INTERNATIONAL STANDARD

ISO/IEC 24773-4

First edition
2023-07

Software and systems engineering — Certification of software and systems engineering professionals —

Part 4: Software engineering

Ingénierie du logiciel et des systèmes — Certification des professionnels de l'ingénierie du logiciel et des systèmes —

Partie 4: Ingénierie du logiciel

Reference number
ISO/IEC 24773-4:2023(E)

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

Published in Switzerland

Contents

Page

Foreword

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

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

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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 JTC 1, *Information Technology*, Subcommittee SC 7, *Software and systems engineering*.

A list of all parts in the ISO/IEC 24773 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 ISO/IEC 24773 series addresses the certification of professionals in software and systems engineering. ISO/IEC 24773-1 contains general requirements for such certification schemes. This document contains requirements specific to certification schemes for software engineering professionals.

The concepts, and requirements for certification schemes contained in ISO/IEC 24773-1 and ISO/IEC 17024 apply to this document.

Software and systems engineering — Certification of software and systems engineering professionals —

Part 4: Software engineering

1 Scope

This document elaborates requirements and recommendations for certifications schemes based on ISO/IEC 24773-1, which are specific to the domain of software engineering.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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 12207, *Systems and software engineering — Software life cycle processes*

ISO/IEC 17024, *Conformity assessment — General requirements for bodies operating certification of persons*

ISO/IEC TS 17027, *Conformity assessment — Vocabulary related to competence of persons used for certification of persons*

ISO/IEC/TR 19759, *Software Engineering — Guide to the software engineering body of knowledge (SWEBOK)*

ISO/IEC 24773-1:2019, *Software and systems engineering — Certification of software and systems engineering professionals — Part 1: General requirements*

Software Engineering Competency Model, version 1.0, IEEE Computer Society 2014

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 17024, ISO/IEC TS 17027, ISO/IEC 24773-1 and the following apply.

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/

3.1
knowledge area
KA
identified group of knowledge

3.2
software engineering
systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software

[SOURCE: ISO/IEC/IEEE 24765:2017, 3.3810, definition 1, modified — The abbreviated terms "SE" and "SWE" have been removed.]

4 Conformity

This document contains requirements for conformity certification schemes which are specific to software engineering. General requirements for conformant schemes defined in ISO/IEC 24773-1 also apply.

A certification scheme (the Scheme) for professionals in software engineering may claim conformance to this document if it satisfies all the requirements contained in this document, ISO/IEC 24773-1 and ISO/IEC 17024.

5 Requirements for certification of software engineering professionals

5.1 General

This clause contains requirements and recommendations specific to the certification of software engineering professionals. The following requirements, recommendations and associated criteria may also be used to compare certification and qualification schemes of software engineering professionals.

5.2 Fundamental components of a conformant scheme

5.2.1 This clause contains additional requirements for fundamental components defined in ISO/IEC 24773-1:2019, 6.3.

5.2.2 The Scheme shall contain a description of software engineering professionals to be certified or targeted by that scheme.

5.2.3 A list of skills that the professional to be certified in software engineering is expected to exhibit shall be mapped by the Scheme to the knowledge areas of the reference body of knowledge (BOK), and to the defined competencies.

NOTE Examples of skills and competencies are shown in Annex B and C respectively. An exemplar mapping is shown in Annex D.

5.2.4 Each task or activity associated with the Scheme shall be “mapped” to at least one of the following:

— KA of the Guide to the SWEBOK, ISO/IEC TR 19759;

— Process, activity or task of ISO/IEC/IEEE 12207;

— Technical skill area of IEEE Software Engineering Competency Model (SWECOM) V1.

The Scheme may define tasks at a different abstraction level of the tasks defined in ISO/IEC/IEEE 12207.

Tasks and their descriptions are associated with any certification scheme. ISO/IEC 17024 contains the requirements that the scheme declares a scope and describe the targeted individuals who would be certified. Part of the required scope and target description includes a list of tasks which represent the activities performed by certificants within the professional scope of that certification. These tasks are

high-level activity descriptions and not a comprehensive set of details of technical tasks which may be included within structure of the knowledge, skills or competencies of the scope.

5.2.5 The Scheme should require a minimum level of formal engineering education, where "formal engineering education" is defined as an engineering or computing degree from an accredited program, and where "computing degree" is one of the following: computer science (CS), computer engineering (CE), software engineering (SE), information systems (IS) and information technology (IT).

NOTE ISO/IEC 24773-1 contains a requirement that all conformant schemes define and document a minimum level of education, but this additional requirement is more specific regarding that minimum.

While minimum formal education requirements allow for various computing degrees, the requirements related to knowledge and coverage of KAs by the BOK (5.3.3) used by the Scheme still apply.

5.2.6 If formal engineering education is required as part of the certification scheme, the requirements shall be clearly stated.

The engineering degree does not have to be in software engineering, but shall be followed by software engineering work/experience.

5.2.7 If on-the-job experience in software engineering is to be acceptable as part or total replacement for formal engineering education, the experience defined by the Scheme shall be objectively described and the verifiable criteria shall be made explicit. The verifiable on-the-job experience in software engineering should be at least as long in years as the formal engineering education, and twice as long (at least eight years) if no formal university engineering degree is required.

5.3 Knowledge

5.3.1 The reference body of knowledge is ISO/IEC 19759 (SWEBOK).

The reference body of knowledge defines the following knowledge areas:

— software requirements;

— software design;

— software construction;

— software testing;

— software maintenance;

— software configuration management;

— software engineering management;

— software engineering process;

— software engineering models and methods;

— software quality;

— software engineering professional practice;

— software engineering economics;

— computing foundations;

— mathematical foundations;

— engineering foundations.

NOTE An elaboration of software engineering knowledge areas is shown in Annex A.

5.3.2 The Scheme shall include requirements for knowledge of ISO/IEC/IEEE 12207.

5.3.3 If the Scheme contains another BOK (i.e. "alternative BOK") other than the reference BOK identified in 5.3.1, the knowledge areas of the "alternative BOK" identified in the Scheme shall be mapped to knowledge areas of the reference BOK. The Scheme shall demonstrate that the "alternative BOK" covers the scope of the reference BOK as defined by the knowledge areas in 5.3.1, with rationale supporting any reduction in depth of coverage.

5.4 Skill

5.4.1 The Scheme shall define software engineering related skills covered by the Scheme, where they are specifically assessed or evaluated.

Skills covered by the Scheme should be placed into groups or categories, where skills are related, and where there are a large number of skills in one group or category.

NOTE Examples of software engineering skills are shown in Annex B.

5.4.2 The skills defined by the Scheme should be mapped to one or more competencies (5.5) defined and covered by the Scheme.

5.4.3 The Scheme should define performance levels for each skill so that various levels of ability are defined and can be measured objectively.

5.5 Competence

5.5.1 The Scheme shall identify software engineering specific competencies required for the software engineering professional to be certified. The competencies defined/required should be consistent with the activities of the professional defined as the target of the scheme (See ISO/IEC 24773-1:2019, 5.5, footnotes 1 and 2, and ISO/IEC 17024).

NOTE Examples of software engineering competencies are shown in Annex C.

5.5.2 Each competency associated with the Scheme shall be defined in terms of outcomes, deliverables, results or activities associated with that competency.

Competencies may be grouped into competency areas or groups. The Scheme may define its own competency groups and competencies within them.

5.5.3 The reference life cycle process is ISO/IEC/IEEE 12207.

NOTE This is an elaboration of the recommendation defined in ISO/IEC 24773-1:2019, 6.5.2.

5.5.4 Proficiency levels should be defined at least for the levels required to accomplish the tasks or activities undertaken by the targeted professional.

Annex A
(Informative)

Elaboration of software engineering knowledge areas

Table A.1 lists and provides some elaboration regarding the knowledge areas contained in the reference body of knowledge as defined in 5.3.

The source document (ISO/IEC TR 19759) contains detailed information regarding sub-topics and supporting literature for each knowledge area.

Table A.1 — Elaboration of software engineering knowledge areas

Software engineering knowledge areas	Sub-areas and sub-topics
Software requirements	Software requirements fundamentals, requirements process, requirements elicitation, requirements analysis, requirements specification, requirements validation, requirements management, requirements tools.
Software design	General design concepts, software design process, software design principles, key issues in software design (concurrency, control and handling of events, data persistence, distribution of components, error and exception handling and fault tolerance, user interaction and presentation, security), software structure and architecture, user interface design, software design quality analysis and evaluation, quality attributes, software design description and notations, software design strategies and methods, software design tools.
Software construction	Construction principles and objectives, reuse, complexity, construction standards, managing construction (planning and measurement), Integration, construction technologies, software construction tools.
Software testing	Testing-related concepts and terminology, test techniques, test-related measures, test process, software testing tools.
Software maintenance	Software maintenance fundamentals definitions and terminology, software evolution, maintenance costs, maintenance process, techniques for maintenance, software maintenance tools.
Software configuration management (SCM)	Management of the SCM process, software configuration identification, software configuration control, software configuration status accounting, software configuration auditing, software release management and delivery, software building, software configuration management tools.
Software engineering management	Project initiation and scope definition, software project planning, risk management, software project enactment, software acquisition and supplier contract management, implementation of measurement process, project review and evaluation, project closure, software engineering measurement, software engineering management tools.
Software engineering process	Software lifecycles, software process definition, software process assessment and improvement, software process and product measurement, software process measurement techniques, software engineering process tools.
Software engineering methods	Models and modelling principles, analysis of models, software engineering methods, formal methods, agile methods.
Software quality	Software quality fundamentals, costs of quality, models and quality characteristics, software safety, software quality management processes, software quality assurance, verification & validation, reviews and audits, software quality requirements, defect characterization, software quality management techniques, software quality measurement, software quality tools.

Table A.1 *(continued)*

Software engineering knowledge areas	Sub-areas and sub-topics
Software engineering professional practice	Professionalism concepts, codes of ethics and professional conduct, nature and role of professional societies, nature and role of software engineering standards, legal issues, team/group dynamics and psychology, interacting with stakeholders, communication skills.
Software engineering economics	Economics concepts and principles, software lifecycle economics, risks, estimates and uncertainty, economic analysis methods.
Computing foundations	Problem solving techniques, abstraction, programming fundamentals, programming language basics, debugging tools and techniques, data structure and representation, algorithms and complexity, concepts of a system, systems engineering, computer organization, compiler basics, operating systems basics, database basics and data management, network communication basics, parallel and distributed computing, parallel and distributed computing models, human factors, secure software development and maintenance, software security guidelines.
Mathematical foundations	Set, relations, functions, basic logic, proof techniques, graphs and trees, discrete probability, finite state machines, grammars, numerical precision, accuracy and errors, algebraic structures.
Engineering foundations	Empirical methods, statistical analysis, measurement, reliability, validity, engineering design, modelling simulation and prototyping, standards, root cause analysis.

Annex B
(Informative)

Examples of software engineering skills

Examples of software engineering skills are presented in this annex. Table B.1 provides examples of software engineering skills, activities, work products and outputs. Some external frameworks may be useful for certification body when defining software engineering skills in their scheme.

e-CF[3]: The skills defined in e-CF provide examples of software engineering skills.

SFIA[4]: The skills defined in SFIA provide examples of software engineering skills.

iCD[5]: The skill items and performance levels defined in iCD skill dictionary provide examples of skills and performance levels.

Specific performance level definitions for the skills are not addressed in Table B.1. See ISO/IEC 24773-1 and ISO/IEC 24773-2 for definition and guidance regarding skills and performance levels.

Table B.1 — Examples of software engineering skills

Skill area	Specific skills, outputs, activities
Software requirements	Requirement elicitation, definition. Requirement validation, definition of use cases, scenarios, stakeholder identification, requirements analysis. Identification of quality attribute / characteristic or non- functional requirement. Constraint identification. Use of tools related to requirements modelling, analysis models, domain models. Use of tools related to requirements management.
Software design	Module level design, class and component design, module interfaces and dependencies, algorithm selection, data structure definition. Design review. Use of tools related to software module and component design.
Software architecture	Problem analysis and architectural / solution objectives. Analysis of system level context, design alternatives, data flows, system boundaries. Analysis of platforms components and technologies, prototype and proof of concept development. Specification of system boundaries and interfaces. Specification of technology platforms and conventions to be used as a basis for module level design and implementation. Use of architectural analysis tools, understanding and application of various technologies and frameworks at the system level.
Software systems engineering (merge with software architecture skill area)	System level attributes definition, non-functional requirements specification. Analysis, specification, design and verification of interfaces between (sub) systems. Distributed systems analysis. Definition of system performance measures and metrics. Technical risk analysis in integration. Special analysis of security aspects (requirements, design, implementation and verification) across integrated systems.
Software construction and implementation	Code and unit testing, code review, refactoring. Tools for code development, unit testing, debugging, and source code analysis.
Software testing	Test case definition, test scenario definition, integration test definition, test implementation and verification, regression test, use of test tools and frameworks.
Software maintenance	Defect detection, reporting, analysis and classification. Impact assessment. Proposed design change or defect correction. Defect fix implementation, including unit testing and documentation. Defect submission to change control. Integration and regression testing of change or change set.

Table B.1 *(continued)*

Skill area	Specific skills, outputs, activities
Software configuration management	Define configuration items, define version and identification policies. Identify change control policy and trace to individual configuration items. Implement source code control and repository mechanisms. Implement change history report generation mechanism and differences summary mechanisms.
Software build, integrate, deploy – transition	Define configuration and automated build scripts and tools. Define and implement integration scripts, processes and incorporate integration level tests and scans. Define deployment format and automate deployment.
Operations, sustainment, maintenance, support	Configure and automate controls over operational environment settings, including automated scaling, failure recovery, and system health / telemetry. Define problem and error reporting. Perform error and failure analysis. Identify maintenance requirements. Identify operational controls and mechanism for deploying updates (patches or new release) as appropriate.
Software process and lifecycle	Define basic lifecycle process and implementation. Implement process and tailor with respect to project requirements and objectives. Define lifecycle stage transitions, integrated with risk management, reviews, change control. Define process performance measures. Define process quality measures.
Software quality assurance	Define quality assurance standards, activities, criteria. Define measures for quality (processes, work products, overall project or aggregate)
Software security	Identify security vulnerabilities in system context and environment. Identify security practices and standards for design and construction.
Software safety	Define Hazards, failure modes. Define safety functional requirements.
Software measurement	Define source code measures, define process measures, implement measurement data collection tools and utilities, produce analysis and summary reports.
Human computer interaction / UI / UX	Define user information needs, define interaction modes, define graphic layout, analyse accessibility, define help or warnings scheme, define usability test scenarios.
Software project management	Define project specific objectives, activities, resources. Establish tracking mechanisms for work, schedule. Establish project monitoring and control as per organizational project methodology.

Annex C
(Informative)

Examples of software engineering competencies

Examples of software engineering competencies are presented in this annex. Table C.1 provides examples of software engineering competencies. Some external frameworks may be useful for certification body when defining software engineering competencies in their scheme.

e-CF[3]: e-CF framework contains ICT competencies, some of which are related to software engineering competencies.

SFIA[4]: The SFIA framework contains examples of various ICT skills, and some of these are related to software engineering competencies.

iCD[5]: iCD framework contains examples of various ICT tasks and skills, some of these are related to software engineering competencies.

Specific proficiency level definitions for the competencies are not addressed in Table C.1. See ISO/IEC 24773-1 and ISO/IEC 24773-2 for definition and guidance regarding competency and proficiency levels.

Table C.1 — Examples of software engineering competencies

Competency area / competency	Description
Software requirements	At basic proficiency, able to properly elicit, identify, document and analyse individual software requirements. At higher levels of proficiency able to form models, review sets of requirements for consistency, and perform requirements control and management. Highest levels of proficiency address creation of requirements management process and mechanisms, and establishment of appropriate metrics and tools related to requirements engineering.
Software design	At basic proficiency, able to assume responsibility for module level design, including interfaces and other module level dependencies. Able to identify functional and non-functional requirements allocated to modules or components, including asynchronous and dynamic behaviour (such as exception handling). Able to specify data structures and algorithms at the module level. Able to compare module and component design alternatives, and to analyse and recommend the use of existing or third-party modules or components if appropriate. At a higher proficiency levels able to establish common design practices and design patterns, and to influence module level decomposition of a system including infrastructure and library modules.
Software architecture	Able to define context, architectural forces on system level, evaluate technologies and facilitate development and implementation from the architectural perspective. Able to model the architecture with various perspectives and emphases as needed by the context of the problem and the system. Able to understand and apply well known patterns and adapt them as needed. Able to follow research and new architectural conventions/models/technologies and apply or adapt them to maintain the relevance and currency of the architecture of the system. Responsible for longer term measures and evolution of the architectural components.

Table C.1 *(continued)*

Competency area / competency	Description
Software systems engineering	At basic proficiency, able to maintain integrity of the system design and implementation, consistency with requirements, architectural decisions. Ensures changes to scope and design are examined across the system. From a technical perspective implements change management and change control in order to enforce adequate reviews of changes and new developments within system scope. Establishes measures and metrics for important engineering activities, processes. At higher proficiency level ensures technical and quality risks (and by extension cost risks) are well managed by evaluating quality and performance of various technical processes, activities and verification steps. Able to make decisions at the highest level concerning trade-offs and product release.
Software construction and implementation	At basic proficiency, able to implement and unit test software at the unit or component level. Responsible for adhering to standards, participating in team goal setting and practices definition. Able to estimate and track her own construction activities, with respect to effort, quality and productivity.
Software testing	At basic proficiency, able to define specific tests from given requirements (functional and non-functional) and from derived requirements and constraints. Able to implement and automate tests at the unit level and functional tests. Able to create and control test artefacts including test data oracles, and to track test results and logs. Able to identify and analyse test failure and causes, and provide defect reporting. At higher proficiency able to define and implement system level tests, including performance, load, capacity, regression.
Software maintenance	At basic proficiency level, able to identify root cause of defects errors or failures, perform code reviews and corrections including refactoring, then verify the corrections/modifications with appropriate tests and verifications. At higher proficiency levels, is able to define the maintenance process from operations to problem reporting to code modification, re-submission and verification. Able to define maintenance aspects / requirements for build, release and deployment processes. Measures and tracks defects, errors and failures and is able to relate maintenance metrics with operations and performance metrics.
Software configuration management (CM)	At basic proficiency level, able to establish a configuration control system and implement change control mechanism. At higher levels is able to plan and enforce additional change control and tracking processes, and integrate CM process/systems with integration, build and deploy. Defines measures and metrics for the configuration management system and the CM process.
Software build, integrate, deploy - transition	At basic proficiency, establish and maintain reliable secure mechanism for builds integration (including any tests and verification) and deployment. At higher proficiency levels able to automate these processes and allow for configured variations in the build/release (e.g. multiple targets for deploy). At higher proficiency responsible for efficiency and turnaround performance of the CI CD pipeline, scalability of the pipeline, error handling/logging/reporting, and metrics.
Software engineering management	At high levels of proficiency, combines knowledge and abilities in order to establish an effective software engineering organization, including direction and management of methodology, software engineering metrics, ongoing management of product n and process quality, software engineering organizational capacity and development, software engineering resources and costs, risk management.
Systems operations, sustainment, maintenance, support	At basic proficiency level responsible for monitoring deployed systems and managing problems or incidents. At higher proficiency levels responsible for automated mechanisms for handling operational incidents or dynamic reconfiguration in production (e.g. scaling, failovers). Operational measures and metrics
Software process and lifecycle	At basic proficiency level able to define and instantiate lifecycle processes at the project level, monitor process performance. At higher proficiency level defines modifications for lifecycle processes including deliverables, product reviews, stage review transitions, risk management and process metrics.

Table C.1 *(continued)*

Competency area / competency	Description
Software quality assurance (SQA)	At basic proficiency level able to define appropriate SQA activities and plans appropriate for project. At higher proficiency able to measure effectiveness of all SQA processes and activities, and larger scale cost of quality metrics for the project / system.
Software security	At basic proficiency, able to define and apply secure software design and verification principles to component or module level designs. At higher proficiency level able to provide additional security requirements influencing architecture, design, construction, implementation, deployment and operations of the system. Able to define appropriate industry security standards and criteria. Able to define metrics and measures and influence other engineering and operations processes.
Software safety	At basic proficiency able to identify safety specific requirements and influence design of safety related functions. Able to define verification methods for all software safety related requirements and participate in software design and code reviews from a safety perspective. At higher proficiency levels, able to assist with systems level safety analysis, hazard analysis and failure mode analysis to ensure completeness of software safety plan and assurance case. Abel to assess safety risks and establish measures to track performance and validation of safety related functions.
Software measurement	At basic proficiency, able to define and generate basic software measures and metrics on code base. Validate and verify metrics. At higher levels adapt and provide additional measures related to defects, defect removal efficiency, defect density, and other metrics related to quality. Define lifecycle metrics for work products, lifecycle stage transitions, reviews, effort.
Human computer interaction / UI / UX	At basic proficiency identify basic interactions, design and validate design. At higher levels ensure accessibility and usability, including adherence to existing codes and standards. Design and implement extensive usability testing and relate the results to design improvements. Define overall measures and track.
Software project management	At basic proficiency level is able to define standard project activities, tasks and schedule, and track, monitor status. At higher levels is able to provide detail in project plans, address risk management, address scope changes and change management.