- size of the organization.

The information provided in the persona description may assist the writer in producing user documentation for different user roles. As well as providing information about typical user goals and skills, the personas may be used to help design questions and tasks that may be used in testing the documentation.

7.3.1.4 User stories

User stories are important for determining, describing and prioritising customer requirements and are usually expressed in terms of feature development. User stories are a simpler and user-focused alternative to producing lengthy detailed requirements specifications. The lack of details and implementation specifics provides a greater scope for collaboration, and changes being made to the implementation and design later in the life cycle.

The user stories define how the software will work for the users, and provide an important source for the design of the software according to user needs, how the software will be tested and what the user documentation should contain.

User stories are collaborative documents to be used by multidisciplinary teams including developers, testers, documentation developers, and other stakeholders. User stories may be written by any member of the software development organization, including the intended users of the software. User stories should also be written for user documentation only features, for example, the production of a glossary, or making changes to an information set to enhance the retrievability of the information.

User stories differ in their level of complexity depending on organizational requirements, but they should always be expressed in business terminology that can be understood by all stakeholders, not just by the development organization.

The main content of the user story often takes the form: “As a (role), I want (goal), so that I can (business value).” For example, “As a network administrator, I want to be able to check the status of groups of network devices with shared characteristics so that I can identify issues early and reduce administrative efforts.” Acceptance criteria are developed to identify when the user story has been implemented, and the user will be able to achieve their goals. For the previous example, the acceptance criteria might include:

- view the current status of one or more network devices;
- group resources by network;
- group resources by vendor;
- groups resources by service;
- group resources by status.

A user story should contain the following information:

- the user role that the story applies to;
- the goal that completion of the story allows the user to achieve;
- the value to the customer;
- acceptance criteria to identify when the user story has been implemented.

User documentation requirements shall be included as part of the acceptance criteria for a user story to ensure that the user documentation is included in the planning, development, and testing of the user story.

A user story may also contain the following information:

- a unique identifier for the story;

- additional detail, for example, the conversation between stakeholders and development, or the planning implementation;
- the name of the user story;
- the priority of the user story;
- the size of the user story.

If the work required to fully complete the implementation, testing, and documentation of a user story is greater than a few days, it should be broken down into smaller stories. Large user stories may be difficult to size and to successfully complete within a single iteration.

User stories are useful sources of information for writers because they provide information about user goals and about how the function under development will work. The user documentation developed should support the user in completing the goals defined in the user story to be developed for the iteration. Together the user documentation and software developed should meet any acceptance criteria requirements.

7.3.1.5 Scenarios

Scenarios extend stories by using a specified user. For example, a persona-based scenario may describe specific user requirements or ways that the user may use a product based on their existing knowledge and experience or environment. A scenario can be a very detailed account of how the user in the persona interacts with a system, or can be briefer and focus on high-level requirements. A story tends to focus on the so-called 'golden path' where the product performs exactly the task the user wants to perform, exactly as they expect it to, with no errors or unexpected behaviour from the user's point of view. A user wants to achieve a particular goal, and the steps for them to do that are likely to be considered. Problems with the golden path may be highlighted using scenarios, and alternate paths may need to be explored to support users with the same requirements as the persona.

User documentation requirements shall be included as part of the acceptance criteria for a scenario to ensure that the user documentation is included in the planning, development, and testing of the scenario.

A scenario may contain the following information:

- the persona that the scenario applies to;
- the goal that completion of the scenario allows the user to achieve;
- the specific needs of the user role defined in the persona;
- the actions the user takes in interacting with the system based on their specific requirements or skills;
- a unique identifier for the scenario;
- additional detail, for example, the conversation between stakeholders and development, or the planning implementation;
- the name of the scenario;
- the priority of the scenario;
- the size of the scenario.

The information provided in the scenario description may assist the writer in producing user documentation for different user roles and in support of the defined user goals. As well as providing information about typical user goals and skills, the scenarios may be used to help design questions and tasks that may be used in testing the documentation.

7.3.2 Information development tasks

The information development lead or project manager or scrum master should ensure that any information architecture or design work is front-loaded in the release, especially where there are major changes required. The information development lead, project manager, or appropriate stakeholder from the documentation team should seek commitment for these items from the agile development project manager and other stakeholders during the project planning stage.

These information architecture or design tasks should be included in the tracking for the project, and broken down into small sized tasks, and allocated to the appropriate resource. These tasks should be included in a small number of sprints in a release, unless an iteration is dedicated to completing infrastructure or architecture type tasks.

The information development lead, project manager, or technical writer should work with the agile development team to identify specific sprints that will focus on user documentation related issues, such as integration, usability, and translation.

7.3.3 Planning of information units

Planning of information units is completed during each individual sprint. The technical writer shall determine which information units are produced in each sprint. Having identified the information units to be produced, development of them may continue. The technical writer should identify requirements that they have from the agile development team to complete their work. For example, the technical writer may require access to the prototypes or time with the developer to acquire the details of the design of the software.

It may be useful in some agile development teams to offset the development of the user documentation by a week or two, thereby delaying the actual documentation work until more stable code is ready. Offsetting may also give the code developers a better chance to be involved in the user documentation testing because they tend to have less work at the end of a sprint compared to other members of the team. The team should balance the offset with the need to provide information units at the appropriate time for testing.

The amount of any offset will be directly affected by the duration of the sprints. Offsetting cannot be used in short sprints, because there will not be enough time for user documentation development and testing.

If the user documentation is provided to customers at the end of each iteration, it is essential to ensure that time is included for editing and testing within the sprint, and that any late changes may easily be incorporated.

In projects where the product and user documentation are not provided for every iteration, a final consolidation iteration may be scheduled, which includes time for final review, testing, and defect fixing on the software and user documentation.

7.4 Testing and reviewing documentation in an agile environment

In all projects, reviewing and testing the user documentation is important in assuring the accuracy, completeness, fitness for task, and consistency of the user documentation.

NOTE ISO/IEC 26513:2009, *Systems and software engineering — Requirements for testers and reviewers of user documentation*, provides requirements and guidance for reviewing, system testing, usability testing, accessibility, and localisation testing of user documentation.

7.4.1 Reviewing user documentation

At a minimum, the agile development team shall peer review all user documentation for adherence to style guidelines and technical accuracy. Other technical writers, editors, or other members of the team with strong communication skills shall conduct the peer review; developers and testers shall review the user documentation for technical accuracy.

Reviewing tools whereby members of the agile development team may provide comments directly to the individual documentation items are helpful in an agile environment.

Another effective method of reviewing user documentation is to hold a walkthrough meeting with the appropriate stakeholders in the team to review the structure and content of the information. Changes and updates may be made to the user documentation either during the meeting, or after the meeting.

Changes or updates required shall be made before the end of the sprint. The user documentation shall be available in draft form and available for review a sufficient time before the end of the sprint to enable the technical writer to make the corrections or additions.

The user documentation shall be approved as complete by the agile development team before the sprint may exit, and the next iteration started.

NOTE ISO/IEC 26513:2009 includes more information about review of user documentation.

7.4.2 System test of documentation

If the project is determined by the project team lead or by the project manager to require a system test effort, the system test plans created by the test lead or the tester shall include plans for testing the user documentation. If system test is included in each sprint, the plan for the system test of user documentation should be established for each sprint. If separate sprints are used for system testing, the plan for the system test of user documentation should be included in the planning for the system test sprints. The system test activities to test the user documentation shall be included in the test plans, along with details of the information products to be tested. These plans shall be reviewed and approved by the technical writer or information development lead or project manager.

Ideally the system test of the user documentation is carried out in parallel with code development and user documentation development in each sprint.

Time should be included in the sprint for corrections to be made to the user documentation after system testing has been completed.

One effective strategy for incorporating changes and updates rapidly into the user documentation during system testing of the documentation is to use a pair programming approach. Using a pair programming approach, the technical writer responsible for the user documentation under test sits with the tester while the tests are performed. The technical writer can make changes to the user documentation on the fly as the tester identifies the problems.

Where this type of activity is not possible, a problem report shall be created by the tester and stored according to the problem resolution standards of the organization.

A further review or test cycle may be required to ensure that updates to the user documentation are correct.

More information about system test of user documentation may be found in ISO/IEC 26513:2009.

7.4.3 Usability testing of user documentation

Usability testing of user documentation shall be carried out using real or representative users. Documentation usability testing is related to user acceptance and validation testing activities to determine whether the prototype design or the draft documentation being tested meets the users' needs. Documentation usability testing may be carried out at earlier stages of the life cycle, or when the software is ready for a release.

Usability testing that includes the user documentation may be carried out in each iteration, but at a minimum, there shall be one usability test of the documentation with a single user. If only one usability test is performed, then the user documentation under test should use the release version of the software. This test may be carried out as a part of a user acceptance iteration, or as part of a customer satisfaction assessment activity.

Feedback from additional usability testing of the product during sprints or customer satisfaction assessment activities shall be incorporated into the user documentation.

7.5 Translation and localisation of user documentation

If user documentation for the project needs to undergo translation or localisation, the timing of this activity needs to be included in the planning for the project. When the organization decides to move to agile development, it is necessary for the project managers to determine how translation and localisation fits in with the processes and sprints.

In order to market products to some countries or some organizations, it may be a requirement to have a translated or localized version available at the same time as a release in the local language/localisation. The agile development team should include these factors into their planning to keep the translated or localized versions up to date.

The information development lead or project manager shall plan the translation schedule for the user documentation. The following items should be taken into consideration in planning the translation schedule:

- whether the user documentation is required to be translated for each sprint;
- if the user documentation is released to the user each iteration;
- time to perform translation or localisation;
- time to package and handle returns of translated or localized documentation;
- time to test translated or localized documentation;
- local translation, local requirements, and regulations.

A strategy that some organizations adopt with translation and localisation is to package and send for translation at the end of each sprint, and incorporate returns into the following iteration.

7.6 Production for manufacturing cycles

If user documentation for the project needs to undergo additional processing or testing before release to customers, the time required for this activity and scheduling should be included in the planning for the project using agile development. For example, printed documentation may require time for preparation and printing; and online documentation may need to be provided on a CD, and the CD image may need to be tested prior to delivery to a manufacturing centre. The information development lead or project manager should ensure that the manufacturing time is included in the planning and schedule for the project using agile development.

Annex A
(informative)

Agile Development Practices

A.1 Methods and terminology used in agile development

Agile development is not a single process, or set of defined methodologies, but rather, a set of principles and related methodologies. Key features of these agile development methodologies are iterative development and the evolution of requirements and solutions over time through collaboration between self-organizing cross-functional teams. Different agile development methodologies have varying terminology. These differences in terminology associated with particular methods are not described by this International Standard.

Agile development methodologies focus on delivering functional value for users and decreasing costs.

Costs are cut by developing the high value requirements as a priority and finding defects early in the development cycle when it is cheaper to fix them.

Agile development methodologies encourage clear and regular communication and all the interested parties may be involved in all stages of development. Because agile development methods are focused on fixed time periods for development, the risk of going over budget is reduced. The timescale and cost of the project is fixed, but the scope and features are flexible.

Functional value is added for users by using iterative approaches where useful function is developed and delivered at the end of each sprint. At the end of each sprint a useful piece of function will have been developed, tested and provided with supporting user documentation. Because function is developed and validated rapidly, this function can be delivered earlier and updated regularly. Early release also provides opportunities to acquire user feedback and new requirements to feed into the next iteration.

Continuous validation through testing and stakeholder involvement provides frequent status on the progress of the product, including the level of quality, and fitness for purpose of the product.

The users or a representative for the users are involved in each stage of the development process, and therefore can check that value is being added, and the right requirements are being addressed. Development may stop if new requirements come in or changes are asked for, so that the work items may be correctly prioritized.

Agile development methods rely on the individual teams to decide how to develop and deliver the requirements that they are tasked with. Involvement from requirements to delivery encourages all team members to have the same knowledge and understanding about what needs to be achieved, what the real user requirements are that are being fulfilled, and how it will be done. Because the team works together, and has the same knowledge of the what, why, and how, there is less immediate need to produce and understand detailed specifications for both the project life cycle and tracking the status of the project. Any problems that occur may be discussed and either resolved quickly so that either progress can continue, or identified as a severe blocking problem that means that the current work is stopped in favour of moving onto something more productive.

Agile development methodologies also provide regular opportunities to take a look at how the team is working, or how the project is working, and make changes to make the process more effective and efficient.

A.1.1 Iterative development

Most agile development methods use some form of iterative development. Iterative development is a term used to describe repeated use of planning, developing, and testing activities. Agile projects have a planning and scheduling strategy, in which the various features of the product are developed to completion during a specified period of time, called a sprint or an iteration. A sprint is usually between two weeks and three months long. During this time the feature is planned, designed, developed, tested, and documented in the user documentation. During the next iteration, the same function might be improved or reworked, or new function may be added, based on user feedback.

Before the development team can move on to a new sprint, the development must be complete and stable: it contains not only working code and functionality, but usable documentation and a working user interface. At the end of the sprint, the feature and the product for which it is being developed is complete enough to deliver to users, as either a product or a beta.

A.1.2 Extreme programming

Extreme Programming (XP) is a software engineering methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile development it advocates frequent short sprints, which are intended to improve productivity and introduce checkpoints where new customer requirements may be adopted.

Other elements of Extreme Programming include:

- programming in pairs or doing extensive code review;
- unit testing of all code;
- avoiding programming of features until they are actually needed;
- a flat management structure;
- simplicity and clarity in code;
- expecting changes in the customer's requirements as time passes;
- frequent communication with the customer and among programmers.

A.1.3 Scrum

Scrum is a methodology for agile development, which features an iterative approach to development in combination with other practices including:

- developing user stories for user requirements;
- concise daily scrum meetings;
- scrum masters to chair meetings;
- producing burndown charts;
- adaptive sprint planning;
- sprint review demonstrations to share progress;
- sprint reflections and retrospectives to improve practices.

Annex B
(informative)

Example interview questions

The following questions are examples of questions that a technical writer should ask a developer to obtain information to include in the user documentation in the absence of a detailed design specification. The information obtained from these questions could also be used to develop a user story to help develop the user documentation.

1. Is user documentation required for this feature?

NOTE If not, add a note to the user story to record that no user documentation is required.

2. Do users need to be educated on the benefits of the implemented feature, or is the feature transparent to users? An example of a transparent feature would be a performance enhancement to an existing feature.

3. Do users need to know that this feature adds, fixes, replaces, or enhances product functionality?

4. Do users need a primer or background information about this feature before using it? For example, does this feature introduce or use new technology that most users would not be familiar with? Are there any new terms or concepts to which the user would not have previous exposure?

5. Are there any special permissions or user types required to use this feature?

6. Do users have to be aware of any cautionary information, equipment damage, data loss, service interruption/degradation, ESD protection, or backup requirements before or when using this feature?

7. Do users need to know the optimum or logical time to use this feature? For example, is there a time when users would derive the most benefit from using it; or is there an operational scenario where users would want to, or have to use this feature?

8. Do users need to know if any pre-conditions or existing tasks that need to be completed before users can use this feature? For example, do other network elements have to be in place or created or do users have to configure other parameters or attributes before the feature can be used?

9. Does this feature affect the product interface? For example:

   - does this feature add or change menu items, buttons, icons, tool tips, or warning messages?

   - does this feature add or change any interface parameters or options? If so, what are their ranges and default values, and when or why does the user need to change them?

10. Do users need to know how the feature gets installed? Are there any new installation prompts introduced with the feature?

11. Do users have to configure anything to make this feature operational after installation such as establishing communication to another network element?

12. Do users need to know that this feature might generate any of the following:

   - error/warning messages,

   - alarms,

- statistics,
- log files,
- problem reports,
- dump files,
- system-generated e-mails,
- notifications,
- reports.

13. Do users need any special skill set or tools before they can use or implement the feature?

14. Do users need to know of any situations where this feature may not work as design, or operational intent? If so, how will users determine this (such as warning or log messages) and how do they troubleshoot or recover from this?

15. Do users need to know if this feature requires periodic maintenance to remain within design limits, or to stay in an optimum state?

NOTE This question may be useful for a system administrator's guide or similar information item

16. Do users need to know if there is an Application Programming Interface (API) equivalent for configuring this feature?

NOTE This question may be useful for highly technical users and the product of a developer's reference guide or similar information item

17. Do users need to know of any tasks that need to be performed after users use the feature?

Bibliography

[1] Agile Manifesto, http:/www.agilemanifesto.org/

[2] ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*, 2004

[3] ISO/IEC TR 10000-1, *Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework*

[4] ISO 10241, *International terminology standards — Preparation and layout*1)

[5] ISO 690, *Information and documentation — Guidelines for bibliographic references and citations to information resources*

[6] ISO/IEC 12207:2008 (IEEE Std 12207-2008), *Systems and software engineering — Software life cycle processes*

[7] ISO/IEC 15288:2008 (IEEE Std 15288-2008), *Systems and software engineering — System life cycle processes*

[8] ISO/IEC/IEEE 15289:2011, *Systems and software engineering — Content of life-cycle information products (documentation)*

[9] ISO/IEC/IEEE 26511:2012, *Systems and software engineering — Requirements for managers of user documentation*

[10] ISO/IEC 26513:2009, *Systems and software engineering — Requirements for testers and reviewers of user documentation* (Also available as IEEE Std 26513-2010, *IEEE Standard for Adoption of ISO/IEC 26513:2009, Systems and Software Engineering — Requirements for Testers and Reviewers of Documentation*)

[11] ISO/IEC 26514:2008, *Systems and software engineering — Requirements for designers and developers of user documentation* (Also available as IEEE Std 26514-2010, *IEEE Standard for Adoption of ISO/IEC 26514:2008, Systems and Software Engineering — Requirements for Designers and Developers of User Documentation*)

1) ISO 10241 has been cancelled and replaced by ISO 10241 (all parts), *Terminological entries in standards*.

Abstract: ISO/IEC/IEEE 26515:2012 specifies the way in which user documentation can be developed in agile development projects. It is intended for use in all organizations that are using agile development, or are considering implementing their projects using these techniques. It applies to people or organizations producing suites of documentation, to those undertaking a single documentation project, and to documentation produced internally, as well as to documentation contracted to outside service organizations. ISO/IEC/IEEE 26515:2012 addresses the relationship between the user documentation process and the life cycle documentation process in agile development. It describes how the information developer or project manager may plan and manage the user documentation development in an agile environment. It is intended neither to encourage nor to discourage the use of any particular agile development tools or methods.

Keywords: agile development, incremental development, software user documentation

ICS 35.080
ISBN 978-0-7381-7151-7 STD97183 (PDF); 978-0-7381-7176-0 STDPD97183 (Print)

Price based on 27 pages