ISO/IEC/IEEE 12207:2026 Systems and software engineering, Software life cycle processes (2e édition, avril 2026)

Cadre de processus pour le cycle de vie du logiciel

63 exigences. Il n'existe pas de certificat accrédité pour cette norme. Version de 07/2026.

ISO/IEC/IEEE 12207 définit un cadre de 30 processus couvrant l'ensemble du cycle de vie du logiciel, de l'accord jusqu'au développement, à l'exploitation et au retrait. Il n'existe pas de schéma de certification accrédité : la norme traite explicitement la conformité comme une autodéclaration selon trois variantes (conformité totale aux résultats, conformité totale aux tâches, ou conformité adaptée avec un dossier d'adaptation documenté), de sorte que le résultat d'une évaluation est une déclaration de conformité et non un certificat. Les organisations qui ont besoin d'un jugement externe font réaliser une évaluation de processus fondée sur la série ISO/IEC 33000, qui fournit un profil de capacité sous forme de rapport.

Entreprise	Date	Établi par

6.1 Processus d'accord (Agreement Processes) 6

6.1.1 Le processus d'Acquisition est en place : le besoin de logiciels ou de services obtenus en externe est identifié, les fournisseurs sont sélectionnés, et un accord est conclu et suivi.

Document

Comment le prouver: Les preuves comprennent les documents d'appel d'offres ou de besoins, les comparatifs d'offres, les contrats ou bons de commande signés, et les procès-verbaux de réception. Dans les petites équipes, un ticket d'acquisition succinct par achat dans le système de suivi des tickets, couvrant le besoin, la comparaison des alternatives et l'approbation, est suffisant.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.1.1.2 Les résultats de l'acquisition sont atteints : les exigences relatives à l'élément acquis sont définies, l'accord est conclu, et l'exécution est confirmée.

Comment le prouver: Critères d'acceptation dans le contrat ou dans le ticket, ainsi que le test de réception ou le procès-verbal de réception. Pour les dépendances open source, le fichier de verrouillage (lockfile) accompagné d'un critère de sélection documenté et l'analyse des licences servent de preuve d'acceptation.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.1.1.3 Les activités d'acquisition sont réalisées : l'identification du besoin, la sélection des fournisseurs, la conclusion de l'accord, le suivi de l'avancement et la réception sont effectués de manière traçable.

Comment le prouver: Enregistrements du système de suivi des tickets et du dossier des contrats, complétés par des rapports d'état des fournisseurs. Une liste des fournisseurs avec la date de la dernière évaluation suffit pour les petites organisations.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.1.2 Le processus de Fourniture est en place : l'organisation examine les demandes, soumet une offre, conclut un accord, livre le produit ou le service, et clôture formellement l'accord.

Document

Comment le prouver: Offres, confirmations de commande, notes de livraison et de version, procès-verbaux de remise, et factures. Les notes de version par version constituent la preuve de livraison la plus pratique pour les équipes logicielles.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.1.2.2 Les résultats de la fourniture sont atteints : le périmètre de la livraison est convenu, le produit est remis, et la responsabilité de celui-ci est établie.

Comment le prouver: Artefacts de version signés avec numéro de version, journal des modifications, et un message de remise au client. En complément, les dispositions de support et de garantie dans les conditions du contrat.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.1.2.3 **Les activités de fourniture sont réalisées : la faisabilité est évaluée, la livraison est exécutée, et l'accord est clôturé une fois qu'il a été exécuté.**

Comment le prouver: Un enregistrement de l'estimation de faisabilité ou de charge dans l'offre, le journal de déploiement du pipeline d'intégration continue comme preuve de livraison, et un bref enregistrement de clôture par commande.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2 Processus organisationnels de soutien aux projets (Organizational Project-Enabling Processes) 9

6.2.1 **La Gestion du modèle de cycle de vie est en place : l'organisation définit, maintient et met à disposition les modèles de cycle de vie, les processus et les procédures utilisés par les projets.** Document

Comment le prouver: Une description de processus allégée dans le dépôt, par exemple un CONTRIBUTING.md accompagné d'une page décrivant le flux de développement avec le modèle de branches, la définition de terminé, et la cadence des versions. Ses modifications sont datables grâce à l'historique Git.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.1.2 **Les résultats sont atteints : les modèles de cycle de vie utilisés sont définis, les projets en ont connaissance, et leur efficacité est revue régulièrement.**

Comment le prouver: Notes de rétrospective ou entrée de revue annuelle dans le dépôt indiquant la raison de l'adaptation du flux de travail. Les demandes de tirage (pull requests) sur les documents de processus constituent la preuve d'efficacité.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.2 **La Gestion de l'infrastructure est en place : l'infrastructure nécessaire au développement, aux tests et à l'exploitation est définie, fournie et maintenue.**

Comment le prouver: Infrastructure en tant que code dans le dépôt (Terraform, Docker Compose, configuration des exécuteurs d'intégration continue), ainsi qu'une liste des services utilisés avec leurs responsables. Pour les petites équipes, une seule page présentant les environnements, les accès et les règles de sauvegarde suffit.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.2.3 **Les activités d'infrastructure sont réalisées : les besoins sont identifiés, les ressources sont acquises, exploitées, et retirées lorsqu'elles ne sont plus nécessaires.**

Comment le prouver: Historique des modifications de la configuration d'infrastructure dans Git, tableaux de bord de supervision, et tickets pour les changements d'environnement. Les environnements mis hors service sont documentés dans le ticket avec une date.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.3 **La Gestion de portefeuille est en place : les projets sont initiés, priorisés, dotés de ressources, évalués en continu, et clôturés lorsqu'ils ne créent plus de valeur.**

Comment le prouver: Une feuille de route produit ou un tableau de portefeuille avec les priorités et les dates de décision, ainsi que des enregistrements des initiatives arrêtées. Un document de feuille de route avec une revue trimestrielle suffit pour les petites organisations.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.4 **La Gestion des ressources humaines est en place : les compétences requises sont déterminées, le personnel est qualifié, et les connaissances sont constituées et conservées.**

Comment le prouver: Une matrice de compétences par rôle, des enregistrements de formation et de certificats, une liste de vérification d'intégration. Dans les petites équipes, un tableau des compétences montrant les écarts et les formations prévues, accompagné des preuves de formation suivie, est suffisant.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.5 **La Gestion de la qualité est en place : les objectifs qualité, la politique qualité et les responsabilités sont définis, et l'atteinte des objectifs est évaluée.** Document

Comment le prouver: Une politique qualité succincte dans le dépôt, des objectifs qualité définis (couverture de tests, densité de défauts, délai de réponse aux incidents), et les indicateurs correspondants issus de l'intégration continue et du système de suivi des tickets.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.5.2 **Les résultats de la gestion de la qualité sont atteints : les objectifs qualité sont mesurables, les écarts sont détectés, et les actions correctives sont suivies.**

Comment le prouver: Rapports de tendance du pipeline d'intégration continue (couverture, échecs de compilation), statistiques de défauts issues du système de suivi des tickets, et tickets pour les actions qui en résultent avec une date d'achèvement.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.2.6 **La Gestion des connaissances est en place : les connaissances, les compétences et les actifs réutilisables sont capturés, partagés, et maintenus disponibles pour les projets ultérieurs.**

Comment le prouver: Une base de connaissances interne ou un wiki, des fiches de décision d'architecture dans le dépôt, des bibliothèques réutilisables documentées. Les notes d'analyse post-incident (postmortem) constituent une preuve solide.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3 Processus techniques de gestion (Technical Management Processes) 16

6.3.1 **La Planification de projet est en place : le périmètre, les livrables, le calendrier, la charge, les rôles et les interfaces sont planifiés, et le plan est tenu à jour.** Document

Comment le prouver: Un plan de projet ou un backlog avec des jalons, des estimations et des responsables. Un tableau de sprint ou de jalons tenu à jour dans le système de suivi des tickets satisfait l'exigence dès lors que le périmètre et les dates sont visibles.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.1.2 **Les résultats de la planification sont atteints : les tâches et les ressources sont allouées, les rôles sont nommés, et les plans sont approuvés avec les personnes concernées.**

Comment le prouver: Tickets assignés avec un responsable et une version cible, ainsi qu'une vue d'ensemble des jalons. Les hypothèses de capacité sont notées brièvement dans le ticket de planification.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.1.3 **Les activités de planification sont réalisées : le périmètre du projet est défini, les plans sont produits, et ils sont mis à jour à chaque changement.**

Comment le prouver: Historique des versions du document de plan ou de feuille de route dans Git, et commentaires de modification sur les jalons. Cela rend chaque mise à jour datable.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.2 **L'Évaluation et la maîtrise du projet sont en place : l'avancement par rapport au plan est évalué, les écarts sont détectés, et des actions de maîtrise sont prises.** Document

Comment le prouver: Rapports d'état, revues de sprint, indicateurs d'avancement (burndown) ou de débit issus du système de suivi des tickets, et enregistrements des décisions prises en cas d'écart. Une brève note d'état hebdomadaire suffit.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.3 **La Gestion des décisions est en place : les décisions sont préparées de manière structurée, les alternatives sont évaluées, et le choix est justifié et consigné.** Document

Comment le prouver: Fiches de décision d'architecture dans le dépôt, couvrant le contexte, les options, la décision et les conséquences. Pour les décisions mineures, un commentaire argumenté dans la demande de tirage suffit.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.3.2 **Les résultats de la gestion des décisions sont atteints : le besoin d'une décision est reconnu, les alternatives sont évaluées, et la décision prise est traçable avec sa justification.**

Comment le prouver: Le dossier des fiches de décision d'architecture avec une numérotation séquentielle et un statut (proposée, acceptée, remplacée). L'historique Git atteste de la date et de l'auteur.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.4 **La Gestion des risques est en place : les risques sont identifiés, analysés, traités, et surveillés en continu.** Document

Comment le prouver: Un registre des risques avec la probabilité, l'impact, le traitement et le responsable. Les risques techniques peuvent être suivis comme des tickets portant une étiquette de risque, avec les résultats des analyses de dépendances et de vulnérabilités.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.4.2 **Les résultats de la gestion des risques sont atteints : les risques sont priorisés, des seuils de déclenchement d'action sont définis, et l'efficacité du traitement est suivie.**

Comment le prouver: Historique des mises à jour du registre des risques, tickets de risque clôturés avec une justification, et rapports d'analyse automatisée des dépendances indiquant la date de correction.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.5 **La Gestion de la configuration est en place : les éléments de configuration sont identifiés, des lignes de référence sont définies, les changements sont maîtrisés, et le statut de chaque élément peut être déterminé à tout moment.** Document

Comment le prouver: Git comme gestion de versions avec des étiquettes (tags) pour les versions publiées, des branches principales protégées, des fichiers de verrouillage (lockfiles) pour les dépendances, et une nomenclature logicielle (SBOM) par version. L'historique des étiquettes constitue l'enregistrement des lignes de référence.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.5.2 **Les résultats de la gestion de la configuration sont atteints : les lignes de référence sont établies, les changements sont traçables, et les versions publiées sont reproductibles.**

Comment le prouver: Une compilation reproductible à partir d'une étiquette Git via le pipeline d'intégration continue, ainsi que des journaux de compilation et un dépôt d'artefacts avec sommes de contrôle.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.5.3 **Les activités de gestion de la configuration sont réalisées : les demandes de changement sont évaluées, approuvées ou rejetées, et leur intégration est vérifiée.**

Comment le prouver: Historique des demandes de tirage avec revue obligatoire et un ticket lié, et règles de fusion dans le dépôt (protection des branches) comme preuve de la maîtrise des approbations.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.3.6 **La Gestion de l'information est en place : les informations à créer et à conserver sont déterminées, et elles sont maintenues, protégées, et mises à disposition des personnes habilitées.** Document

Comment le prouver: Un concept d'archivage couvrant le lieu de stockage, le responsable, la durée de conservation et les droits d'accès. Les permissions du dépôt et les journaux de sauvegarde attestent de la protection et de la disponibilité.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.3.7 **Le processus de Mesure est en place : les besoins d'information sont déterminés, des indicateurs adaptés sont définis, recueillis, analysés, et utilisés pour la prise de décision.**

Comment le prouver: Indicateurs issus de l'intégration continue et du système de suivi des tickets, tels que le délai de livraison, le taux de défauts, la couverture de tests, la fréquence de déploiement. Ce qui compte, c'est le chemin documenté entre la mesure et la décision, par exemple dans le compte-rendu de rétrospective.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.3.7.2 **Les résultats de la mesure sont atteints : les données recueillies sont valides, l'analyse est documentée, et les résultats sont communiqués.**

Comment le prouver: Rapports de pipeline générés automatiquement avec horodatage, et une synthèse régulière des indicateurs publiée dans le canal de l'équipe ou le wiki.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.3.8 **L'Assurance qualité est en place : les produits et les processus sont vérifiés de manière indépendante par rapport aux critères définis, et les écarts sont enregistrés et résolus.** Document

Comment le prouver: Revue croisée obligatoire (principe des quatre yeux) dans la demande de tirage, exécutions de l'analyseur de code (linter) et des tests comme condition de fusion, et enregistrement des écarts constatés sous forme de tickets. Une petite équipe obtient l'indépendance par une rotation des relecteurs plutôt que par un service qualité dédié.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.3.8.2 **Les résultats de l'assurance qualité sont atteints : la conformité du processus et la conformité du produit sont évaluées, et les actions correctives sont suivies jusqu'à leur clôture.**

Comment le prouver: Commentaires de revue avec un statut de résolution, tickets d'anomalie clôturés référençant le commit de correction, et une exécution de pipeline réussie pour l'étiquette de version.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.4 Processus techniques (Technical Processes) 26

6.4.1 **L'Analyse des activités ou de la mission est réalisée : le problème à résoudre, l'espace des solutions et les contraintes du projet sont déterminés.**

Comment le prouver: Une vision produit, un énoncé du problème, ou une analyse de rentabilité dans le wiki, complétés par une analyse de marché ou des entretiens utilisateurs. Une épopée (epic) avec une description du problème et un objectif dans le système de suivi des tickets suffit pour les petits projets.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.4.2 **La Définition des besoins et exigences des parties prenantes est réalisée : les parties prenantes sont identifiées, leurs besoins sont recueillis, et ils sont transformés en exigences vérifiables des parties prenantes.** Document

Comment le prouver: Une liste des parties prenantes, des notes d'entretien, des récits utilisateur (user stories) avec critères d'acceptation. Les critères d'acceptation dans le ticket constituent la forme vérifiable de l'exigence.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.2.2 **Les résultats sont atteints : les exigences des parties prenantes sont approuvées, priorisées, et traçables jusqu'au besoin sous-jacent.**

Comment le prouver: Un backlog priorisé reliant les récits aux épopées, ainsi que le commentaire d'approbation du client dans le ticket.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.3 **La Définition des exigences système est réalisée : les exigences techniques du système logiciel sont dérivées des exigences des parties prenantes, y compris les caractéristiques de qualité et les contraintes.** Document

Comment le prouver: Un document d'exigences ou des tickets structurés couvrant les exigences fonctionnelles et non fonctionnelles (performance, sécurité, accessibilité). Relier chaque exigence à un cas de test rend la vérifiabilité explicite.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.3.2 **Les résultats de la définition des exigences sont atteints : les exigences sont sans ambiguïté, exemptes de contradictions, vérifiables, et traçables jusqu'aux exigences des parties prenantes.**

Comment le prouver: Une matrice de traçabilité, ou à défaut des liens entre le récit, le test et le commit. Les tests de recette automatisés constituent en même temps la preuve de vérifiabilité.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.4 **La Définition de l'architecture système est réalisée : une architecture comportant des vues, des blocs de construction et des interfaces est définie et évaluée au regard des exigences.** Document

Comment le prouver: Une vue d'ensemble de l'architecture dans le dépôt (par exemple des diagrammes arc42 ou C4 sous forme d'image accompagnée de texte), ainsi que des fiches de décision d'architecture rassemblant les alternatives évaluées.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.4.2 **Les résultats de la définition de l'architecture système sont atteints : l'architecture est évaluée au regard des exigences, les alternatives sont pesées, et le choix est justifié.**

Comment le prouver: Une fiche de décision d'architecture pesant les options et les conséquences, ainsi que les résultats d'une revue d'architecture dans la demande de tirage ou dans un compte-rendu de réunion.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.5 **La Définition de la conception est réalisée : des conceptions existent pour les blocs de construction architecturaux et sont suffisantes pour la mise en œuvre, y compris le modèle de données et les contrats d'interface.**

Comment le prouver: Spécifications d'interface (OpenAPI, schéma GraphQL, définitions de types), scripts de migration de base de données, et notes de conception dans la demande de tirage. Le code lui-même, avec ses types et ses limites de modules, constitue une preuve de conception acceptable dès lors que les décisions sont documentées.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.5.2 **Les résultats de la définition de la conception sont atteints : les caractéristiques de conception sont traçables jusqu'à l'architecture et la faisabilité est confirmée.**

Comment le prouver: Références de la demande de tirage vers la fiche de décision d'architecture et vers l'exigence, et prototypes ou explorations techniques (spikes) accompagnés d'une note de résultat comme preuve de faisabilité.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.6 **L'Analyse système est réalisée : des questions techniques sont examinées à l'aide de méthodes analytiques afin d'étayer les décisions relatives à l'architecture, à la conception ou aux risques.**

Comment le prouver: Mesures de charge et résultats de tests de performance (benchmarks), analyses de coûts ou d'arbitrages, notes de modélisation des menaces. Les résultats sont rattachés à la fiche de décision d'architecture correspondante sous forme de brève note d'analyse.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.7 **La Mise en œuvre est réalisée : les éléments logiciels sont réalisés conformément à la conception et aux règles de codage, et les résultats sont placés sous maîtrise de configuration.**

Comment le prouver: Historique Git avec des commits référençant des tickets, règles d'analyse de code (linter) et de formatage comme condition de fusion, et compilations réussies dans le pipeline d'intégration continue.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.7.3 **Les activités de mise en œuvre sont réalisées : des normes de codage sont définies, la réalisation est revue, et les éléments sont mis à disposition pour l'intégration.**

Comment le prouver: Fichiers de configuration de l'analyseur de code (linter) et du formateur dans le dépôt, revues de demandes de tirage validées, et artefacts de compilation publiés dans le registre.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.8 **L'Intégration est réalisée : les éléments logiciels sont assemblés progressivement en un système opérationnel et les interfaces sont vérifiées.**

Comment le prouver: Exécutions d'intégration continue à chaque fusion, tests d'intégration et tests de contrat sur les interfaces, ainsi que les journaux de compilation horodatés.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.8.2 **Les résultats de l'intégration sont atteints : la stratégie d'intégration est définie, les interfaces sont cohérentes, et les défauts d'intégration sont enregistrés et résolus.**

Comment le prouver: La définition du pipeline dans le dépôt tenant lieu de stratégie d'intégration, des exécutions en échec suivies d'exécutions réussies comme preuve de résolution des défauts, et des tickets d'anomalie avec le commit de correction.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.9 **La Vérification est réalisée : pour chaque exigence spécifiée, il existe une preuve vérifiable et traçable que le logiciel tel que construit la satisfait effectivement.** Document

Comment le prouver: Tests unitaires, d'intégration et de non-régression automatisés avec un rapport de test issu du pipeline d'intégration continue, analyse statique, et revues de code. Le rapport de test par étiquette de version constitue la preuve centrale.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.9.2 **Les résultats de la vérification sont atteints : les critères de vérification sont définis, les contrôles sont exécutés, et les écarts sont enregistrés.**

Comment le prouver: Seuils d'acceptation définis (par exemple une couverture de tests minimale, l'absence de constat critique), rapports de test archivés, et un ticket pour chaque écart.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.9.3 **Les activités de vérification sont réalisées : le périmètre et les méthodes de vérification sont planifiés, exécutés, et les résultats sont analysés.**

Comment le prouver: Un plan de test ou une stratégie de test sous forme d'une brève page dans le dépôt, la configuration du pipeline avec ses étapes de vérification, et l'analyse figurant dans l'approbation de version.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.10 **Le processus de Transition est réalisé : le système est installé dans son environnement cible, remis prêt à l'emploi, et sa capacité opérationnelle est confirmée.** Document

Comment le prouver: Un manuel opérationnel de déploiement (runbook), des journaux de déploiement issus du pipeline, des tests de fumée (smoke tests) après la mise en production, et un plan de retour arrière documenté. Les notes de version attestent de la remise aux utilisateurs et à l'exploitation.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.11 **La Validation est réalisée : dans les conditions d'utilisation prévues, il est démontré que le logiciel répond effectivement aux besoins initiaux des parties prenantes.** Document

Comment le prouver: Tests de recette avec le client, une phase bêta ou pilote avec retours d'utilisateurs, et critères d'acceptation dans le ticket marqués comme satisfaits. Les données d'usage issues de l'exploitation apportent un appui supplémentaire à la validation.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.11.2 **Les résultats de la validation sont atteints : les critères de validation sont définis, la validation a lieu dans un environnement représentatif, et les écarts sont enregistrés.**

Comment le prouver: Un procès-verbal de recette avec date et participants, résultats de l'environnement de préproduction ou pilote, et tickets de retour avec leur statut de résolution.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.12 **Le processus d'Exploitation est en place : le système est exploité conformément à sa destination, il est surveillé, et les utilisateurs sont assistés.**

Comment le prouver: Supervision et alertes avec des seuils définis, tickets d'incident avec des délais de réponse, et une boîte de support ou un canal de tickets. Un manuel d'exploitation de quelques pages suffit pour les petites organisations.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.12.2 **Les résultats de l'exploitation sont atteints : la stratégie d'exploitation et ses mesures sont définies, les incidents sont enregistrés, et la performance est évaluée au regard des attentes.**

Comment le prouver: Indicateurs de disponibilité et d'erreurs issus de la supervision, journaux d'incidents, et analyses post-incident (postmortems) accompagnées des actions qui en découlent.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.13 **Le processus de Maintenance est en place : les corrections de défauts, les adaptations et les améliorations sont enregistrées, évaluées, mises en œuvre, et publiées sans compromettre l'intégrité du système.** Document

Comment le prouver: Tickets d'anomalie et de changement avec priorité et cause racine, branches de correctif urgent (hotfix), tests de non-régression avant publication, et notes de version par version de maintenance. Les mises à jour régulières des dépendances sont attestées par l'historique du fichier de verrouillage (lockfile).

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.13.2 **Les résultats de la maintenance sont atteints : les changements sont analysés, leur impact est évalué, et l'efficacité de la correction est confirmée.**

Comment le prouver: Une analyse d'impact dans le ticket, une exécution de tests de non-régression après la correction, et une confirmation par la personne à l'origine du signalement. Les mises à jour de sécurité sont attestées par des rapports d'analyse de vulnérabilités indiquant la date de correction.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.14 **Le processus de Retrait est en place : le retrait de systèmes ou de parties de systèmes est planifié, les données sont migrées ou supprimées, et les parties concernées sont informées.** Document

Comment le prouver: Un plan de retrait avec des dates, des preuves de migration ou de suppression des données (y compris les sauvegardes), un enregistrement de l'arrêt de l'infrastructure, et l'annonce aux utilisateurs ou aux clients.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

6.4.14.2 **Les résultats du retrait sont atteints : les obligations de conservation sont respectées, les environnements sont arrêtés, et la responsabilité du système prend fin.**

Comment le prouver: Une archive des données soumises à des obligations de conservation avec la durée de conservation indiquée, un ticket pour l'arrêt des ressources avec une date, et une confirmation de suppression de la part du fournisseur d'hébergement.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

A Conformité et adaptation (Tailoring) 6

4.1 **Le périmètre d'application est défini (article 4.1, Généralités) : il est déterminé si la norme s'applique à un projet, à une organisation ou à un accord, et quels processus sont couverts.** Document

Comment le prouver: Une brève déclaration de périmètre dans la documentation de processus nommant le projet, le produit et les groupes de processus appliqués. Une page dans le dépôt suffit.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.2 **La variante de conformité choisie est déclarée : soit la conformité totale à l'ensemble des résultats des processus revendiqués, soit la conformité totale à l'ensemble des tâches des processus revendiqués.** Document

Comment le prouver: Une déclaration de conformité sous forme de document distinct nommant la variante et renvoyant aux preuves de chaque processus. Cela produit une autodéclaration, et non un certificat.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.3 **Lorsque la conformité adaptée est revendiquée, l'adaptation est documentée : il est précisé quels processus, tâches ou résultats ont été omis, modifiés ou ajoutés.** Document

Comment le prouver: Un tableau d'adaptation comportant les colonnes processus, décision (adopté, adapté, omis), justification, et risque. Pour les petites équipes, ce document constitue la preuve décisive.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

5 **Les concepts clés sont appliqués à l'organisation : la relation entre le modèle de cycle de vie, la structure du projet et les processus utilisés est présentée de manière compréhensible.**

Comment le prouver: Une page de synthèse mettant en correspondance l'approche choisie (par exemple itérative, livraison continue) avec les 30 processus. Un diagramme accompagné d'un tableau suffit.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

A.1 **L'adaptation est traitée comme une décision délibérée : l'adaptation est autorisée, mais elle n'est appliquée que lorsque les contraintes du projet la justifient, et elle est toujours justifiée.** Document

Comment le prouver: Une colonne de justification dans le tableau d'adaptation faisant référence à la taille du projet, à la classe de risque, à la situation contractuelle ou aux exigences légales. Des suppressions générales sans justification ne constituent pas une adaptation valable.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

A.2 **Le processus d'adaptation est réalisé : les facteurs d'influence sont identifiés, les parties prenantes sont impliquées, et les décisions d'adaptation sont prises et documentées.** Document

Comment le prouver: Un enregistrement ou un ticket pour la décision d'adaptation nommant les rôles impliqués, ainsi que le tableau d'adaptation versionné dans Git. Les effets sur les risques sont notés dans le registre des risques.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

ISO/IEC 25010:2023, Ingénierie des systèmes et du logiciel, SQuaRE, Modèle de qualité du produit

Modèle de qualité pour les produits logiciels

40 exigences. Il n'existe pas de certificat accrédité pour cette norme. Version de 07/2026.

L'ISO/IEC 25010:2023 est un modèle de qualité, pas une norme d'exigences avec des clauses auditables, et ne peut donc pas être certifiée par accréditation. Le modèle sert de grille pour formuler les exigences de qualité de manière complète ; la mesure suit l'ISO/IEC 25020 et l'évaluation suit l'ISO/IEC 25040, et le résultat est un rapport d'évaluation traçable, pas un certificat. Il n'est pas permis de faire de la publicité pour une prétendue certification ISO 25010.

Entreprise	Date	Établi par

1 Aptitude fonctionnelle 3

1.1 L'ensemble des fonctions couvre toutes les tâches spécifiées et les objectifs des utilisateurs pour lesquels le produit est utilisé (Exhaustivité fonctionnelle). Les écarts entre les exigences documentées et les fonctions mises en œuvre sont connus et évalués.

Comment le prouver: La preuve provient d'une matrice de traçabilité reliant chaque exigence à un cas de test, mesurable comme taux de couverture des exigences en pourcentage. Dans les petites équipes, un outil de suivi des tickets avec une étiquette par exigence et un test de recette par objectif utilisateur suffit.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

1.2 Pour les entrées définies, le produit fournit des résultats avec le degré de précision requis (Exactitude fonctionnelle). Les tolérances et écarts admissibles sont spécifiés.

Comment le prouver: La grandeur mesurable est la densité de défauts pour mille lignes de code ou par fonction, étayée par des tests unitaires et d'intégration comparant les résultats attendus aux résultats obtenus, ainsi que par les statistiques d'anomalies de l'outil de suivi. Pour la logique de calcul, conservez des jeux de données de référence avec les résultats attendus.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

1.3 Les fonctions fournies soutiennent les tâches des utilisateurs sans étapes superflues et correspondent à l'usage prévu (Pertinence fonctionnelle).

Comment le prouver: La preuve provient du taux de réussite des tâches et du nombre d'étapes par tâche utilisateur lors d'un test d'utilisabilité restreint avec cinq participants, consigné par tâche. Les données d'usage ou les demandes d'assistance révélant des détours complètent le tableau.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

2 Efficacité de performance 3

2.1 Les temps de réponse, de traitement et de débit restent inférieurs aux seuils définis (Comportement temporel). Les seuils sont définis pour les scénarios d'utilisation typiques.

Comment le prouver: Mesurable via le temps de chargement et la latence d'interaction relevés avec Lighthouse ou Web Vitals (LCP, INP), ainsi que les temps de réponse serveur issus des journaux au 95e centile. Consignez les seuils par scénario dans un court budget de performance.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

2.2 **La consommation de temps processeur, de mémoire, de stockage et de bande passante réseau reste dans les limites définies (Utilisation des ressources).**

Comment le prouver: La preuve provient des valeurs de supervision du CPU, de la RAM et du volume de données en charge, ainsi que de la taille du paquet livré issue du rapport de compilation. Pour les petites entreprises, un relevé de mesure issu d'un test de charge par rapport aux limites du matériel cible suffit.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

2.3 **Le nombre maximal d'utilisateurs, de transactions ou d'enregistrements simultanés est connu et respecté (Capacité).**

Comment le prouver: La preuve provient d'un test de charge (par exemple avec k6 ou JMeter) avec un point de saturation documenté et le point à partir duquel le taux d'erreurs commence à augmenter. Résumez le résultat dans un court rapport de capacité indiquant la limite réelle et la demande prévue.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

3 Compatibilité 2

3.1 **Le produit fonctionne dans un environnement partagé sans nuire aux autres produits (Coexistence).**

Comment le prouver: La preuve provient d'un test réalisé dans l'environnement cible avec d'autres logiciels s'exécutant en parallèle, en consignant les conflits de ports, l'utilisation mémoire et les accès aux fichiers. La conteneurisation avec des limites de ressources définies est la voie pratique pour les petites équipes.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

3.2 **L'échange de données avec d'autres systèmes fonctionne via des interfaces et des formats définis (Interopérabilité).**

Comment le prouver: La preuve provient d'une description d'interface versionnée (OpenAPI ou fichier de schéma) et de tests de contrat par rapport à cette description. Vérifiez les formats d'import et d'export avec de véritables fichiers d'exemple et conservez les résultats.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4 Capacité d'interaction 8

4.1 **Les utilisateurs peuvent déterminer sans aide extérieure si le produit correspond à leurs besoins (Facilité de reconnaissance de la pertinence). L'objectif et le bénéfice ressortent clairement de l'interface et des textes du niveau d'entrée.**

Comment le prouver: La preuve provient d'un court test de première impression : les participants décrivent l'objectif du produit après 30 secondes et le taux de réussite est consigné. Des captures d'écran de la page d'accueil et des écrans d'intégration viennent appuyer le constat.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.2 **Les nouveaux utilisateurs atteignent les tâches principales dans un délai défini et avec un nombre limité d'erreurs (Facilité d'apprentissage).**

Comment le prouver: Mesurable comme le temps jusqu'à la première tâche principale réussie et le nombre de tentatives échouées par tâche, relevé lors d'un test avec cinq nouveaux utilisateurs. Documentez l'intégration, la visite guidée ou les données de démonstration comme mesures d'accompagnement.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.3 **L'exploitation est efficace et maîtrisable, et les tâches récurrentes peuvent être réalisées sans détour (Facilité d'exploitation).**

Comment le prouver: La preuve provient d'une analyse du parcours de clics par tâche standard (étapes cibles par rapport aux étapes réelles) et d'une exploitation possible au clavier seul, sans souris. Le journal du test d'utilisabilité avec le taux d'abandon par tâche complète le dossier.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.4 **Le produit prévient les erreurs d'exploitation ou les rend réversibles (Protection contre les erreurs utilisateur). Les actions critiques exigent une confirmation ou peuvent être annulées.**

Comment le prouver: La preuve provient de cas de test avec des saisies invalides et des valeurs limites, ainsi que de la validation, de la boîte de dialogue de confirmation et de la fonction d'annulation. Évaluez les saisies erronées du journal de production et suivez leur fréquence comme indicateur.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.5 **L'interaction est conçue pour être attrayante et maintient les utilisateurs motivés à poursuivre la tâche (Engagement de l'utilisateur).**

Comment le prouver: Mesurable via le taux de retour, la durée de session et le taux d'achèvement des processus engagés, tirés de statistiques d'usage respectueuses de la vie privée. Une courte enquête de satisfaction avec une échelle d'évaluation suffit en complément, y compris pour les petites entreprises.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.6 **Le produit est utilisable par des personnes aux capacités, langues et contextes culturels différents (Inclusivité).**

Comment le prouver: La preuve provient d'une vérification par rapport aux WCAG 2.2 niveau AA : ratios de contraste, navigation au clavier, libellés pour lecteurs d'écran, testés avec axe ou l'audit d'accessibilité de Lighthouse plus un échantillon manuel. Démontrez la couverture linguistique et la localisation par l'exhaustivité des fichiers de traduction.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.7 **Les utilisateurs reçoivent une aide adaptée à la tâche quand ils en ont besoin, y compris dans les situations d'erreur et les cas limites (Assistance à l'utilisateur).**

Comment le prouver: La preuve provient du manuel, des textes d'aide contextuelle et de messages d'erreur compréhensibles qui suggèrent une action suivante. Indicateur : la part des demandes d'assistance que l'aide existante aurait pu résoudre, tirée du système de tickets.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

4.8 **L'interface s'explique d'elle même : les états, les effets des actions et les saisies requises sont reconnaissables sans connaissance préalable (Caractère auto descriptif).**

Comment le prouver: La preuve provient d'une revue de l'interface par rapport à une liste de contrôle : libellés explicites, état du système visible, indicateurs de progression et de chargement, unités sur les champs numériques. Un journal d'observation issu du test utilisateur renforce le dossier.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

5 Fiabilité 4

5.1 **En exploitation normale, le produit fonctionne sans défaillance causée par des défauts du logiciel (Absence de défauts).**

Comment le prouver: Mesurable via la densité de défauts et le temps moyen entre pannes (MTBF) tirés des journaux de production, ainsi que via la couverture des tests comme indicateur préventif. La supervision des erreurs, comme Sentry, fournit les données brutes avec peu d'effort.

◯ Ouvert ◯ En cours ◯ Satisfait ◯ Non applicable

Preuve / justification

5.2 **Le produit est accessible et utilisable pendant les plages horaires convenues (Disponibilité). L'objectif de disponibilité est défini.**

Comment le prouver: La preuve provient de la mesure du temps de fonctionnement par une supervision externe, exprimée en disponibilité en pourcentage par mois et comparée à l'objectif convenu. Consignez les interruptions avec leur durée et leur cause dans un journal d'incidents simple.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

5.3 **En présence de défaillances matérielles ou logicielles, le produit reste dans un état défini et utilisable (Tolérance aux pannes).**

Comment le prouver: La preuve provient de cas de test avec des pannes simulées (réseau indisponible, base de données indisponible, système tiers ne répondant pas) et du comportement documenté, par exemple mode hors ligne, nouvelle tentative ou affichage de repli. Consignez les résultats dans un journal de tests.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

5.4 **Après une défaillance, les données et l'état de fonctionnement peuvent être rétablis dans le délai défini (Aptitude au rétablissement).**

Comment le prouver: Les grandeurs mesurables sont le délai de rétablissement visé (RTO) et la perte de données maximale tolérable (RPO), démontrées par un test de restauration effectivement réalisé avec date et durée. Pour les petites entreprises, un test de restauration semestriel avec un journal succinct suffit.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6 Sécurité 6

6.1 **Les données ne sont accessibles qu'aux personnes et systèmes autorisés (Confidentialité). Les droits d'accès sont accordés selon le principe du besoin d'en connaître.** Document

Comment le prouver: La preuve provient d'un concept de rôles et de permissions, du chiffrement en transit et au repos, et de tests confirmant que l'accès est refusé sans autorisation. Un contrôle des en têtes et du TLS de l'application complète le dossier.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.2 **Les modifications non autorisées des programmes et des données sont empêchées ou détectées (Intégrité).**

Comment le prouver: La preuve provient de sommes de contrôle ou de signatures des artefacts, de la validation des saisies et d'une analyse des dépendances (npm audit, Dependabot) montrant que les anomalies ont été corrigées. Les contraintes de base de données et les journaux de migration renforcent ce point.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.3 **Les actions réalisées ne peuvent pas être niées ultérieurement (Non répudiation).** Document

Comment le prouver: La preuve provient de journaux infalsifiables avec horodatage et auteur, et de signatures lorsque la transaction a une portée juridique. Une destination de journalisation centrale et protégée en écriture suffit pour les petites entreprises.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.4 **Les actions dans le système peuvent être rattachées sans ambiguïté à une personne ou un service responsable (Imputabilité).** Document

Comment le prouver: La preuve provient de journaux d'audit portant un identifiant utilisateur pour chaque action pertinente pour la sécurité, combinés à une interdiction des comptes partagés. Contrôle ponctuel : suivez une transaction de bout en bout dans le journal et documentez le résultat.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.5 **L'identité des utilisateurs et des systèmes est vérifiée de manière fiable (Authenticité).**

Comment le prouver: La preuve provient de la méthode d'authentification, de la politique de mots de passe et de l'authentification à deux facteurs pour les accès administratifs, ainsi que de la validation des jetons et des certificats. Des cas de test avec des jetons expirés et falsifiés démontrent son efficacité.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

6.6 **Le produit résiste aux attaques et reste opérationnel pendant une attaque (Résistance).**

Comment le prouver: La preuve provient d'une analyse de vulnérabilités, d'une analyse statique de code (SAST) et, lorsque cela est proportionné, d'un test d'intrusion avec une liste de constats et leur statut de correction. Pour les petites entreprises, une analyse automatisée dans le pipeline plus un court test externe annuel est la voie pratique.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

7 Maintenabilité 5

7.1 **Le produit se compose de composants séparés dont la modification affecte le moins possible d'autres composants (Modularité).**

Comment le prouver: Mesurable via le couplage et la cohésion issus d'une analyse statique et via des graphes de dépendances (madge, dependency cruiser) montrant l'absence de cycles. Consignez les limites des modules dans un court schéma d'architecture.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

7.2 **Les composants sont conçus pour pouvoir être réutilisés dans plusieurs contextes (Réutilisabilité).**

Comment le prouver: La preuve provient de bibliothèques ou composants partagés avec leur propre gestion de versions et une interface documentée. Indicateur : la part de composants réutilisés et le niveau de duplication de code issu de l'analyse statique.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

7.3 **Les causes des défaillances et l'impact des modifications prévues peuvent être déterminés efficacement (Analysabilité).**

Comment le prouver: La preuve provient de la journalisation structurée, de la supervision des erreurs et de métriques de code lisibles telles que la complexité cyclomatique issue de SonarQube ou d'ESLint. Indicateur : temps moyen jusqu'à l'identification de la cause racine, tiré du système de tickets.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

7.4 **Les modifications peuvent être effectuées sans introduire de nouveaux défauts et sans dégrader la qualité (Facilité de modification).**

Comment le prouver: La preuve provient des tests de non régression dans le pipeline et du taux d'échec des modifications, c'est à dire la part de défauts introduits par les changements. La revue de code obligatoire et le contrôle de version avec des commits traçables renforcent ce point.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

7.5 **Des critères de test et des tests efficaces peuvent être établis et exécutés pour le produit et ses composants (Testabilité).**

Comment le prouver: Mesurable via le taux de couverture des tests en pourcentage (lignes et branches) issu du rapport de couverture et via la durée d'exécution de la suite de tests. Une exécution automatisée des tests à chaque commit en constitue la preuve.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

8 Flexibilité 4

8.1 Le produit peut être adapté à des environnements différents ou changeants sans devoir être reconstruit (Adaptabilité).

Comment le prouver: La preuve provient d'une configuration maintenue en dehors du code (variables d'environnement, fichiers de configuration) et de tests menés sur plusieurs environnements cibles, navigateurs ou systèmes d'exploitation dans une matrice de test. Consignez le résultat sous forme de tableau de compatibilité.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

8.2 Le produit gère des volumes croissants ou décroissants d'utilisateurs et de données sans baisse inacceptable de performance (Évolutivité).

Comment le prouver: La preuve provient de tests de charge avec un nombre croissant d'utilisateurs, en suivant le temps de réponse et le taux d'erreur, complétés par la stratégie de mise à l'échelle (horizontale ou verticale). Indicateur : temps de réponse à charge simple, double et quintuple.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

8.3 Le produit peut être installé, mis à jour et désinstallé de manière fiable dans l'environnement cible (Facilité d'installation).

Comment le prouver: La preuve provient d'un guide d'installation reproductible et d'un test d'installation automatisé sur un système propre, mesuré en taux de réussite et en temps nécessaire. Effectuez également une fois la désinstallation et le retour arrière, et consignez le résultat.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

8.4 Le produit peut remplacer un autre produit utilisé dans le même but tout en préservant les données et les flux de travail (Remplaçabilité).

Comment le prouver: La preuve provient de fonctions d'import et d'export dans des formats ouverts et d'un test de migration avec des données réelles issues de l'ancien système, vérifié en termes d'exhaustivité et de fidélité des données. Conservez le modèle de données et les scripts de migration sous contrôle de version.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

9 Sécurité fonctionnelle 5

9.1 Le domaine de fonctionnement sûr est défini, et le produit détecte et gère tout écart par rapport à ce domaine (Contrainte opérationnelle).

Comment le prouver: La preuve provient de limites de fonctionnement documentées (valeurs seuils, états admissibles) et de tests vérifiant le comportement en dehors de ces limites. Si le logiciel ne présente aucun potentiel de dommage pour les personnes ou les biens, ce point n'est généralement pas applicable : cette classification doit être justifiée et documentée, et non simplement omise.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

9.2 Les dangers pouvant résulter du produit ou de son utilisation sont identifiés et évalués de façon systématique (Identification des risques). Document

Comment le prouver: La preuve provient d'une analyse des dangers et des risques indiquant la probabilité, la gravité et la mesure d'atténuation pour chaque danger, réalisable sous forme de tableau dans le dossier qualité. Non applicable uniquement avec une justification écrite de l'absence de potentiel de dommage pour les personnes ou les biens.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

9.3 **En cas de dysfonctionnement, le produit passe dans un état sûr (Sécurité en cas de défaillance).**

Comment le prouver: La preuve provient de cas de test avec une défaillance provoquée et de la transition documentée vers l'état sûr, par exemple arrêt, verrouillage ou retour à une valeur par défaut vérifiée. Pour un logiciel sans potentiel de dommage, ce point n'est généralement pas applicable, mais la justification doit tout de même figurer dans l'évaluation.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

9.4 **Le produit avertit suffisamment tôt de conditions dangereuses imminentes pour qu'une réaction soit possible (Alerte de danger).**

Comment le prouver: La preuve provient d'un concept d'alarmes et d'avertissements avec seuils, délai d'anticipation et destinataires, ainsi que d'un journal de test des avertissements déclenchés. Sans potentiel de dommage, ce point n'est généralement pas applicable : la justification doit être documentée.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification

9.5 **L'interaction avec d'autres systèmes ne crée pas de nouveaux dangers (Intégration sûre).**

Comment le prouver: La preuve provient d'une revue des risques liés à l'interface et de tests d'intégration avec les systèmes connectés, y compris le comportement en cas de défaillance d'un système partenaire ou d'envoi de données erronées. Pour un logiciel sans potentiel de dommage, ce point n'est généralement pas applicable, ce qui doit être consigné avec une justification.

○ Ouvert ○ En cours ○ Satisfait ○ Non applicable

Preuve / justification