ISO/IEC/IEEE 12207:2026 Systems and software engineering, Software life cycle processes (2nd edition, April 2026)

Process framework for the software life cycle

63 requirements. There is no accredited certificate for this standard. As of 07/2026.

ISO/IEC/IEEE 12207 defines a framework of 30 processes covering the complete software life cycle, from agreement through development and operation to disposal. There is no accredited certification scheme: the standard explicitly treats conformance as a self declaration in three variants (full conformance to outcomes, full conformance to tasks, or tailored conformance with a documented tailoring record), so the result of an assessment is a conformance statement and not a certificate. Organizations that need an external judgement commission a process assessment based on the ISO/IEC 33000 series, which delivers a capability profile in the form of a report.

Company	Date	Prepared by

6.1 Agreement Processes 6

6.1.1 The Acquisition process is in place: the need for externally sourced software or services is identified, suppliers are selected, and an agreement is concluded and monitored.

Document

What proves it: Evidence includes tender or requirement documents, comparisons of offers, signed contracts or purchase orders, and acceptance records. In small teams a short acquisition ticket per purchase in the issue tracker, covering the need, the comparison of alternatives, and the approval, is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.1.2 The outcomes of acquisition are achieved: requirements for the acquired item are defined, the agreement is concluded, and fulfilment is confirmed.

What proves it: Acceptance criteria in the contract or in the ticket, plus the acceptance test or acceptance record. For open source dependencies, the lockfile with a documented selection criterion and the license scan serve as acceptance evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.1.3 The acquisition activities are carried out: identification of the need, supplier selection, conclusion of the agreement, progress monitoring, and acceptance are performed in a traceable way.

What proves it: Records from the issue tracker and the contract folder, supplemented by supplier status reports. A supplier list with the date of the last evaluation is enough for small organizations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.2 The Supply process is in place: the organization reviews requests, submits an offer, concludes an agreement, delivers the product or service, and formally closes the agreement.

Document

What proves it: Offers, order confirmations, delivery and release notes, handover records, and invoices. Release notes per version are the most practical delivery evidence for software teams.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.2.2 The outcomes of supply are achieved: the scope of delivery is agreed, the product is handed over, and responsibility for it is settled.

What proves it: Signed release artifacts with version number, changelog, and a handover message to the customer. In addition, support and warranty arrangements in the contract terms.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.1.2.3 **The supply activities are carried out: feasibility is assessed, the delivery is executed, and the agreement is closed once it has been fulfilled.**

What proves it: A record of the feasibility or effort estimate in the offer, the deployment log of the CI pipeline as delivery evidence, and a short closure record per order.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2 Organizational Project-Enabling Processes 9

6.2.1 **Life Cycle Model Management is in place: the organization defines, maintains, and makes available the life cycle models, processes, and procedures that projects use.** Document

What proves it: A lean process description in the repository, for example CONTRIBUTING.md plus a one page development workflow with branching model, definition of done, and release cadence. Changes to it are datable through the Git history.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.1.2 **The outcomes are achieved: the life cycle models in use are defined, projects are aware of them, and they are reviewed regularly for effectiveness.**

What proves it: Retrospective notes or an annual review entry in the repository that gives the reason for adapting the workflow. Pull requests against the process documents are the evidence of effectiveness.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.2 **Infrastructure Management is in place: the infrastructure needed for development, test, and operation is defined, provided, and maintained.**

What proves it: Infrastructure as code in the repository (Terraform, Docker Compose, CI runner configuration), plus a list of the services in use with their owners. For small teams a single page with environments, access, and backup rules is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.2.3 **The infrastructure activities are carried out: needs are identified, resources are acquired, operated, and retired when no longer required.**

What proves it: Change history of the infrastructure configuration in Git, monitoring dashboards, and tickets for environment changes. Decommissioned environments are documented in the ticket with a date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.3 **Portfolio Management is in place: projects are initiated, prioritized, resourced, evaluated on an ongoing basis, and terminated when they no longer deliver value.**

What proves it: A product roadmap or portfolio board with priorities and decision dates, plus records of initiatives that were stopped. A roadmap document with a quarterly review is enough for small organizations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.4 **Human Resource Management is in place: the required competencies are determined, staff are qualified, and knowledge is built up and retained.**

What proves it: A competence matrix per role, training and certificate records, an onboarding checklist. In small teams a skills table showing gaps and planned training, together with evidence of completed training, is sufficient.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.2.5 **Quality Management is in place: quality objectives, the quality policy, and responsibilities are defined, and achievement of the objectives is evaluated.** Document

What proves it: A short quality policy in the repository, defined quality objectives (test coverage, defect density, response time for incidents), and the corresponding metrics from CI and the issue tracker.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.2.5.2 **The outcomes of quality management are achieved: quality objectives are measurable, deviations are detected, and corrective actions are tracked.**

What proves it: Trend reports from the CI pipeline (coverage, failed builds), defect statistics from the issue tracker, and tickets for the resulting actions with a completion date.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.2.6 **Knowledge Management is in place: knowledge, skills, and reusable assets are captured, shared, and kept available for later projects.**

What proves it: An internal knowledge base or wiki, architecture decision records in the repository, reusable libraries with documentation. Postmortem notes after incidents are strong evidence.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.3 Technical Management Processes 16

6.3.1 **Project Planning is in place: scope, deliverables, schedule, effort, roles, and interfaces are planned, and the plan is kept up to date.** Document

What proves it: A project plan or backlog with milestones, estimates, and owners. A maintained sprint or milestone board in the issue tracker meets the requirement as long as scope and dates are visible.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.3.1.2 **The planning outcomes are achieved: tasks and resources are allocated, roles are named, and the plans are agreed with those involved.**

What proves it: Assigned issues with an owner and a target version, plus the milestone overview. Capacity assumptions are noted briefly in the planning ticket.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.3.1.3 **The planning activities are carried out: the project scope is defined, the plans are produced, and they are updated whenever changes occur.**

What proves it: Version history of the plan or roadmap document in Git, and change comments on milestones. This makes every update datable.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.3.2 **Project Assessment and Control is in place: progress against plan is assessed, deviations are detected, and control actions are taken.** Document

What proves it: Status reports, sprint reviews, burndown or throughput metrics from the issue tracker, and records of the decisions taken when deviations occurred. A short weekly status note is sufficient.

◯ Open ◯ In progress ◯ Met ◯ Not applicable

Evidence / justification

6.3.3 **Decision Management is in place: decisions are prepared in a structured way, alternatives are evaluated, and the selection is justified and recorded.** Document

What proves it: Architecture decision records in the repository, covering context, options, decision, and consequences. For smaller decisions a reasoned comment in the pull request is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.3.2 **The outcomes of decision management are achieved: the need for a decision is recognized, alternatives are evaluated, and the decision taken is traceable together with its rationale.**

What proves it: The ADR folder with sequential numbering and status (proposed, accepted, superseded). The Git history evidences the date and the author.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.4 **Risk Management is in place: risks are identified, analyzed, treated, and monitored on an ongoing basis.** Document

What proves it: A risk register with likelihood, impact, treatment, and owner. Technical risks can be tracked as issues with a risk label, together with the results of dependency and vulnerability scans.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.4.2 **The outcomes of risk management are achieved: risks are prioritized, thresholds for action are defined, and the effectiveness of the treatment is tracked.**

What proves it: Update history of the risk register, closed risk tickets with a rationale, and reports from automated dependency scanning showing the remediation date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.5 **Configuration Management is in place: configuration items are identified, baselines are defined, changes are controlled, and the status of every item can be determined at any time.** Document

What proves it: Git as version control with tags for releases, protected main branches, lockfiles for dependencies, and a software bill of materials (SBOM) per release. The tag history is the baseline record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.5.2 **The outcomes of configuration management are achieved: baselines are established, changes are traceable, and releases are reproducible.**

What proves it: A reproducible build from a Git tag through the CI pipeline, plus build logs and an artifact registry with checksums.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.5.3 **The configuration management activities are carried out: change requests are evaluated, approved or rejected, and their incorporation is verified.**

What proves it: Pull request history with mandatory review and a linked issue, and merge rules in the repository (branch protection) as evidence of approval control.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.6 **Information Management is in place: the information to be created and retained is determined, and it is maintained, protected, and made available to those entitled to it.** Document

What proves it: A records concept covering storage location, owner, retention period, and access rights. Repository permissions and backup logs evidence protection and availability.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.7 **The Measurement process is in place: information needs are determined, suitable measures are defined, collected, analyzed, and used for decisions.**

What proves it: Metrics from CI and the issue tracker, such as lead time, defect rate, test coverage, deployment frequency. What matters is the evidenced path from the measure to the decision, for example in the retrospective record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.7.2 **The outcomes of measurement are achieved: the collected data are valid, the analysis is documented, and the results are communicated.**

What proves it: Automatically generated pipeline reports with a timestamp, and a regular metrics summary posted in the team channel or wiki.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.8 **Quality Assurance is in place: products and processes are checked independently against the defined criteria, and deviations are recorded and resolved.** Document

What proves it: Mandatory four eyes review in the pull request, linter and test runs as a merge gate, and recording of the deviations found as issues. A small team achieves independence through rotating reviewers rather than a dedicated QA department.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.3.8.2 **The outcomes of quality assurance are achieved: process conformance and product conformance are evaluated, and corrective actions are tracked to closure.**

What proves it: Review comments with a resolution status, closed bug tickets referencing the fix commit, and a green pipeline run for the release tag.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4 Technical Processes 26

6.4.1 **Business or Mission Analysis is performed: the problem to be solved, the solution space, and the constraints of the undertaking are determined.**

What proves it: A product vision, problem statement, or business case in the wiki, supplemented by market analysis or user interviews. An epic with a problem description and objective in the issue tracker is enough for small undertakings.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.2 **Stakeholder Needs and Requirements Definition is performed: stakeholders are identified, their needs are elicited, and they are transformed into verifiable stakeholder requirements.** Document

What proves it: A stakeholder list, interview notes, user stories with acceptance criteria. The acceptance criteria in the ticket are the verifiable form of the requirement.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.2.2 **The outcomes are achieved: the stakeholder requirements are agreed, prioritized, and traceable back to the underlying need.**

What proves it: A prioritized backlog that links story to epic, plus the customer approval comment in the ticket.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.3 **System Requirements Definition is performed: the technical requirements for the software system are derived from the stakeholder requirements, including quality characteristics and constraints.** Document

What proves it: A requirements document or structured issues covering functional and non functional requirements (performance, security, accessibility). Linking each requirement to a test case makes verifiability explicit.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.3.2 **The outcomes of requirements definition are achieved: the requirements are unambiguous, free of contradictions, verifiable, and traceable back to the stakeholder requirements.**

What proves it: A traceability matrix, or alternatively issue links from story to test and to commit. Automated acceptance tests are at the same time the evidence of verifiability.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.4 **System Architecture Definition is performed: an architecture with views, building blocks, and interfaces is defined and evaluated against the requirements.** Document

What proves it: An architecture overview in the repository (for example arc42 or C4 diagrams as an image plus text), together with architecture decision records holding the evaluated alternatives.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.4.2 **The outcomes of system architecture definition are achieved: the architecture is evaluated against the requirements, alternatives are weighed, and the selection is justified.**

What proves it: An ADR weighing options and consequences, plus the results of an architecture review in the pull request or in a meeting record.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.5 **Design Definition is performed: designs exist for the architectural building blocks that are sufficient for implementation, including the data model and interface contracts.**

What proves it: Interface specifications (OpenAPI, GraphQL schema, type definitions), database migration scripts, and design notes in the pull request. The code itself, with its types and module boundaries, is acceptable design evidence provided the decisions are documented.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.5.2 **The outcomes of design definition are achieved: the design characteristics are traceable to the architecture and feasibility is confirmed.**

What proves it: References from the pull request to the ADR and to the requirement, and prototypes or spikes with a result note as evidence of feasibility.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.6 **System Analysis is performed: technical questions are investigated with analytical methods in order to substantiate decisions on architecture, design, or risks.**

What proves it: Load measurements and benchmark results, cost or trade off analyses, threat model notes. Results belong with the related ADR as a short analysis note.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.7 **Implementation is performed: the software elements are realized in line with the design and the coding rules, and the results are placed under configuration control.**

What proves it: Git history with commits that reference issues, linter and formatting rules as a merge gate, and successful builds in the CI pipeline.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.7.3 **The implementation activities are carried out: coding standards are defined, the realization is reviewed, and the elements are made available for integration.**

What proves it: Linter and formatter configuration files in the repository, passed pull request reviews, and published build artifacts in the registry.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.8 **Integration is performed: software elements are progressively assembled into a working system and the interfaces are checked.**

What proves it: Continuous integration runs on every merge, integration tests and contract tests against the interfaces, plus the build logs with timestamps.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.8.2 **The outcomes of integration are achieved: the integration strategy is defined, the interfaces are consistent, and integration defects are recorded and resolved.**

What proves it: The pipeline definition in the repository as the integration strategy, failing runs followed by green runs as evidence of defect resolution, and bug tickets with the fix commit.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.9 **Verification is performed: for every specified requirement there is verifiable, traceable evidence that the software as built actually satisfies it.** Document

What proves it: Automated unit, integration, and regression tests with a test report from the CI pipeline, static analysis, and code reviews. The test report per release tag is the central piece of evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.9.2 **The outcomes of verification are achieved: verification criteria are defined, the checks are executed, and deviations are recorded.**

What proves it: Defined acceptance thresholds (for example a minimum test coverage, no critical findings), archived test reports, and an issue for every deviation.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.9.3 **The verification activities are carried out: the scope and the methods of verification are planned, executed, and the results are analyzed.**

What proves it: A test plan or test strategy as a short page in the repository, the pipeline configuration with its verification stages, and the analysis in the release approval.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.10 **The Transition process is performed: the system is installed in its target environment, handed over ready for use, and its operational capability is confirmed.** Document

What proves it: A deployment runbook, deployment logs from the pipeline, smoke tests after the rollout, and a documented rollback plan. Release notes evidence the handover to users and to operations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.11 **Validation is performed: under the intended conditions of use it is demonstrated that the software actually meets the original needs of the stakeholders.** Document

What proves it: Acceptance tests with the customer, a beta or pilot phase with user feedback, and acceptance criteria in the ticket marked as met. Usage data from operation gives the validation additional support.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.11.2 **The outcomes of validation are achieved: validation criteria are defined, validation takes place in a representative environment, and deviations are recorded.**

What proves it: An acceptance record with date and participants, results from the staging or pilot environment, and feedback tickets with their resolution status.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.12 **The Operation process is in place: the system is operated as intended, it is monitored, and users are supported.**

What proves it: Monitoring and alerting with defined thresholds, incident tickets with response times, and a support mailbox or ticket channel. An operations manual of a few pages is enough for small organizations.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.12.2 **The outcomes of operation are achieved: the operation strategy and its measures are defined, incidents are recorded, and performance is evaluated against expectations.**

What proves it: Availability and error metrics from monitoring, incident logs, and postmortems with the actions derived from them.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.13 **The Maintenance process is in place: defect fixes, adaptations, and improvements are recorded, evaluated, implemented, and released without compromising the integrity of the system.** Document

What proves it: Bug and change tickets with priority and root cause, hotfix branches, regression tests before release, and release notes per maintenance version. Regular dependency updates are evidenced through the lockfile history.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.13.2 **The outcomes of maintenance are achieved: changes are analyzed, their impact is assessed, and the effectiveness of the fix is confirmed.**

What proves it: An impact analysis in the ticket, a regression test run after the fix, and confirmation by the reporter. Security updates are evidenced through vulnerability scan reports showing the remediation date.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.14 **The Disposal process is in place: the retirement of systems or parts of systems is planned, data are migrated or deleted, and affected parties are informed.** Document

What proves it: A retirement plan with dates, evidence of data migration or deletion (including backups), a shutdown record for the infrastructure, and the announcement to users or customers.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

6.4.14.2 **The outcomes of disposal are achieved: retention obligations are met, environments are shut down, and responsibility for the system is ended.**

What proves it: An archive of the data subject to retention obligations with the retention period stated, a ticket for shutting down resources with a date, and a deletion confirmation from the hosting provider.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

A Conformance and Tailoring 6

4.1 **The scope of application is defined (clause 4.1, General): it is determined whether the standard is applied to a project, an organization, or an agreement, and which processes are covered.** Document

What proves it: A short scope statement in the process documentation naming the project, the product, and the process groups applied. One page in the repository is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.2 **The chosen conformance variant is declared: either full conformance to all outcomes of the claimed processes, or full conformance to all tasks of the claimed processes.** Document

What proves it: A conformance statement as a separate document that names the variant and points to the evidence for each process. This produces a self declaration, not a certificate.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

4.3 **Where tailored conformance is claimed, the tailoring is documented: it is stated which processes, tasks, or outcomes have been omitted, modified, or added.** Document

What proves it: A tailoring table with the columns process, decision (adopted, adapted, omitted), rationale, and risk. For small teams this document is the decisive piece of evidence.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

5 **The key concepts are applied to the organization: the relationship between the life cycle model, the project structure, and the processes in use is presented in an understandable way.**

What proves it: An overview page mapping the chosen approach (for example iterative, continuous delivery) onto the 30 processes. A diagram plus a table is enough.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

A.1 **Tailoring is handled as a deliberate decision: tailoring is permitted, but it is only applied where the constraints of the undertaking justify it, and it is always justified.** Document

What proves it: A rationale column in the tailoring table referring to project size, risk class, contractual situation, or legal requirements. Blanket removals without a rationale are not valid tailoring.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification

A.2 **The tailoring process is carried out: influencing factors are identified, stakeholders are involved, and tailoring decisions are taken and documented.** Document

What proves it: A record or ticket for the tailoring decision naming the roles involved, plus the versioned tailoring table in Git. Effects on risks are noted in the risk register.

○ Open ○ In progress ○ Met ○ Not applicable

Evidence / justification