LIVRE BLANC

# Agents IA .NET utiles et gouvernés

## Les 7 décisions à prendre avant un POC avec Microsoft Agent Framework

Comment passer d'une démonstration agentique convaincante à un premier projet réellement cadré, utile et maîtrisable

Avec :

- une grille pour évaluer la maturité d'un cas d'usage ;
- les 7 erreurs classiques avant un premier POC agentique ;
- une ressource complémentaire : les 5 exemples de la vidéo à télécharger.

Olivier Dahan

Architecte .NET – Microsoft MVP

Contact : odahan@e-naxos.com

Site : www.e-naxos.com

Blog : www.e-naxos.com/blog

La vidéo « Initiation à la programmation Agentique en C# »
https://youtu.be/0pdswhV3_Uo

Le code source des exemples est téléchargeables par le lien suivant :
https://www.e-naxos.com/ex-site

## Table des matières

## Avant-propos

Les agents IA suscitent beaucoup d'intérêt, mais aussi beaucoup de confusion.

Dans bien des cas, on parle d'"agent" alors qu'on désigne en réalité un assistant conversationnel un peu plus avancé. Or ce n'est pas la même chose. Un agent utile n'est pas seulement un composant capable de produire une bonne réponse. C'est une brique logicielle qui sait utiliser des outils, s'appuyer sur un état réel du système, proposer ou déclencher une action dans un cadre contrôlé, puis retourner un résultat exploitable par une application.

> *« Un agent utile n'est pas un chatbot qui parle plus ou mieux. C'est une brique logicielle capable d'utiliser des outils, d'agir sous contrôle, et de retourner un résultat exploitable. »*

C'est précisément la logique de la vidéo d'initiation consacrée à Microsoft Agent Framework (lien vidéo en 1ère page) : partir d'un échange simple avec un modèle, introduire des outils C#, distinguer observation et action, conserver une validation humaine sur les opérations sensibles, puis montrer qu'une sortie structurée est souvent plus utile qu'une réponse simplement bien rédigée. L'initiation assume d'ailleurs clairement son périmètre : poser des bases solides sans prétendre couvrir à elle seule les enjeux complets d'industrialisation, comme le RAG, l'observabilité, le multi-agents ou le cadrage production.

***Ce livre blanc prolonge cette logique***. Il ne remplace ni la vidéo, ni les exemples. Son rôle est différent : **aider** une équipe, un architecte, un éditeur logiciel ou un responsable technique **à décider dans quel cadre un premier POC agentique a du sens, et comment éviter les erreurs de départ**.

Autrement dit, ce document ne répond pas à la question : « *Comment faire une démo qui marche ?* » Il répond à une autre question, plus utile : *« Comment savoir si cette démo mérite de devenir un projet ? »*

> On notera que souvent, la vidéo d'initiation est citée car elle illustre des concepts importants, mais on peut parfaitement lire le présent Livre Blanc sans avoir vu la vidéo si les détails d'implémentation sortent du champ des préoccupations du lecteur.

## À qui s'adresse ce document ?

- aux équipes .NET qui veulent comprendre ce qu'un agent peut réellement apporter à une application ou à un processus existant ;
- aux architectes et lead developers qui doivent cadrer un premier projet sans s'enfermer trop tôt dans de mauvais choix ;
- aux éditeurs qui veulent enrichir un logiciel sans ouvrir une zone de flou technique ou opérationnel ;
- aux décideurs techniques qui ont besoin d'une lecture claire des gains possibles, des limites réelles et des garde-fous nécessaires.

Il s'adresse aussi à celles et ceux qui ont déjà vu une démonstration convaincante et qui savent qu'entre une démo et une solution exploitable se glissent rapidement des sujets structurants : architecture, gouvernance, validation, sécurité, robustesse, observation. C'est d'ailleurs exactement la séparation assumée par la vidéo d'initiation, qui montre la mécanique sans prétendre raconter toute la suite.

## Ce que vous trouverez dans ce livre blanc

Nous allons examiner les **7 décisions qui conditionnent la qualité d'un premier POC agentique** :

- définir si l'on cherche un simple assistant ou un agent réellement utile ;
- choisir une stratégie d'architecture cohérente entre local, cloud et hybride ;
- clarifier le degré d'indépendance vis-à-vis du fournisseur IA ;
- déterminer quels outils l'agent peut réellement utiliser ;
- poser les règles de validation humaine ;
- cadrer la forme de la sortie attendue ;
- décider si l'on cherche une démonstration séduisante ou une trajectoire sérieuse vers la production.

Vous trouverez également **deux compléments** pratiques :

- une **grille d'évaluation** pour savoir si votre cas d'usage est mûr pour un POC ;
- une page sur **les 7 erreurs classiques à éviter** avant d'exposer un agent à une vraie logique métier.

Enfin, une ressource complémentaire propose au lecteur le téléchargement des 5 exemples de la vidéo. Ce point est cohérent avec la structure même de la vidéo d'initiation, qui déroule cinq étapes concrètes : réponse de base, outil passif, outil actif, validation humaine, puis sortie structurée exploitable par l'application.

## Décision n°1 — Voulez-vous un chatbot plus agréable, ou un agent réellement utile ?

C'est la première décision, et souvent la plus mal traitée.

Un assistant conversationnel peut déjà rendre service : reformuler, résumer, guider, suggérer. Mais un agent utile va plus loin. Il ne se contente pas de produire du texte ; il sait s'appuyer sur un contexte réel, consulter des informations externes, **appeler des outils**, et parfois **proposer** ou **exécuter une action** dans un **cadre défini**.

La différence est décisive, car elle conditionne tout le reste : architecture, gouvernance, validation, responsabilité et valeur métier.

La vidéo d'initiation repose justement sur cette progression. On ne s'arrête pas à une réponse générée par le modèle ; on montre comment l'agent peut lire une information utile, préparer une action, attendre une validation, puis produire une sortie structurée directement exploitable par le logiciel.

La clarification de départ est donc simple : cherchez-vous une meilleure interface de dialogue, ou une capacité logicielle nouvelle dans votre application ? Cette question paraît élémentaire. En réalité, elle évite beaucoup de POC mal engagés.

### Ce qu'il faut décider

- l'utilité exacte attendue ;
- le type d'assistance ou d'action visé ;
- le niveau d'autonomie acceptable ;
- la valeur métier réellement recherchée.

## Décision n°2 — Votre cible est-elle locale, cloud, ou hybride ?

Le débat entre local et cloud est souvent mal formulé. Il ne s'agit pas de choisir un camp. Il s'agit de choisir une architecture adaptée au besoin.

Une approche local-first peut avoir beaucoup de sens : démonstration, postes légers pour prototypage, confidentialité, maîtrise de certaines données, arbitrage sur les coûts, dépendance réduite au cloud, expérimentation plus libre. La vidéo d'initiation met volontairement en avant ce scénario, tout en conservant l'idée qu'une évolution vers le cloud reste possible sans refaire tout le code métier, c'est l'un des avantages que propose Microsoft Agent Framework, en plus de pouvoir s'insérer dans SI plus vaste pour peu qu'il soit écrit sous Dotnet.

Mais le local n'est pas une réponse universelle. La latence, la taille du modèle, la qualité attendue, le matériel disponible, le volume et les contraintes d'exploitation changent complètement la décision. Le local est très utile dans certaines situations listées plus haut, mais le choix de production dépend du niveau de service attendu.

> *« Le local-first n'est pas une posture idéologique. C'est un choix d'architecture qui peut améliorer la maîtrise des données, des coûts et de la dépendance au cloud. »*

### Ce qu'il faut décider

- les contraintes sur les données ;
- les attentes de performance ;
- l'objectif réel du POC ;
- la possibilité, ou non, d'une architecture évolutive entre local, cloud et hybride.

## Décision n°3 — Quel niveau de dépendance acceptez-vous vis-à-vis du fournisseur IA ?

Beaucoup de prototypes s'écrivent trop vite autour d'un fournisseur précis. Cela permet d'aller vite au début, mais cela coûte ensuite en souplesse, en lisibilité et en maintenabilité.

L'intérêt du socle présenté dans la vidéo d'initiation est de conserver une séparation nette entre la logique applicative, la logique agentique et l'accès au modèle. On ne code plus "pour un modèle", *mais pour une interface*, avec **la possibilité de *changer de fournisseur sans bouleverser la logique métier***.

Ce point n'a pas besoin d'être surchargé ici par des détails de plomberie logicielle. Ce qui compte dans le cadre du livre blanc, c'est la conséquence : un POC bien posé doit éviter d'enfermer trop tôt l'équipe dans un choix technique qui n'a pas encore été pleinement justifié. Et la liberté dans le choix des LLM utilisés et leur localisation offre une tranquillité essentielle sur ce point.

### Ce qu'il faut décider

- ce qui relève du besoin métier ;
- ce qui relève de la logique agentique ;
- ce qui relève du fournisseur choisi à un instant donné ;
- le niveau de réversibilité souhaité.

## Décision n°4 — Quels outils l'agent peut-il utiliser ?

C'est ici que l'agent devient réellement intéressant.

Tant qu'un modèle répond uniquement à partir d'un prompt, on reste dans une logique conversationnelle. La situation change lorsqu'on lui permet de **s'appuyer sur des outils exposés par l'application**.

La vidéo d'initiation distingue très justement deux catégories : les outils passifs, qui lisent une information sans modifier l'état du système ; et les outils actifs, qui déclenchent une opération et produisent un effet de bord.

Cette distinction est fondamentale. Un outil passif permet, par exemple, à l'agent d'obtenir une donnée réelle plutôt que d'improviser une réponse. Un outil actif lui permet de faire progresser un processus métier : créer un ticket, déclencher une opération, enrichir un dossier, alimenter un workflow, piloter des IoT...

> *Les agents donnent des yeux et des oreilles à un LLM pour lire l'environnement et des mains pour agir sur celui-ci. On passe d'un Chatbot - sympathique « Eliza » des 60's amélioré - à un des acteurs agissant dans le monde réel.*

Mais toute la gouvernance commence précisément là. Plus un outil agit, plus son exposition doit être pensée avec rigueur. La vraie question n'est pas de savoir si l'agent sait bien parler, mais s'il agit dans le cadre qui lui est imposé.

> *« Un outil passif observe. Un outil actif modifie l'état du système. Toute la gouvernance commence là. »*

### Ce qu'il faut décider

- son rôle exact ;
- sa nature passive ou active ;
- son périmètre ;
- les données ou systèmes concernés ;
- les conditions dans lesquelles il peut être appelé.

## Décision n°5 — Où placez-vous la validation humaine ?

Dès qu'un agent peut produire un effet de bord, la validation humaine cesse d'être un détail. Elle devient un élément central du dispositif.

La vidéo d'initiation montre très bien cette logique avec une validation avant création effective d'un ticket. C'est un bon exemple, parce qu'il illustre exactement ce que doit devenir un agent dans un contexte professionnel : non pas un acteur autonome hors de contrôle, mais un composant capable de préparer, proposer et exécuter dans un cadre gouverné.

C'est là que beaucoup d'organisations se trompent. Elles pensent automatiser plus intelligemment. En réalité, si elles ne cadrent pas cette étape, elles automatisent surtout le risque.

La validation humaine ainsi que les traces, la télémétrie et d'autres mesures, ne servent pas à rassurer artificiellement. Elles **servent à déterminer où se situe la responsabilité**, comment une action est autorisée, et dans quelles conditions elle peut être **refusée**, **auditée** ou rejouée.

> *« On peut automatiser vite, mais pas au prix de la perte de contrôle. »*

### Ce qu'il faut décider

- quelles actions sont librement exécutables ;
- lesquelles exigent une validation ;
- lesquelles restent interdites ;

- qui valide ;
- comment la trace de cette décision est conservée.

## Décision n°6 — Quel type de sortie attendez-vous réellement ?

*Un agent ne doit pas seulement bien parler. Il doit aussi renvoyer un résultat exploitable.*

La vidéo d'initiation a raison d'y consacrer une séquence spécifique. Une sortie utile n'est pas toujours un texte agréable à lire. Dans de nombreux cas, la vraie valeur est dans une **structure de données exploitable par le reste du système** : une information validable, un objet métier, un brouillon prêt à être transformé, une action proposée avec ses paramètres, un contrat de sortie cohérent.

**Cette logique change profondément la manière de penser le projet**. Dès lors, la question ne devient plus seulement : "*Le modèle répond-il correctement ?*" Elle devient aussi : "*Le système peut-il consommer, contrôler et exploiter ce que l'agent produit ?*"

> *« Un agent ne doit pas seulement bien parler. Il doit aussi savoir produire une sortie que le reste du système peut exploiter proprement. Finalement, le but est de forcer un LLM non déterministe dans un processus industriel qui lui doit l'être tout en 'vampirisant' ce qu'il fait le mieux (comprendre le langage naturel, le traité, raisonner…). »*

### Ce qu'il faut décider

- ce qui relève du texte libre ;
- ce qui doit être structuré ;
- quels champs sont attendus ;
- quel niveau de validation est nécessaire ;
- comment l'application réagit en cas de sortie incomplète ou invalide.

## Décision n°7 — Cherchez-vous une démo impressionnante, ou une trajectoire crédible ?

C'est souvent la décision la plus négligée.

Une démonstration peut être réussie sans répondre à toutes les questions d'un système de production. C'est normal. Le problème n'est pas là. Le problème commence lorsqu'une organisation prend une démo d'introduction pour une préfiguration implicite de sa cible finale.

La vidéo d'initiation assume clairement son périmètre : elle pose les bases, montre la mécanique, et n'entre pas dans toute l'industrialisation nécessaire ensuite — sécurité, observabilité, données métier, RAG, robustesse, stratégies avancées d'intégration. Cette

honnêteté en fait, en partie, la force. Confondre initiation rapide et POC serait désastreux.

Un bon POC n'a pas vocation à tout résoudre. Il doit en revanche **permettre de décider** : si le cas d'usage mérite une suite ; si l'approche retenue est **viable** ; quels **garde-fous** sont indispensables ; et quels sujets doivent être traités avant toute mise en exploitation.

### Ce qu'il faut décider

- qu'essaie-t-on de prouver ;
- sur quel périmètre ;
- avec quels critères de succès ;
- pour décider quoi ensuite.

## Votre cas d'usage est-il mûr pour un POC agentique ?

Cette grille n'a pas vocation à remplacer un atelier de cadrage. En revanche, elle permet de distinguer assez vite un sujet encore flou d'un sujet déjà exploitable.

Pour chaque point, attribuez : 0 point si le sujet n'est pas clair ou pas traité ; 1 point s'il est partiellement défini ; 2 points s'il est clairement défini.

*1. Le besoin métier est-il précis ?*
Le problème à résoudre doit être concret, compréhensible et suffisamment borné. Un besoin formulé de manière trop vague conduit presque toujours à une démo sans véritable valeur décisionnelle.

*2. L'utilité de l'agent va-t-elle au-delà d'un simple chatbot ?*
Si l'objectif est uniquement de mieux dialoguer, un agent n'est pas forcément nécessaire. Le sujet devient plus pertinent dès qu'il faut consulter un état, appeler des outils ou proposer une action.

*3. Le périmètre fonctionnel est-il borné ?*
Un bon POC ne cherche pas à tout faire. Il se concentre sur un cas d'usage circonscrit, avec une promesse claire et un niveau de risque maîtrisable.

*4. Les outils nécessaires sont-ils identifiés ?*
Il faut savoir quelles fonctions, quelles données ou quels systèmes l'agent devra mobiliser. Tant que cela reste abstrait, le POC repose davantage sur des suppositions que sur un cadre réel.

*5. Les effets de bord sont-ils connus et maîtrisés ?*
Dès qu'un agent peut créer, modifier, envoyer ou déclencher quelque chose, il faut savoir exactement ce qui peut être impacté et dans quelles limites.

*6. Le niveau de validation humaine est-il défini ?*
Le projet doit préciser où l'humain garde la main. Sans ce point, la capacité d'action devient vite un problème de gouvernance plus qu'un gain métier.

*7. Le format de sortie attendu est-il clair ?*
Texte libre, JSON, objet métier, proposition d'action, brouillon exploitable : cette question doit être tranchée tôt, car elle conditionne l'intégration réelle dans l'application.

*8. Les contraintes de données sont-elles connues ?*
Confidentialité, localisation, exposition au cloud, journalisation, conformité, traçabilité : ces éléments changent souvent plus le projet que le choix du modèle lui-même.

*9. Le mode d'hébergement visé est-il cohérent ?*
Local, cloud ou hybride : l'important n'est pas la préférence de départ, mais l'adéquation entre le contexte, les contraintes et le niveau de service attendu.

*10. Les critères de succès du POC sont-ils formulés ?*
Un POC doit permettre de décider. Il faut donc savoir à l'avance ce qui permettra de statuer que le sujet est concluant, partiellement concluant ou non concluant.

Lecture du score

**0 à 6 :** le sujet est encore trop flou. Avant un POC, **il faut d'abord clarifier le besoin et le périmètre**.

**7 à 10 :** le cas est intéressant, mais encore insuffisamment cadré. **Un atelier de cadrage est recommandé** avant toute preuve de concept.

**11 à 14 :** le sujet est déjà **suffisamment structuré pour envisager un workshop POC** ou un cadrage court débouchant rapidement sur un prototype ciblé.

**15 à 20 :** le cas est **clairement mûr pour un POC** agentique resserré, avec des objectifs explicites et une trajectoire de décision réaliste.

Ce barème est volontairement pragmatique. L'objectif n'est pas de disqualifier trop tôt, mais d'orienter vers la bonne étape : clarification, cadrage, workshop POC ou preuve de concept ciblée.

## Les 7 erreurs classiques avant un premier POC agentique

*1. Confondre chatbot et agent*
On croit viser l'action, alors qu'on ne définit qu'un dialogue plus fluide ou plus efficace (cas du RAG ou de recherche documentaire).

*2. Exposer un outil actif trop tôt*
Donner un pouvoir d'action avant d'avoir défini le cadre et la télémétrie est une faute de conception.

*3. Oublier la forme de la sortie*
Une bonne réponse n'est pas forcément une sortie utile au système si elle ne se présente pas dans un format non ambigu exploitable par du code classique.

*4. Sous-estimer la gouvernance*
Validation, permissions, journalisation, audit : tout cela ne peut pas être laissé pour plus tard.

*5. Mélanger les données et les instructions*
C'est une voie classique vers des comportements erratiques, voire vers des formes d'injection de prompt.

*6. Choisir trop tôt le fournisseur ou le modèle*

Le besoin, le périmètre et les contraintes doivent venir d'abord.

*7. Lancer un POC sans critère de décision*

Si vous ne savez pas ce que le POC doit permettre d'arbitrer, il ne vous aidera pas vraiment.

## Ressource complémentaire : les 5 exemples de la vidéo

Pour celles et ceux qui souhaitent prolonger la lecture par la pratique, les 5 exemples de la vidéo sont proposés au téléchargement en début de document.

- la conversation de base ;
- l'usage d'un outil passif ;
- l'usage d'un outil actif ;
- la validation humaine avant action ;
- la production d'une sortie structurée.

C'est le bon niveau de complément. Le livre blanc cadre le sujet. Les exemples, eux, permettent de manipuler le socle technique. Cette articulation est cohérente avec la manière dont la vidéo a été structurée autour de cinq démonstrations distinctes.

Ressource complémentaire

Vous souhaitez revoir à votre rythme les mécanismes montrés dans la vidéo d'initiation ? Les 5 exemples de la vidéo sont disponibles au téléchargement pour prolonger la démonstration et manipuler concrètement le socle présenté. Les formations courtes peuvent aussi vous aider à mieux formuler vos besoins.

## Trois manières d'aller plus loin

Formation 1 jour — **Comprendre et construire**

Pour poser des bases propres et opérationnelles sur les agents IA en environnement .NET.

Formation 2 jours — **Structurer et industrialiser**

Pour aller au-delà de la démonstration et commencer à traiter les vrais sujets d'architecture et de robustesse.

Workshop POC — **Valider sur votre contexte**

Pour partir d'un cas d'usage réel, avec de vraies données, de vraies contraintes et un objectif de décision clair.

Ces trois formats prolongent logiquement ce qui est présenté dans lz vidéo d'initiation gratuite : compréhension, structuration, validation sur le terrain.

## Conclusion

Les agents IA ne valent pas par leur capacité à produire des réponses impressionnantes. **Ils valent par leur capacité à s'insérer dans une architecture utile**, à appeler les bons outils, **à agir dans un cadre contrôlé**, et **à retourner un résultat réellement exploitable**.

C'est exactement à cet endroit que se joue la différence entre une démonstration séduisante et un projet défendable.

Un agent utile n'est pas un gadget. Mais un agent mal cadré devient vite un problème plus qu'un progrès.

Ce livre blanc n'a pas pour objet de promettre une industrialisation facile. Il vise autre chose : aider à poser les bonnes décisions au bon moment. C'est une ambition plus sobre, mais aussi plus sérieuse.

Et c'est probablement la seule qui vaille dans un sujet où l'enthousiasme technique peut très vite masquer des lacunes de cadre, de gouvernance ou d'utilité réelle.

## Prochaine étape

Si vous avez :

- une application existante ;
- un processus métier identifié ;
- un cas d'usage IA concret ;
- et le besoin de trancher proprement entre démonstration, POC et trajectoire d'industrialisation,

alors le bon point de départ n'est pas nécessairement de coder davantage. Le bon point de départ est souvent de cadrer le sujet avec méthode.

C'est précisément à cela que peuvent servir une formation ciblée ou un workshop POC, ou même une consultation de cadrage : non pas vous vendre une promesse vague, mais **vous aider à déterminer ce qui mérite réellement d'être démontré, validé, sécurisé et poursuivi.**

## (4ème de couverture)

**Un agent utile ne se résume pas à une bonne conversation.**

Il doit pouvoir utiliser des outils, agir dans un cadre maîtrisé, et retourner un résultat exploitable par votre application.

Ce livre blanc vous aide à poser les bonnes décisions avant un premier POC avec Microsoft Agent Framework : cadrer le cas d'usage, choisir le bon niveau d'autonomie, distinguer outils passifs et actifs, conserver la validation humaine sur les actions sensibles, préparer une sortie réellement intégrable, et éviter les erreurs classiques qui affaiblissent un projet dès son démarrage.

Vous y trouverez également :

- une grille pour évaluer la maturité d'un cas d'usage ;
- les 7 erreurs classiques avant un premier POC agentique ;
- une ressource complémentaire en fin de document pour télécharger les 5 exemples de la vidéo.

### Pour aller plus loin

www.e-naxos.com/IA

- Formation 1 jour — Comprendre et construire
- Formation 2 jours — Structurer et industrialiser
- Workshop POC — Valider sur votre contexte

Olivier Dahan

Architecte .NET -Microsoft MVP

odahan@e-naxos.com

www.e-naxos.com/IA