# Testing Agent

Notice technique

Version 0.8.3

28 octobre 2026

Moamen Elmasry

Gratuit · moamen.fr/fr/testing-agent/

# Sommaire

Résumé

1. Le problème

2. La méthode

3. Architecture

4. Données et fichiers

5. Règles et décisions humaines

6. Azure DevOps

7. Installation et désinstallation

8. Limites et usage conseillé

9. Historique des versions

Annexe. Les huit compétences

# Résumé

Le Testing Agent est un serveur MCP, c'est-à-dire un petit programme que l'assistant IA de votre éditeur appelle pour utiliser des outils. Il équipe l'assistant pour qu'il sache tester : la méthode, les outils et les règles d'un testeur, appliqués aux user stories d'une équipe, y compris celles dont une partie est générée par IA. L'assistant devient testeur, et guide des testeurs.

Pour chaque critère d'une user story, il distingue ce qui se teste au résultat exact (code classique), ce qui se mesure sur un jeu de référence (réponses générées) et ce qui mêle les deux. Il propose les cas par phase (composant, intégration, système, évaluation IA), écrit et lance les tests de code, lit les rapports JUnit et les builds Azure DevOps, guide les tests d'écran pas à pas, prépare la preuve et le brouillon de bug, et tient un suivi en fichiers texte dans le dépôt du projet. Une interface web locale montre à chacun ce qui l'attend, le fait passer ses tests pas à pas et garde ses captures comme preuve.

Approuver un cas, poser un résultat, accepter un écart et décider d'un Go / No Go restent des gestes humains. Le registre de suivi refuse certaines écritures et enregistre chaque décision au nom de la personne qui la prend.

| En un coup d'œil | |
|---|---|
| **Type** | Serveur MCP local, en Python 3.10 ou plus |
| **Fonctionne avec** | Installation prévue pour GitHub Copilot, Claude Code, Codex et Cursor |
| **Outil de suivi** | Azure DevOps et Azure Test Plans, ou aucun |
| **Données** | Elles restent sur le poste. À l'exécution, le Testing Agent n'ouvre aucune connexion vers l'extérieur. |
| **Prix** | Gratuit pendant la phase de test ; s'il devient payant, un préavis de 30 jours, et personne ne paie sans l'avoir accepté. Sans revente ni redistribution. |
| **Support** | Un auteur, sans engagement |
| **Version** | 0.8.3, 28 octobre 2026 |

### À savoir avant de l'adopter

- Conçu pour l'essai et l'usage en équipe. Faites-le valider par votre service informatique avant un déploiement.
- Il ne vérifie pas l'identité d'une personne : la trace repose sur ce qu'elle déclare.
- Les fichiers ne sont pas signés : à l'installation à la main, Windows peut afficher un avertissement ou bloquer le fichier (chapitre 7).
- Le dépôt du code n'est pas public.

# 1. Le problème

Une user story mélange souvent plusieurs sortes de critères. Certains se testent : un résultat exact est attendu (un lien expire après trente minutes). D'autres se mesurent : plusieurs réponses générées sont acceptables, et seule une évaluation sur un jeu de référence permet de juger (un résumé reprend les décisions prises). Dans un produit hybride, les deux coexistent dans la même user story.

Les outils de test classiques attendent un résultat exact et ne savent traiter que les premiers. Pour les seconds, l'équipe fixe alors un seuil au hasard, ou juge à l'œil, sans trace, ou fond les deux dans une moyenne qui masque un échec.

Dans la pratique de l'auteur, la recette d'un produit IA reposait presque entièrement sur une personne : elle classait les cas à la main, les créait dans l'outil, exécutait la plupart des tests et gardait la méthode pour elle. Le Testing Agent met cette méthode à la portée de toute l'équipe, dans l'assistant que chacun utilise déjà. Un skill seul est un texte de consignes ; ici, un serveur local agit sur les fichiers du projet, garde l'état et refuse certaines écritures. L'assistant reste lui-même ; sa capacité à tester est augmentée, comme un exosquelette augmente celle de celui qui le porte.

*Ce constat vient de l'expérience de l'auteur. Il n'a pas fait l'objet d'une enquête.*

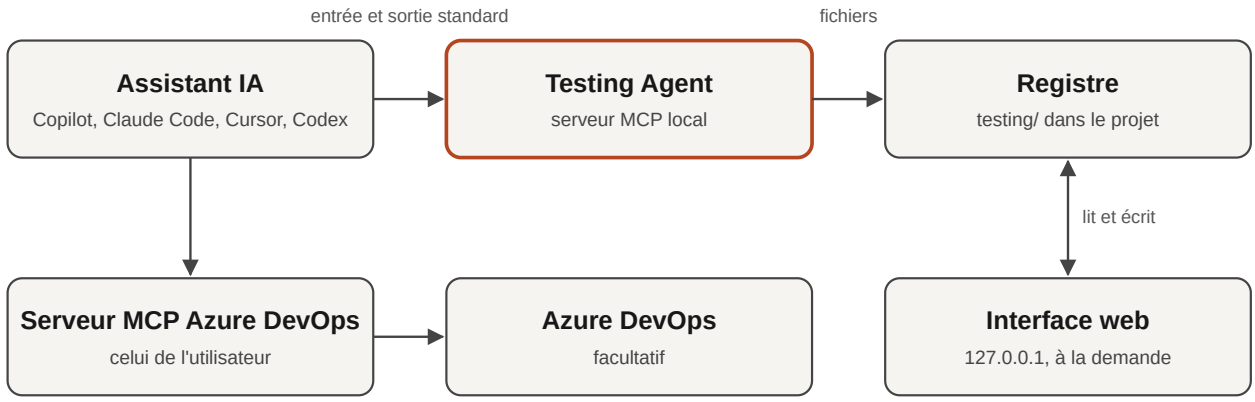
# 2. La méthode

La méthode est écrite en 21 notes, livrées dans l'archive (en anglais). Elle repose sur la pratique de l'auteur et sur les niveaux de test classiques. Elle ne se réclame d'aucune norme.

| Notion | Contenu |
|---|---|
| **Definition of Done** | Ce qui se teste, au résultat exact. Un test instable compte comme un échec. |
| **Definition of Trustworthy** | Ce qui se mesure : plusieurs réponses sont acceptables, jugées sur un jeu de référence annoté, contre un seuil approuvé par une personne. Sans seuil validé, le verdict est « pas prêt ». |
| **Déterministe** | Un seul résultat correct, vérifié exactement. |
| **Probabiliste** | Plusieurs réponses acceptables, mesurées sur un jeu de référence. Des vétos arrêtent une erreur critique, des tranches empêchent une moyenne de masquer un groupe : une bonne moyenne n'annule jamais un échec critique. |
| **Hybride** | La partie exacte est testée exactement, la partie générée est mesurée à part. |
| **Quatre phases** | Composant, intégration, système, évaluation IA. Le registre suit aussi un niveau d'acceptation. |

Un produit hybride n'est accepté que si chaque porte applicable est franchie séparément : la Definition of Done, la Definition of Trustworthy et, quand elle s'applique, l'acceptation par les utilisateurs. Les résultats ne sont jamais fondus en un seul score. Un juge IA, lorsqu'il est utilisé, doit lui-même être testé, calibré et versionné.

# 3. Architecture




Le Testing Agent dialogue avec l'assistant par l'entrée et la sortie standard (protocole MCP) : il n'écoute sur aucun port. Seule l'interface web, lancée à la demande, écoute sur 127.0.0.1. Il lit et écrit des fichiers du projet.

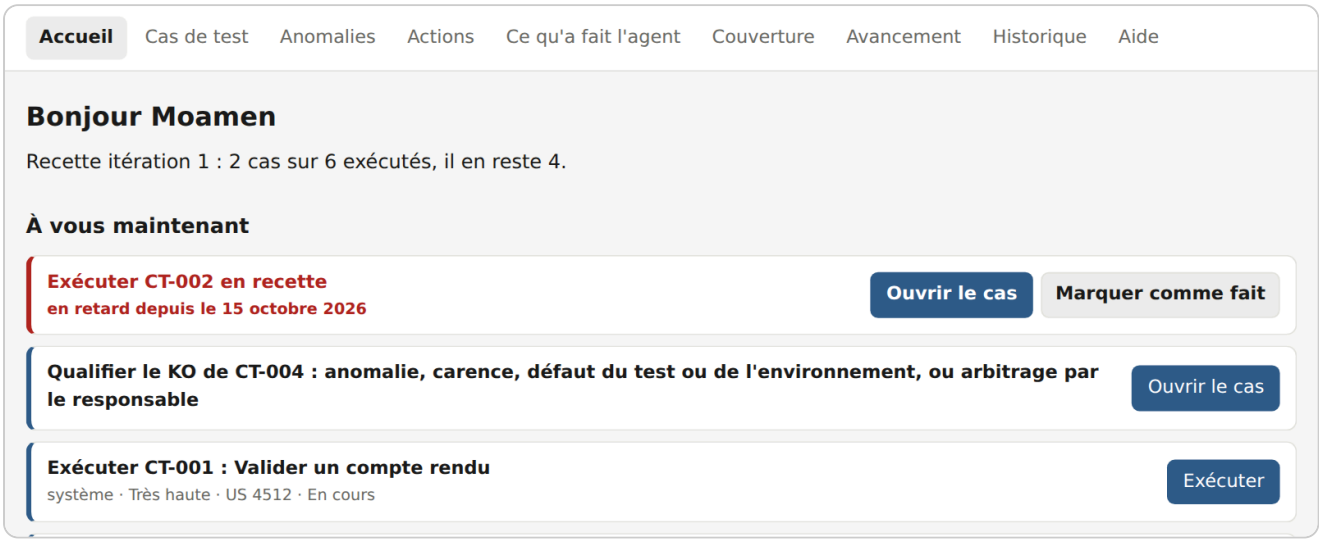
Il n'appelle pas Azure DevOps lui-même : l'assistant le fait avec le serveur MCP Azure DevOps de l'utilisateur, et le Testing Agent prépare ou enregistre ce qui s'y rapporte.

### L'interface web locale

Lancée à la demande par l'assistant, elle sert aux personnes à voir et à décider elles-mêmes. Chaque écriture y passe par les mêmes règles que celles de l'agent, au nom de la personne qui l'a saisie.

- Accueil : ce qui vous attend maintenant, du plus urgent au moins urgent, ce qui attend les autres, et l'avancement de la campagne par phase.

- Exécuter un test : une étape à la fois, avec les valeurs exactes à saisir prêtes à copier, puis Conforme, Non conforme ou Bloqué. Une capture se colle ou se glisse comme preuve.
- Ce qu'a fait l'agent : chaque écriture de l'agent avec sa raison, et ce qu'une personne doit revoir.
- Cas de test, avec une recherche et l'approbation de plusieurs cas d'un coup ; anomalies, actions, couverture, avancement, et l'historique lisible en phrases.
- Export vers Excel : une feuille par sujet (cas, résultats, pas, anomalies, actions, exigences), à trier, filtrer et partager.
- Utilisable au clavier seul, en mode clair ou sombre, en français ou en anglais. Un test commencé se reprend après un rafraîchissement.




L'accueil : ce qui vous attend maintenant, puis ce qui attend les autres.

Un test pas à pas : les valeurs à saisir, prêtes à copier, puis le verdict.

## Les outils exposés à l'assistant

### Accueil, profil et méthode

| Outil | Rôle |
|---|---|
| `onboarding_status` | Où en est l'accueil : profil, projet, connexion Azure DevOps, parcours, guide. |
| `onboarding_ado_connection` | Relie l'accueil au serveur MCP Azure DevOps ; ne modifie la configuration qu'avec l'accord de la personne. |
| `onboarding_save_profile` | Enregistre le nom, le rôle et la langue de la personne dans son profil local. |
| `tracking_settings` | Enregistre les réglages du projet : langue, parcours, responsable des décisions, Azure DevOps. |
| `get_guide` | Donne le guide d'utilisation, en français ou en anglais, limité au rôle de la personne. |
| `list_skills` | Liste les huit compétences et leur usage. |
| `get_skill` | Donne la procédure d'une compétence et la liste de ses références. |
| `read_skill_file` | Lit une référence, un modèle, un schéma ou un exemple d'une compétence. |
| `read_methodology` | Lit une note de la méthode. |
| `update_check` | Dit si une version plus récente existe, ce qu'elle change, garde et remplace, vérifié tous les 7 jours sans déranger : l'assistant lit un fichier public sur moamen.fr, rien n'est envoyé, le Testing Agent ne se connecte jamais ; installation seulement avec l'accord de la personne. |

### Registre et suivi

| Outil | Rôle |
|---|---|
| `tracking_init` | Crée le registre de suivi (dossier testing/) dans un projet. |
| `tracking_upsert` | Crée ou met à jour un élément du suivi : périmètre, exigence, cas de test, campagne, anomalie, action. |
| `tracking_status` | État d'une campagne : couverture, avancement, ce qui reste, ce qui attend une personne. |
| `tracking_next_step` | Donne la seule action suivante pour une user story ou la campagne, ou la question à poser à une personne nommée. |
| `tracking_render` | Régénère l'état des tests (Markdown) et le tableau de bord hors ligne (HTML). |
| `tracking_open_dashboard` | Lance le tableau de bord local (127.0.0.1 uniquement) et donne son lien. |

### User story, contexte et critères d'acceptation

| Outil | Rôle |
|---|---|
| `tracking_story_sync` | Enregistre une user story lue dans Azure DevOps avec ses parents, liens et commentaires, découpe ses critères d'acceptation et repère ce qui a changé. |
| `tracking_story_context` | Enregistre la lecture du contexte par l'agent (contraintes, hors périmètre, incohérences, exigences implicites), à valider par une personne. |
| `tracking_story` | Montre la fiche d'une user story ou son dossier de clôture ; enregistre la validation d'une personne. |

### Tests de code et tests d'écran

| Outil | Rôle |
|---|---|
| `summarize_test_results` | Résume des rapports JUnit XML par niveau : composant, intégration, système. |
| `tracking_import` | Importe des résultats automatisés : rapport JUnit XML ou rapport d'évaluation. |
| `tracking_case_brief` | Prépare l'exécution d'un cas : pas avec les valeurs du jeu de données, résultats attendus, historique. |
| `tracking_record_result` | Enregistre une exécution d'un cas approuvé, pas à pas, au nom de la personne qui l'a passée. |

### Évaluation d'IA

| Outil | Rôle |
|---|---|
| `validate_dataset` | Valide un jeu de référence (JSONL) : schéma, doublons, couverture des tranches. |
| `check_readiness` | Vérifie qu'un manifeste d'évaluation est prêt, sans l'exécuter. |
| `run_evaluation` | Exécute un manifeste d'évaluation approuvé : cas, évaluateurs, vétos, tranches, agrégats. |

### Azure DevOps

| Outil | Rôle |
|---|---|
| `ado_diagnose` | Explique un refus d'Azure DevOps (droits, licence Test Plans, outil non visible) et ce que la personne peut faire. |
| `tracking_ado_checklist` | Génère la checklist de tests d'une user story pour Azure DevOps, critère d'acceptation par critère ; seul un OK est coché. |
| `tracking_ado_testplan` | Prépare la publication d'une campagne dans Azure Test Plans : plan, suites, cas. |
| `tracking_ado_suite` | Crée la suite liée à l'exigence d'une user story par l'API REST, avec la connexion Azure CLI de la personne, après son accord. |

| Outil | Rôle |
|---|---|
| `tracking_import_ado_plan` | Charge dans le registre un plan publié dans Azure Test Plans. |
| `tracking_import_ado_build` | Enregistre les résultats automatisés d'un build, lus dans Azure DevOps. |
| `tracking_import_ado_results` | Compare les résultats lus dans Test Plans au registre : enregistre, confirme ou signale un écart, sans rien écraser. |
| `tracking_publish_results` | Liste les résultats à poser dans Test Plans, chacun avec sa personne ; publiés seulement à sa demande. |
| `tracking_mark_published` | Enregistre qu'un résultat a été posé dans Test Plans à la demande de la personne qui a testé. |
| `tracking_validation_note` | Prépare la note de preuve à coller dans Test Plans et le résultat à y poser. |
| `tracking_declare_ado_validation` | Enregistre, au nom de la personne, qu'elle a posé un résultat dans Test Plans. |
| `tracking_bug_draft` | Rédige le brouillon d'un bug Azure DevOps à partir d'un KO, à relire avant création. |

# 4. Données et fichiers

| Emplacement | Contenu |
|---|---|
| **%LOCALAPPDATA%\TestingAgent\<version> (Windows) ou ~/.testing-agent/app/<version>** | Une copie du Testing Agent, avec un environnement Python propre. |
| **testing/data/ dans le projet** | La source de vérité du suivi : un fichier JSONL par type d'élément, un fichier de projet et le journal des écritures. |
| **testing/ÉTAT DES TESTS.md et testing/tableau-de-bord.html** | Des vues générées, régénérables à tout moment. |
| **testing/preuves/ dans le projet** | Les captures et journaux joints aux résultats : images, PDF ou texte, vérifiés par leur contenu, 3 Mo au plus chacun. Versionnés avec le suivi. |
| **.vscode/mcp.json, .mcp.json, .cursor/mcp.json ou ~/.codex/config.toml** | L'entrée testing-agent de la configuration de l'assistant choisi. Les autres serveurs ne sont pas modifiés. |
| **~/.testing-agent** | Le profil : nom, rôle, langue. |

### Le journal des écritures

Chaque écriture est consignée dans testing/data/journal.jsonl : qui (nom et nature, humain ou agent), quand, quel élément, et chaque champ avec sa valeur avant et après. Ce sont des fichiers texte : ils se versionnent avec le code, se comparent et se fusionnent comme lui. Quand deux personnes enregistrent en même temps, chacune dans sa copie, le Testing Agent règle la fusion à l'ouverture : rien n'est perdu, et un élément renuméroté l'est dans le journal.

### Réseau

- Le Testing Agent n'ouvre aucune connexion vers l'extérieur et n'envoie rien à son auteur.
- L'interface web n'écoute que sur 127.0.0.1, refuse les requêtes dont l'hôte ou l'origine n'est pas locale, et ne peut pas être affiché dans un autre site.
- L'installation télécharge deux dépendances Python (PyYAML et jsonschema) depuis le dépôt public PyPI, dans un environnement propre au Testing Agent.
- Le Testing Agent ne demande ni ne stocke de clé ni de jeton. Azure DevOps est appelé par le serveur MCP Azure DevOps de l'utilisateur, avec ses propres droits.

### Données personnelles

Le suivi peut contenir des noms et des observations saisis par les utilisateurs. Il reste dans le dépôt du projet, soumis aux règles d'accès de ce dépôt.

## 5. Règles et décisions humaines

### Ce que le registre refuse

Appliqué par le code : une écriture qui ne respecte pas la règle est rejetée.

- Clore une anomalie sans nouveau test réussi.
- Approuver un cas dont le résultat attendu reste « à confirmer ».
- Attribuer une action à l'agent plutôt qu'à une personne nommée.
- Prendre un Go / No Go sur une campagne non terminée.
- Enregistrer une écriture sans nom d'auteur.

### Les consignes données à l'assistant

Suivies par l'assistant, non imposées par le code.

- Ne pas inventer de seuil ni d'importance.
- Ne pas poser un résultat dans Azure Test Plans à la place de la personne.
- Ne créer un bug dans Azure DevOps qu'avec l'accord de la personne.
- Ne marquer réussi un test manuel que si une personne dit l'avoir passé.

### Qui décide

| Décision | Qui |
|---|---|
| **Approuver un cas de test** | Le responsable des décisions de test. Un cas approuvé par la personne qui l'exécute est signalé pour une relecture par un pair. |
| **Déclarer qu'un test manuel a été passé** | La personne qui l'a passé |
| **Poser un résultat dans Azure Test Plans** | La personne qui a exécuté le test, avec son compte |
| **Accepter un KO** | Le responsable, en indiquant l'écart accepté et la raison |
| **Valider la gravité métier d'une anomalie** | Une personne côté métier |
| **Décider d'un Go / No Go** | Le responsable nommé |

### Limite : l'identité

Le serveur ne vérifie pas l'identité de la personne. Une décision est enregistrée au nom de la personne que l'assistant déclare, lorsqu'elle l'a prise. Pour un contrôle d'identité, il faut s'appuyer sur le dépôt (historique, droits d'accès) et sur Azure DevOps, où le résultat est posé par la personne elle-même, sous son compte. Un assistant peut aussi s'écarter d'une consigne : le journal garde la trace de ce qui a été écrit, par qui, avec l'état avant et après.

## 6. Azure DevOps

Azure DevOps est la référence partagée de l'équipe. Le Testing Agent ne l'appelle pas lui-même : l'assistant le fait avec le serveur MCP Azure DevOps officiel, celui de l'utilisateur.

### Ce qu'il lit

- Les projets et les itérations : le sprint en cours est proposé, la personne confirme.
- Les user stories et leurs critères d'acceptation.
- Les plans de test existants, pour rejoindre un plan au lieu d'en créer un.
- Les résultats automatisés d'un build.

### Ce qu'il prépare, et n'écrit qu'après l'accord de la personne

- Un plan de test par sprint, une suite par user story, un cas de test par jeu de données, avec la phase dans le titre et un lien vers la story.
- Une section « Suivi des tests » dans la user story, régénérée après chaque résultat.
- Un bug à partir d'un KO : étapes de reproduction jusqu'au pas en échec, attendu, observé, gravité proposée.

### Ce qu'il ne fait pas

- Supprimer quoi que ce soit dans Azure DevOps.

- Poser un résultat dans Azure Test Plans : la personne le fait, avec la note de preuve que l'agent lui prépare.

Ces accords font partie des consignes données à l'assistant (section 5). Créer des plans dans Azure Test Plans demande une licence Basic + Test Plans. Avec Basic seul, on exécute des tests sans en créer. Sans Azure DevOps, les user stories sont données en texte et tout le reste fonctionne en local.

# 7. Installation et désinstallation

### Avec votre assistant (recommandé)

Copier la phrase de la page du Testing Agent, moamen.fr/fr/testing-agent/, puis la coller dans l'assistant, en mode Agent, dans le projet à équiper. L'assistant télécharge moamen.fr/downloads/testing-agent.zip, vérifie son empreinte SHA-256, la décompresse dans un dossier temporaire et suit le fichier « Pour votre assistant » de l'archive : il vérifie Python 3.10 ou plus et propose de l'installer s'il manque, lance l'installation, puis indique comment activer le Testing Agent dans l'outil. Il annonce chaque commande, que la personne valide.

### Si l'assistant n'y arrive pas

- Télécharger l'archive : moamen.fr/downloads/testing-agent.zip.
- La décompresser : clic droit puis « Extraire tout » sous Windows, double-clic sur Mac.
- Ouvrir un terminal dans le dossier « Testing Agent/Fichiers - Files » et lancer py -3 "mcp-server\wizard.py" (Windows) ou python3 "mcp-server/wizard.py" (macOS, Linux). Python 3.10 ou plus est nécessaire.
- Répondre à deux questions : l'assistant utilisé, puis le projet. L'installation se termine par les étapes pour commencer.

L'archive ne contient aucun script de lancement (.cmd, .command) : l'installation passe par l'assistant ou par Python. L'installateur copie le Testing Agent dans un dossier permanent, crée un environnement Python, installe deux dépendances, configure l'assistant choisi pour le projet, vérifie que le serveur répond et affiche les étapes pour commencer. Si le fichier de configuration contient des commentaires, il n'est pas réécrit : l'entrée à ajouter est affichée.

### Vérifier le fichier, installer sans fenêtre

L'empreinte SHA-256 publiée sur la page se compare avec Get-FileHash testing-agent.zip -Algorithm SHA256 (Windows) ou shasum -a 256 testing-agent.zip (macOS, Linux).

En ligne de commande : py -3.12 "<dossier>\Fichiers - Files\mcp-server\install.py" --client vscode --project "<projet>" (clients : vscode, claude, cursor, codex, other).

### Mettre à jour

Redonner la phrase de la page à l'assistant, ou lancer wizard.py de la nouvelle version. Les registres de suivi et le profil ne bougent pas.

### Désinstaller

Retirer l'entrée testing-agent de la configuration de l'assistant et supprimer le dossier permanent du Testing Agent. Les registres de suivi restent dans les projets ; le profil se supprime avec ~/.testing-agent.

# 8. Limites et usage conseillé

- Conçu pour l'essai et l'usage en équipe. Faites-le valider par votre service informatique avant un déploiement.
- L'identité d'une personne n'est pas vérifiée (section 5).
- Les consignes données à l'assistant dépendent de lui : le comportement varie avec le modèle utilisé.
- D'autres clients MCP peuvent être configurés à la main ; ce n'est pas vérifié.
- Azure DevOps est le seul outil de suivi pris en charge. Jira et Monday ne le sont pas.
- Les fichiers ne sont pas signés.
- Le dépôt du code n'est pas public. Gratuit, sans revente ni redistribution.
- Un seul auteur, sans support garanti ni calendrier de mises à jour.
- La méthode est celle de l'auteur ; elle ne se réclame d'aucune norme.

# 9. Historique des versions

| Version | Date | Changement |
|---|---|---|
| **0.8.3** | 2026-10-28 | Inspection HTTPS des entreprises (Netskope, Zscaler) : publication dans Test Plans rétablie avec Python 3.13 et plus, certificats vérifiés comme le fait Windows. |
| **0.8.2** | 2026-10-28 | Phase de test : gratuit pendant toute la phase, accès depuis l'espace sur moamen.fr ; s'il devient payant, un préavis de 30 jours, et personne ne paie sans l'avoir accepté. |
| **0.8.1** | 2026-10-23 | Mises à jour : vérification silencieuse tous les 7 jours ; l'agent dit si une version plus récente existe ; l'installateur garde les données, en fait une copie et conserve l'ancienne version jusqu'à confirmation. |
| **0.8.0** | 2026-10-23 | La user story lue en entier (description, critères d'acceptation, parents, commentaires) ; suivi critère par critère, changements repérés ; suites de besoins dans Test Plans ; résultats publiés et relus dans Azure DevOps ; prochaine étape guidée ; refus expliqués ; tableaux triables et filtrables. |
| **0.7.7** | 2026-10-21 | Installation par l'assistant uniquement : les scripts de double-clic, signalés à tort par un antivirus, sont retirés. |
| **0.7.6** | 2026-10-20 | Recherche dans les cas, approbation de plusieurs cas d'un coup, export du suivi vers Excel. |
| **0.7.5** | 2026-10-20 | Interface web repensée : accueil par personne, exécution pas à pas avec valeurs à copier et captures jointes, ce qu'a fait l'agent et pourquoi ; travail en équipe par git ; accessibilité. |
| **0.7.4** | 2026-10-20 | Installation par l'assistant, à partir d'une phrase à copier ; archive rangée en quatre éléments, au nom lisible. |
| **0.7.3** | 2026-10-20 | Mentions de droits dans la méthode et les compétences ; conditions d'utilisation étendues à la méthode. |
| **0.7.2** | 2026-10-20 | Archive recentrée sur l'application de la méthode : 21 notes. |
| **0.7.1** | 2026-10-17 | Installation en un double-clic, sous Windows et sur Mac, avec une copie permanente du Testing Agent. |
| **0.7.0** | 2026-10-17 | Français et anglais : accueil, guide, tableau de bord, état des tests, checklist, note de preuve, brouillon de bug. |
| **0.6.1** | 2026-10-17 | Sans Azure DevOps, l'accueil ne pose plus de question sur Azure DevOps ni sur la licence Test Plans. |
| **0.6.0** | 2026-10-17 | Gratuit. Installation pour tout assistant MCP en mode agent. Parcours sans outil de suivi. |
| **0.5.0** | 2026-10-16 | Aligné sur le serveur MCP Azure DevOps officiel ; lecture des résultats d'un build. |
| **0.4.0** | 2026-10-16 | Accueil connecté à Azure DevOps : les projets et le sprint sont lus au lieu d'être saisis. |
| **0.3.0** | 2026-10-16 | Phases par user story, jeux de données, arbitrage, checklist Azure DevOps. |

# Annexe. Les huit compétences

| Compétence | Rôle |
|---|---|
| **audit-ai-evaluation** | Audite une évaluation : est-elle gouvernée correctement ? |
| **build-golden-dataset** | Construit le jeu de référence annoté. |
| **design-ai-evaluation** | Conçoit l'évaluation d'une fonction IA : critères, jeu de référence, seuils. |
| **design-software-tests** | Conçoit les tests de composant, d'intégration et de système. |
| **prepare-ai-testing-session** | Transforme une fonction, une user story, des exigences et des risques en dossier de session de test. |
| **run-product-evaluation** | Exécute une évaluation approuvée sur le produit. |
| **select-ai-evaluators** | Choisit les évaluateurs et les métriques. |
| **track-test-campaign** | Suit une campagne : registre, tableau de bord, décisions. |