Concevoir un harnais agentique

Document de référence — cycle de vie complet

Version 2 — étend la V1 aux aspects design, exploitation/maintenance et décommissionnement, et ajoute les référentiels de bonnes pratiques, la gouvernance et les indicateurs de pilotage.

Préparé pour Hugues — 1 août 2026

Clic droit sur le sommaire puis « Mettre à jour les champs » pour l'actualiser dans Word.

1. Objectif et vision

Ce document est la version de référence du harnais agentique : il couvre désormais l'intégralité du cycle de vie d'un projet informatique, du cadrage initial jusqu'au décommissionnement, en passant par le design, la réalisation, le déploiement et l'exploitation. Il sert de fil conducteur pour la mise en œuvre : toute personne qui construit ou fait évoluer ce harnais peut s'y référer pour savoir quelles phases couvrir, quels artefacts produire, quels référentiels respecter, et à quel moment un humain doit trancher.

Trois ajouts distinguent cette version de la précédente. La couverture du design : les artefacts UX/UI, absents de la première version, conditionnent pourtant l'expérience finale autant que l'architecture technique. La couverture de l'exploitation et de la maintenance après la mise en production : un système vit en moyenne bien plus longtemps après son déploiement qu'avant. La couverture du décommissionnement : la fin de vie d'un système est aussi structurante que sa naissance, avec des données à archiver ou détruire, des accès à désactiver, et des obligations légales à respecter. Un principe transversal complète ces ajouts : chaque famille d'artefacts doit être produite en respectant des référentiels reconnus plutôt que des conventions ad hoc, pour que la qualité produite par les agents soit vérifiable par un tiers plutôt que seulement plausible.

2. Cadre de référence du cycle de vie

Le découpage retenu s'inspire des grandes familles de processus définies par la norme ISO/IEC/IEEE 12207 relative aux processus du cycle de vie du logiciel (développement, exploitation, maintenance et retrait), adaptées et complétées pour refléter la pratique courante des projets numériques : ajout d'une phase de design distincte de la conception technique, et détail des artefacts de gouvernance transverses.

Neuf phases structurent ainsi le document : cadrage, design, conception technique, planification, réalisation, vérification, déploiement, exploitation et maintenance, décommissionnement. Une dixième catégorie, transverse, regroupe les artefacts qui traversent toutes les phases plutôt que d'appartenir à une seule.

3. Panorama des artefacts à couvrir

Un projet informatique produit des artefacts à chaque phase de son cycle de vie, y compris après sa mise en production. Le harnais doit couvrir l'ensemble de ce cycle plutôt qu'une phase isolée, car la valeur du système vient largement de la cohérence qu'il maintient entre artefacts en amont et en aval — cadrage jusqu'à décommissionnement compris.

3.1 Cadrage

- Note d'opportunité et objectifs métier

- Cahier des charges (besoins, contraintes, périmètre)
- Analyse des risques et des parties prenantes
- Estimation budgétaire et calendaire de premier niveau

3.2 Design (UX/UI)

- Personas et parcours utilisateurs
- Wireframes et maquettes basse puis haute fidélité
- Prototype interactif
- Charte graphique et design system (composants réutilisables)
- Guide d'accessibilité (conformité RGAA/WCAG)
- Protocole et résultats des tests d'utilisabilité

3.3 Conception technique

- Spécifications fonctionnelles
- Spécifications techniques
- Architecture logicielle (diagrammes de composants, de séquence, de déploiement)
- Décisions d'architecture documentées (ADR — Architecture Decision Records)
- Modèle de données
- Choix technologiques et justification

3.4 Planification

- Backlog produit et user stories
- Découpage en lots ou sprints
- Planning et jalons

3.5 Réalisation

- Code source et configuration d'environnement
- README et documentation d'installation
- Conventions de code et structure du dépôt

3.6 Vérification

- Plan de tests
- Tests unitaires, d'intégration et de bout en bout
- Rapports de couverture

- Revue de sécurité, de conformité et d'accessibilité

3.7 Déploiement

- Pipeline d'intégration et de livraison continues (CI/CD)
- Scripts d'infrastructure as code
- Documentation d'exploitation (runbook) et plan de reprise d'activité

3.8 Exploitation et maintenance

- Plan de maintenance (corrective, évolutive, préventive)
- Journal des incidents et post-mortems
- Notes de version et changelog
- Registre de dette technique
- Tableau de bord de supervision (monitoring, alerting, indicateurs SLO/SLI)
- Plan de gestion des correctifs de sécurité (patch management)
- Rapports d'audit périodiques (sécurité, performance, conformité)

3.9 Décommissionnement

- Plan de décommissionnement (calendrier, dépendances, communication aux utilisateurs)
- Plan de migration, d'export et d'archivage des données
- Procédure de désactivation sécurisée des accès et des services
- Preuve de destruction ou d'anonymisation des données (conformité RGPD)
- Bilan de fin de vie et capitalisation (retour d'expérience, leçons apprises)

3.10 Transverse

- Documentation utilisateur
- Plan de gestion de projet et comptes-rendus
- Matrice de traçabilité exigences ↔ tests
- Registre des décisions et glossaire du projet

4. Référentiels et bonnes pratiques par famille d'artefacts

Produire un artefact ne suffit pas : encore faut-il qu'il respecte des règles reconnues plutôt que des conventions inventées au fil de l'eau par chaque agent. Le tableau suivant associe à chaque famille d'artefacts un ou plusieurs référentiels que les agents doivent connaître et appliquer — soit en les citant explicitement dans l'artefact produit (un ADR par décision d'architecture, par exemple), soit en les utilisant comme grille d'auto-évaluation avant de marquer l'artefact prêt pour validation humaine. Cette exigence doit être injectée directement dans le prompt de chaque agent spécialisé, pas laissée à l'appréciation implicite du modèle.

Famille d'artefacts	Référentiel(s)	Ce qu'il garantit
Design UX/UI	WCAG 2.2 / RGAA, ISO 9241-11 (utilisabilité)	Accessibilité et expérience utilisateur mesurable
Architecture	ADR, C4 model, ISO/IEC/IEEE 42010	Décisions tracées, diagrammes lisibles à plusieurs niveaux de zoom
Spécifications	Exigences identifiables et vérifiables, matrice de traçabilité	Exigences traçables jusqu'aux tests
Code	Guide de style du langage cible, Conventional Commits, revue systématique	Lisibilité, historique exploitable, cohérence d'équipe
Tests	Pyramide de tests, couverture mesurée, OWASP Top 10 (sécurité)	Fiabilité vérifiée plutôt que déclarée
Documentation	Cadre Diátaxis (tutoriel / guide / référence / explication)	Documentation utile selon l'intention du lecteur
Déploiement	Principes "12-factor app", infrastructure as code versionnée	Portabilité et reproductibilité des environnements
Exploitation / maintenance	ITIL (incidents, changements), pratiques SRE (SLO/SLI)	Continuité de service pilotée par des indicateurs
Décommissionnement	RGPD (droit à l'effacement, durées de conservation), ISO/IEC 27001	Fin de vie conforme et sans risque résiduel
Transverse (qualité globale)	ISO/IEC 25010 (caractéristiques de qualité logicielle)	Grille d'évaluation commune à tous les artefacts

Ces référentiels ne sont pas figés : le registre devrait permettre de les faire évoluer (nouvelle version d'une norme, choix différent pour un projet donné) via un fichier de configuration associant type d'artefact et référentiel(s), sans modifier le code de l'orchestrateur.

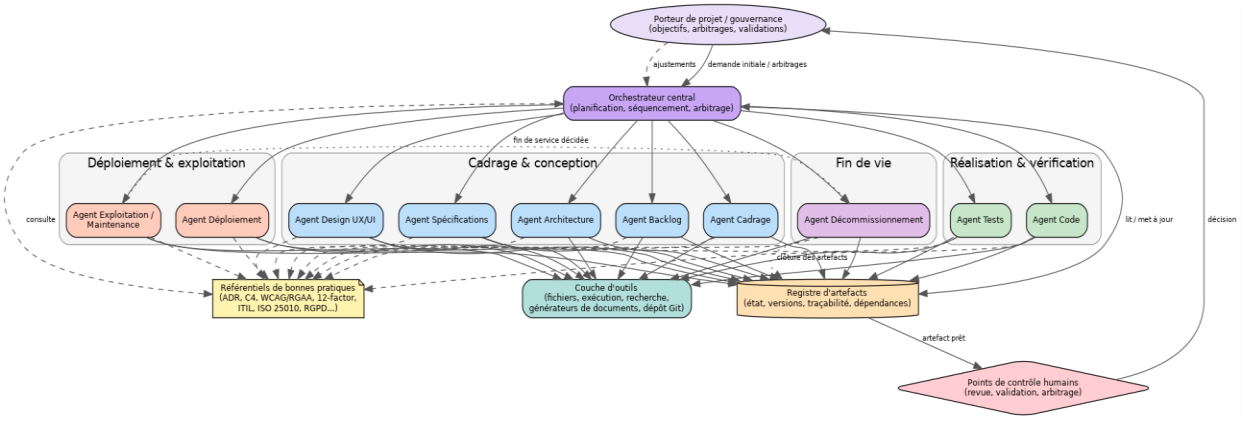
5. Principes d'architecture

Huit principes structurent une architecture agnostique capable de tenir dans la durée, indépendamment du framework choisi pour l'implémenter.

- **Le registre d'artefacts comme source de vérité unique.** Chaque artefact généré (document ou code) est enregistré dans un registre central avec ses métadonnées : type, version, statut (brouillon, en revue, validé), auteur (agent ou humain), et liens vers les artefacts dont il dépend. Aucun agent ne doit maintenir sa propre copie parallèle de l'état du projet.
- **Des agents spécialisés par type d'artefact plutôt qu'un agent généraliste unique.** Un agent "Design" n'a pas les mêmes besoins en outils, en style de sortie ni en critères de qualité qu'un agent "Code" ou "Exploitation". La spécialisation permet des instructions plus précises, une évaluation plus simple, et la possibilité de faire évoluer un agent sans affecter les autres.
- **Des contrats de dépendance explicites entre artefacts.** Les spécifications dérivent du cahier des charges, l'architecture dérive des spécifications, le code dérive de l'architecture et des user stories, les tests dérivent des spécifications et du code. Ces dépendances doivent être déclarées formellement dans le registre afin que toute modification amont déclenche une alerte ou une régénération des artefacts avals concernés.
- **Une boucle agentique générique par agent.** Chaque agent spécialisé suit le même cycle : lecture du contexte et des artefacts dont il dépend, planification, génération, auto-vérification (cohérence, complétude, conformité au référentiel applicable), puis publication dans le registre.
- **Des points de contrôle humains aux transitions critiques.** Certaines transitions (validation du cahier des charges, validation de l'architecture, mise en production, décommissionnement) doivent rester des décisions humaines explicites. Le harnais doit pouvoir suspendre le flux à ces points et reprendre après validation.
- **Des garde-fous de qualité systématiques.** Revue croisée entre agents, linters et analyseurs statiques pour le code, exécution effective des tests générés, vérification de cohérence terminologique entre documents. Un artefact qui n'a pas passé ces contrôles ne doit pas être marqué "validé" dans le registre.
- **La conformité aux référentiels comme critère de qualité systématique.** Chaque agent spécialisé doit connaître le ou les référentiels applicables à son artefact (section 4) et les utiliser comme grille d'auto-vérification avant publication — un artefact qui ignore le référentiel qui lui est associé ne doit pas être considéré comme complet, même s'il est par ailleurs bien écrit.
- **Une couverture du cycle de vie qui ne s'arrête pas à la mise en production.** Le registre et les points de contrôle humains doivent rester actifs après le déploiement : les artefacts d'exploitation (supervision, incidents, dette technique) et de décommissionnement font partie du même registre que les artefacts de conception, avec les mêmes exigences de traçabilité — un système n'est pas "terminé" au sens du harnais tant qu'il n'a pas été décommissionné ou repris par un dispositif équivalent.

6. Architecture proposée

Le schéma ci-dessous illustre l'agencement des composants, désormais organisés en quatre macro-phases : cadrage & conception, réalisation & vérification, déploiement & exploitation, et fin de vie. Il reste indépendant du framework : orchestrateur, registre, agents, outils et référentiels sont des rôles fonctionnels que l'on peut implémenter avec n'importe quelle pile technique.



6.1 Composants

- **Orchestrateur central.** Reçoit la demande initiale, détermine la séquence de génération en fonction des dépendances déclarées, distribue le travail aux agents spécialisés, et gère les reprises en cas d'échec ou de modification amont — y compris en phase d'exploitation, où son rôle devient plus cyclique que linéaire.
- **Registre d'artefacts.** Base d'état versionnée qui stocke chaque artefact, son statut, ses dépendances et son historique, du cadrage au décommissionnement. C'est la mémoire persistante et partagée du projet — elle doit survivre à l'arrêt et au redémarrage du harnais, et rester consultable même après la clôture du système qu'elle documente.
- **Référentiels de bonnes pratiques.** Base de référence (ADR, C4, WCAG/RGAA, 12-factor, ITIL, ISO 25010, RGPD, etc.) que les agents consultent pour produire des artefacts conformes plutôt que plausibles. Voir la section 4 pour le détail par famille d'artefacts.
- **Agents spécialisés.** Dix agents répartis en quatre macro-phases : cadrage, design, spécifications, architecture et backlog pour l'amont ; code et tests pour la réalisation ; déploiement et exploitation/maintenance pour la mise en production et son suivi ; décommissionnement pour la fin de vie. Chacun est instruit avec un contexte, des outils et des référentiels propres à son artefact.
- **Couche d'outils.** Accès aux fichiers, exécution de code, recherche d'information, génération de documents bureautiques, interaction avec un dépôt de code. Les agents ne doivent avoir accès qu'aux outils nécessaires à leur artefact (principe de moindre privilège).
- **Points de contrôle humains.** Interface — même minimale — permettant à un humain d'approuver, rejeter ou amender un artefact avant que le flux ne se poursuive. En exploitation, ces points

deviennent périodiques plutôt que ponctuels ; au décommissionnement, ils impliquent typiquement plus d'une personne (sponsor et référent protection des données).

7. Flux de génération

Exemple de déroulé pour un projet standard, illustrant l'articulation entre génération automatique et validation humaine, de la demande initiale à la mise en production :

- 1. Le porteur de projet formule sa demande initiale (objectifs, contraintes, périmètre pressenti).
- 2. L'agent Cadrage produit une note d'opportunité et un cahier des charges ; point de contrôle humain avant de poursuivre.
- 3. Les agents Design, Spécifications et Architecture travaillent à partir du cahier des charges validé, puis se recoupent pour vérifier leur cohérence mutuelle (le design correspond-il aux exigences fonctionnelles, le modèle de données correspond-il aux deux).
- 4. L'agent Backlog découpe les artefacts validés en user stories et propose un planning.
- 5. L'agent Code génère l'implémentation par lot ; l'agent Tests génère en parallèle les tests correspondants et les exécute réellement.
- 6. L'agent Déploiement produit le pipeline CI/CD et la documentation d'exploitation une fois le code stabilisé.
- 7. Point de contrôle humain final avant mise en production.

Ce flux initial se termine à la mise en production, mais le harnais ne s'arrête pas là. Deux flux supplémentaires prennent le relais.

- **Flux d'exploitation (cyclique).** Une fois en production, l'agent Exploitation/Maintenance opère en boucle continue plutôt qu'en séquence à sens unique : il met à jour le tableau de bord de supervision, consigne les incidents et leurs post-mortems, tient le registre de dette technique, et déclenche des points de contrôle humains périodiques plutôt qu'un point de contrôle unique. Ce flux dure généralement bien plus longtemps que le flux de construction initial.
- **Flux de décommissionnement (terminal).** Déclenché par une décision humaine explicite (fin de contrat, remplacement par un autre système, arrêt d'activité), il mobilise l'agent Décommissionnement, qui produit le plan de fin de vie, orchestre l'archivage ou la destruction des données, et documente le bilan de capitalisation. Ce flux se termine par la clôture des artefacts correspondants dans le registre plutôt que par leur suppression : la trace du projet doit rester consultable même après l'arrêt du système lui-même.

À chaque étape, une modification déclenchée en amont doit se propager automatiquement aux artefacts avals concernés, via les dépendances déclarées dans le registre — c'est le mécanisme qui évite la dérive documentaire, l'un des risques principaux de ce type de système.

8. Gouvernance et rôles

Le harnais automatise la production des artefacts, pas la décision. Une répartition claire des responsabilités entre agents et humains évite deux dérives symétriques : un système où l'humain valide sans vraiment lire (le point de contrôle devient une formalité), et un système où l'humain doit tout réécrire (le harnais n'apporte plus de valeur). La répartition suivante est une base à adapter au contexte de gouvernance de chaque organisation :

- **Cadrage.** Validation par le sponsor ou le commanditaire du projet.
- **Design.** Validation par un référent UX ou produit, appuyée sur les tests d'utilisabilité.
- **Architecture.** Validation par un architecte ou un tech lead.
- **Code et tests.** Revue par un pair humain avant fusion, même lorsque les tests sont exécutés automatiquement par l'agent.
- **Déploiement en production.** Validation par un release manager ou un rôle équivalent.
- **Exploitation.** Décisions courantes déléguées à l'agent (application de correctifs mineurs, mise à jour de la documentation d'exploitation) ; décisions structurantes (refonte, montée de version majeure) systématiquement remontées à un humain.
- **Décommissionnement.** Validation conjointe du sponsor et d'un référent protection des données, en raison des obligations légales de conservation ou de destruction qui s'appliquent.

9. Enjeux et bonnes pratiques

- **Cohérence inter-artefacts.** Le risque principal n'est pas qu'un agent produise un mauvais artefact isolé, mais que plusieurs artefacts dérivent silencieusement les uns par rapport aux autres. Les contrats de dépendance et les revues croisées entre agents sont la principale parade.
- **Traçabilité et gestion du changement.** Chaque artefact doit pouvoir être retracé jusqu'à sa source et chaque modification amont doit déclencher une analyse d'impact sur les artefacts avals, avec re-validation humaine si nécessaire.
- **Fiabilité et vérifiabilité.** Les agents de génération de code et de tests doivent être ancrés sur une exécution réelle (compilation, exécution des tests, linters) plutôt que sur une simple génération de texte plausible.
- **Sécurité et confinement.** L'exécution de code généré par les agents doit se faire dans un environnement isolé, avec des permissions limitées, en particulier pour les artefacts de déploiement et d'infrastructure.
- **Coût et parallélisation.** Certains artefacts peuvent être générés en parallèle, d'autres sont strictement séquentiels. L'orchestrateur doit exploiter le parallélisme possible sans casser les dépendances réelles.
- **La dette technique et l'obsolescence.** Un système en exploitation accumule des écarts entre ce que le registre décrit et ce qui tourne réellement (correctifs d'urgence non documentés, dépendances obsolètes). Le registre de dette technique doit être un artefact de première classe, pas une note informelle, pour que cet écart reste mesurable plutôt que silencieux.
- **La conformité en fin de vie.** Le décommissionnement engage des obligations qui dépassent le strict cadre technique : durées légales de conservation de certaines données, droit à l'effacement, portabilité pour les utilisateurs concernés. Ces obligations doivent être vérifiées avant la destruction de tout artefact ou donnée, pas après.

10. Indicateurs de pilotage et amélioration continue

Le harnais lui-même doit pouvoir être évalué, au même titre que les artefacts qu'il produit. Quelques indicateurs simples suffisent à commencer : le taux d'artefacts validés dès le premier passage (révèle la qualité réelle de la génération, pas seulement sa vitesse), le délai moyen entre la création d'un artefact et sa validation, le nombre de régénérations déclenchées par une modification amont (révèle la stabilité des dépendances déclarées), et le coût en tokens par artefact produit.

Le bilan de capitalisation produit lors d'un décommissionnement (section 3.9) ne devrait pas rester un document isolé : les leçons qui y sont consignées doivent alimenter les prompts des agents amont pour les projets suivants — c'est ce qui transforme une suite de projets ponctuels en un système qui s'améliore réellement dans la durée.

11. Feuille de route de mise en œuvre

- Étape 1 — Définir le schéma du registre d'artefacts : types d'artefacts, métadonnées, statuts, modèle de dépendances.
- Étape 2 — Orchestrateur minimal avec les agents Cadrage, Spécifications et Architecture, et un point de contrôle humain.
- Étape 3 — Traçabilité complète et propagation des changements amont → aval.
- Étape 4 — Ajouter l'agent Design UX/UI, rattaché au registre au même titre que les autres artefacts de conception.
- Étape 5 — Étendre aux agents de réalisation (Code, Tests) avec exécution réelle et vérification automatique.
- Étape 6 — Ajouter les agents Déploiement et Documentation.
- Étape 7 — Ajouter l'agent Exploitation/Maintenance et le flux cyclique de supervision post mise en production.
- Étape 8 — Ajouter l'agent Décommissionnement et le flux terminal correspondant.
- Étape 9 — Industrialiser : parallélisation, observabilité, indicateurs de pilotage (section 10), boucle d'amélioration continue.

Cette progression permet de valider le mécanisme central — cohérence et traçabilité entre artefacts — sur un périmètre restreint avant de l'étendre à l'ensemble du cycle de vie, de la naissance à la fin de vie du système.

12. Tableau de synthèse — fil conducteur

Vue d'ensemble à utiliser comme référence rapide lors de la mise en œuvre ou de l'audit du harnais.

Phase	Artefacts clés	Agent(s)	Point de contrôle
Cadrage	Note d'opportunité, cahier des charges, analyse des risques	Agent Cadrage	Validation du périmètre et du budget
Design UX/UI	Personas, wireframes, prototype, design system, guide d'accessibilité	Agent Design UX/UI	Test d'utilisabilité + validation accessibilité
Conception technique	Spécifications, architecture, ADR, modèle de données	Agent Spécifications, Agent Architecture	Revue d'architecture
Planification	Backlog, user stories, planning	Agent Backlog	Validation du découpage et des priorités
Réalisation	Code source, README	Agent Code	Revue de code + linters
Vérification	Plan de tests, tests automatisés, rapports de couverture	Agent Tests	Exécution réelle des tests, seuils de couverture
Déploiement	Pipeline CI/CD, infrastructure as code	Agent Déploiement	Validation avant mise en production
Exploitation & maintenance	Plan de maintenance, supervision, changelog, dette technique	Agent Exploitation/Maintenance	Revues périodiques, SLO respectés
Décommissionnement	Plan de fin de vie, archivage/export, preuve de destruction des données	Agent Décommissionnement	Validation du sponsor + du référent RGPD