L'architecture event-driven comme complément architectural essentiel de l'IA agentique : de la cognition déléguée à l'action située, traçable et gouvernable

2 Avril 2026

Mots-clés : IA agentique, Architecture event-driven, Architecture d'entreprise, Gouvernance de l'IA, Jumeaux numériques, Systèmes multi-agents, AI Act, NIST AI RMF

Résumé

La littérature émergente sur l'IA agentique se concentre sur la planification, l'utilisation d'outils, l'orchestration et l'exécution semi-autonome, mais néglige largement le substrat architectural dans lequel les agents doivent opérer pour produire une valeur durable en environnement d'entreprise. Cet article soutient que l'architecture event-driven (EDA) constitue le complément architectural essentiel de l'IA agentique dans les environnements d'entreprise caractérisés par l'hétérogénéité des systèmes et le besoin d'action causalement reconstruisible sous contraintes de gouvernance. Sans cette couche, les agents restent des raisonneurs périphériques connectés à des API et des workflows ad hoc. Avec elle, ils deviennent des participants situés, ancrés temporellement, capables de percevoir des transitions d'état, de coopérer avec d'autres composants et de produire des traces causalement reconstruisibles. L'article formalise cinq fonctions structurantes que l'EDA apporte à l'IA agentique : ancrage temporel, délégation graduée, découplage inter-systèmes, auditabilité et écologie multi-agents distribuée. Il soutient que cette articulation fournit une base architecturale pour satisfaire les exigences de cadres de gouvernance actuels, notamment le NIST AI Risk Management Framework et le règlement européen sur l'intelligence artificielle. L'argumentation s'appuie sur une étude de cas issue de l'évolution architecturale d'une plateforme de jumeaux numériques cliniques en oncologie, où la transition d'un pipeline AutoML monolithique vers une architecture componentisée event-driven illustre les cinq fonctions identifiées. La conclusion est que l'IA agentique est fondamentalement un problème d'architecture d'entreprise.

1. Introduction

L'essor récent de l'IA agentique a déplacé le débat technologique de la génération de contenu vers la délégation partielle de tâches, de décisions et d'actions à des systèmes outillés. Les définitions convergent sur une idée simple : un agent est un système capable d'accomplir des tâches pour le compte d'un utilisateur ou d'un autre système, en mobilisant raisonnement, outils et séquences d'action (Sapkota et al., 2025 ; Wang et al.,

Jérôme Vetillard
CTO | VP R&D | Chief Product Officer | AI-Powered Healthcare & Life Sciences Products | Compliance by Design | PhD AgroParisTech | CPO MIT Sloan | Exec MBA IE Business School & Brown University
Twingital-institute / Twingital-ventures : twingital-ventures.com

2024). Les travaux de synthèse distinguent de plus en plus l'agent isolé, l'IA agentique comme paradigme plus large, et les configurations multi-agents ou hybrides (Sapkota et al., 2025 ; Xi et al., 2023). Les guides d'implémentation insistent, à juste titre, sur les outils, l'orchestration, les instructions, les garde-fous et les boucles d'exécution (OpenAI, 2025 ; Anthropic, 2025).

Pourtant, une dimension critique reste sous-théorisée : la forme du monde dans lequel l'agent agit. Ce n'est pas un point décoratif. Un agent peut être performant dans un modèle requête/réponse, appeler des outils, produire des réponses élaborées, exécuter des procédures bornées. Cela ne suffit pas à en faire un participant robuste de l'entreprise réelle. Une organisation n'est pas une succession de prompts. C'est un tissu de transitions d'état, de signaux, d'exceptions, de dépendances temporelles et d'escalades, précisément les phénomènes qu'une architecture event-driven est conçue pour traiter (Hohpe & Woolf, 2003 ; Michelson, 2006 ; Luckham, 2002).

La thèse de cet article est que l'architecture event-driven n'est pas un simple choix d'intégration parmi d'autres pour l'IA agentique. Dans une classe significative et croissante de cas d'usage d'entreprise, spécifiquement ceux caractérisés par l'hétérogénéité des systèmes et le besoin d'action causalement reconstruisible sous contraintes de gouvernance, elle en constitue le complément architectural essentiel. Sans elle, l'IA agentique reste une couche cognitive surplombante, capable mais désincarnée. Avec elle, elle acquiert un milieu opérationnel. Cette distinction importe parce qu'elle sépare les démonstrateurs séduisants des systèmes effectivement déployables.

L'article apporte trois contributions. Premièrement, il recadre l'IA agentique comme un problème d'architecture d'entreprise plutôt que comme un seul problème de conception d'agent. Deuxièmement, il propose un cadre à cinq fonctions et un spectre de maturité pour classifier les systèmes agentiques selon leur degré de situatedness architecturale. Troisièmement, il illustre la pertinence design de ce cadre à travers une étude de cas en IA clinique régulée, présentée comme un raisonnement de conception (*design rationale*) plutôt que comme une validation empirique.

La nature de cette contribution est celle d'un *article conceptuel orienté design*. Il n'offre ni preuve formelle, ni benchmarking empirique, ni comparaison contrôlée. Il propose un argument structuré, fondé sur la convergence de trois corpus (IA agentique, architecture event-driven, gouvernance de l'IA) et illustré par une étude de cas en phase de conception. Le lecteur doit l'évaluer en conséquence : comme une proposition pour structurer un champ émergent, non comme un rapport d'ingénierie achevée.

L'article est organisé comme suit. La section 2 passe en revue la littérature pertinente. La section 3 formalise le cadre des cinq fonctions structurantes. La section 4 articule le cœur conceptuel de l'argument, la double insuffisance de la cognition sans situation et

de la réaction sans interprétation. La section 5 présente l'étude de cas. La section 6 met en correspondance le cadre avec les exigences de gouvernance. La section 7 discute les limites. La section 8 conclut par une proposition doctrinale pour l'architecture d'entreprise.

2. État de l'art

2.1. IA agentique : taxonomie et état de l'art architectural

Le terme « IA agentique » s'est rapidement imposé pour désigner des systèmes qui dépassent la génération passive pour adopter des comportements orientés objectif, outillés et partiellement autonomes (Sapkota et al., 2025). Plusieurs efforts taxonomiques ont tenté de clarifier le paysage. Sapkota et al. (2025) distinguent les agents IA (systèmes individuels utilisant des outils) de l'IA agentique (un paradigme plus large englobant planification, adaptation et exécution multi-étapes). Wang et al. (2024) et Xi et al. (2023) proposent des synthèses détaillées des architectures d'agents fondés sur les LLM, identifiant des composants communs : modules de perception, mécanismes de planification, systèmes de mémoire, interfaces d'utilisation d'outils et couches d'exécution d'actions.

Les guides pratiques des grands laboratoires d'IA reflètent cette décomposition. Le guide d'OpenAI pour la construction d'agents (2025) met l'accent sur une progression allant de systèmes mono-agent outillés vers des configurations multi-agents orchestrées, avec une attention explicite portée aux garde-fous, à l'authentification et aux mécanismes humain-dans-la-boucle. Le guide d'Anthropic (2025) recommande de manière similaire de commencer par des patterns simples de LLM augmentés avant d'introduire des boucles agentiques et la délégation multi-agents.

Ce qui est notable dans cette littérature est l'absence relative de discussion sur le substrat architectural dans lequel les agents doivent opérer. Les architectures d'agents sont discutées principalement en termes de structure interne, raisonnement, planification, sélection d'outils, itération. L'environnement externe est typiquement abstrait comme un ensemble d'API ou d'outils. Tupe et al. (2025) constituent une exception partielle, notant que les architectures d'API traditionnelles sont mal équipées pour les comportements agentiques dynamiques et orientés objectif. Mais même ce travail se concentre sur l'adaptation des API plutôt que sur la question plus large du paradigme architectural.

Cette lacune mérite attention. La sophistication interne d'un agent est une condition nécessaire mais insuffisante de valeur opérationnelle. Ce qui est également nécessaire est une forme de couplage environnemental qui permette à l'agent de percevoir, de participer et d'être gouverné au sein de la dynamique du système d'entreprise.

2.2. Architecture event-driven : principes et patterns

L'architecture event-driven (EDA) est un style architectural bien établi dans lequel la production, la détection, la consommation et la réaction aux événements constituent le mécanisme principal de coordination entre composants du système (Michelson, 2006 ; Hohpe & Woolf, 2003). Un événement représente un changement d'état significatif ou la notification d'une condition, publié de manière asynchrone à des consommateurs découplés (Microsoft, 2026 ; AWS, s.d. ; Red Hat, s.d.).

Les travaux fondateurs de Hohpe et Woolf (2003) sur les patterns d'intégration d'entreprise ont codifié les patterns de messaging (publish-subscribe, routage de messages, routage par contenu, agrégation d'événements, files de lettres mortes) qui sous-tendent les implémentations modernes d'EDA. Luckham (2002) a étendu cette perspective avec le concept de traitement d'événements complexes (CEP), où des patterns à travers de multiples événements sont détectés et traités en quasi temps réel. Chandy et Schulte (2010) ont formalisé le traitement d'événements comme discipline de conception de systèmes d'entreprise agiles et réactifs, en distinguant le traitement d'événements simples, le traitement de flux et le traitement d'événements complexes. Richardson (2018) et Newman (2021) ont plus récemment situé l'EDA dans le paradigme des microservices, soutenant que la communication event-driven entre services fournit le découplage, la résilience et l'évolutivité requis par les systèmes distribués (voir aussi Dragoni et al., 2017).

Un concept apparenté mais distinct est l'event sourcing, formalisé notamment par Fowler (2005), dans lequel les changements d'état sont stockés comme une séquence immuable d'événements plutôt que comme un état courant mutable. L'event sourcing offre la capacité non seulement de connaître l'état courant d'un système, mais de reconstruire comment cet état a été atteint, une propriété d'une pertinence particulière pour l'auditabilité et la traçabilité.

Ces contributions établissent un gradient de sophistication dans le traitement d'événements qui est directement pertinent pour l'intégration de l'IA agentique. Au niveau le plus simple, le *traitement d'événements simples* gère des événements individuels par filtrage, routage et déclenchement, suffisant pour un agent qui réagit à des signaux discrets (par exemple, le franchissement d'un seuil déclenchant une alerte). Au niveau intermédiaire, le *traitement de flux* gère des flux continus d'événements avec fenêtrage, agrégation et transformation à état, pertinent pour des agents qui doivent détecter des tendances, calculer des statistiques glissantes ou maintenir une conscience situationnelle sur des fenêtres temporelles (Kleppmann, 2017). Au niveau le plus sophistiqué, le *traitement d'événements complexes* (CEP) détecte des patterns, corrélations et séquences causales à travers de multiples flux d'événements hétérogènes (Luckham, 2002 ; Chandy & Schulte, 2010) permettant à des agents d'identifier des situations composites (par exemple, la combinaison d'une métrique de qualité en déclin,

d'un patient aberrant et d'une mise à jour concurrente de source de données qui ensemble justifient une escalade mais seraient individuellement anodins). Le choix du niveau de sophistication EDA n'est pas neutre pour la conception de l'IA agentique : il détermine la richesse du champ perceptif disponible pour l'agent et, par conséquent, la complexité des situations qu'il peut interpréter et sur lesquelles il peut agir. Plus précisément, *le niveau de sophistication du traitement d'événements constitue une borne supérieure de la complexité situationnelle qu'un agent peut percevoir et sur laquelle il peut agir*. Un agent connecté à un simple filtre d'événements ne peut pas détecter des situations composites multi-flux, quelle que soit sa capacité de raisonnement. Cette observation a une implication directe en matière de conception : augmenter l'intelligence de l'agent sans augmenter le substrat de traitement d'événements produit des rendements architecturaux décroissants.

Les plateformes cloud modernes (Microsoft Azure Event Grid, AWS EventBridge, Apache Kafka, Confluent, Pulsar) fournissent l'infrastructure pour l'EDA à grande échelle, supportant des patterns allant du simple pub-sub au traitement de flux complexe. La maturité architecturale de l'EDA est bien établie. Ce qui n'a pas été suffisamment exploré est sa pertinence spécifique comme substrat pour les systèmes d'IA agentique en environnement d'entreprise.

3. Cadre : cinq fonctions structurantes de l'EDA pour l'IA agentique

Cette section propose un cadre identifiant cinq fonctions structurantes que l'architecture event-driven apporte à l'IA agentique. Ces fonctions ne sont pas de simples commodités opérationnelles. Elles répondent à des exigences architecturales fondamentales qui déterminent si un système agentique peut passer de la démonstration au déploiement industriel. Un caveat important s'applique à l'ensemble : toutes les architectures event-driven ne fournissent pas le même degré de visibilité temporelle, de rejouabilité ou de support d'audit. Ces propriétés dépendent de combinaisons spécifiques d'event streaming (par opposition au messaging transitoire), de gouvernance des schémas, de politiques de rétention des événements et de mécanismes de corrélation de traces. Le cadre ci-dessous identifie des fonctions que l'EDA *peut* fournir et qui sont nécessaires à l'industrialisation de l'IA agentique, mais leur réalisation présuppose une discipline architecturale, pas seulement l'adoption d'un bus d'événements. Comme établi en section 2.2, le niveau de sophistication du traitement d'événements constitue une borne supérieure de la complexité situationnelle qu'un agent peut percevoir et sur laquelle il peut agir. Les cinq fonctions ci-dessous spécifient ce que cette borne supérieure doit recouvrir pour une IA agentique de niveau entreprise.

3.1. Ancrage temporel

Une architecture transactionnelle ou centrée sur la requête capture principalement des états ou des besoins exprimés ponctuellement. Une architecture event-driven capture des transitions. Elle ne demande pas seulement « que sait-on du système ? » mais « qu'est-il en train de lui arriver ? ». Cette distinction est architecturalement décisive.

Fowler (2005) l'articule clairement pour l'event sourcing : connaître l'état courant est parfois insuffisant ; il faut aussi pouvoir reconstruire comment cet état a été atteint. Même sans event sourcing intégral, cette intuition est structurante pour l'IA agentique. Un agent efficace dans le monde réel doit percevoir la dynamique, pas seulement le cliché instantané. Un agent clinique n'a pas seulement besoin d'un dossier patient à l'instant *t*. Il doit détecter qu'un résultat vient d'être reclassé, qu'une mesure capteur vient de sortir d'une plage normale, qu'un consentement vient d'être retiré, qu'un seuil de risque vient d'être franchi. L'événement est l'unité minimale qui expose ces transitions. L'EDA apporte à l'agent ce que l'on peut appeler une *sensibilité située au changement*.

3.2. Délégation graduée

L'IA agentique est fréquemment présentée à travers un imaginaire d'autonomie générale. Dans les environnements d'entreprise, et plus encore dans les environnements régulés, l'autonomie n'est presque jamais totale. Elle est déléguée par classes de situations, par seuils, par niveaux de criticité, par contextes d'usage, et sous mécanismes d'escalade. Cette observation n'est pas nouvelle en ingénierie des systèmes. La littérature en facteurs humains a depuis longtemps établi que l'allocation de fonctions entre humain et machine n'est pas binaire mais graduée. Sheridan (1992) a proposé une taxonomie des niveaux d'automatisation, du pleinement manuel à l'aide à la décision puis à l'automatisation complète. Parasuraman, Sheridan et Wickens (2000) l'ont affinée en un cadre multidimensionnel distinguant acquisition d'information, analyse d'information, sélection de décision et implémentation d'action. Ces travaux fournissent le fondement théorique de ce que nous appelons ici délégation graduée. Hollnagel et Woods (2005) soutiennent en outre, dans une perspective de systèmes cognitifs joints, qu'une coordination humain-machine efficace exige non seulement des niveaux d'autorité mais aussi une observabilité mutuelle et la capacité pour chaque partie d'ajuster le périmètre d'action de l'autre.

L'architecture event-driven ne remplace pas le problème d'allocation de fonctions issu des facteurs humains. Elle fournit un mécanisme architectural concret pour opérationnaliser des allocations différenciées d'autorité selon les classes de situations. Concrètement, le routage par contenu et le filtrage d'événements au niveau du consommateur permettent de différencier les politiques d'action par type d'événement, par métadonnées de sévérité ou par contexte de domaine. Tel type d'événement peut déclencher une action automatique. Tel autre ne suscite qu'une recommandation. Tel

autre exige une validation humaine. Tel autre ne fait qu'alimenter la surveillance et la journalisation. L'agent n'est pas « libre ». Il est autorisé à intervenir sur certaines familles d'événements, selon des contrats explicites et des règles de gouvernance.

Cette articulation converge avec l'accent mis sur les garde-fous, l'authentification et la défense en profondeur dans les guides pratiques de construction d'agents (OpenAI, 2025 ; Anthropic, 2025). Elle converge avec l'insistance du profil IA générative du NIST AI RMF sur la compétence opérateur, la surveillance continue et l'évaluation des configurations humain-IA (NIST, 2024, §§ GV-1, MG-3.2, MS-2.6). Elle converge avec les exigences du règlement européen sur l'IA en matière de mesures de contrôle humain et d'enregistrement automatique des événements tout au long du cycle de vie du système (règlement (UE) 2024/1689, articles 14 et 12). Plus largement, elle répond à la reconnaissance croissante que le *contrôle humain significatif* (*meaningful human control*) sur les systèmes d'IA exige non seulement l'autorité formelle d'intervenir, mais des structures architecturales qui rendent l'intervention informée, opportune et traçable (Santoni de Sio & van den Hoven, 2018). Sans de telles structures, le contrôle risque de devenir ce que la littérature sur l'automatisation appelle le *biais d'automatisation*, une tendance à sur-estimer la fiabilité des sorties du système précisément parce que l'opérateur manque de visibilité suffisante sur le raisonnement et le contexte du système (Parasuraman & Riley, 1997).

3.3. Découplage inter-systèmes

Les systèmes d'information d'entreprise sont hétérogènes, stratifiés, partiellement legacy, partiellement cloud-native, souvent multi-fournisseurs et rarement réécrits de zéro. L'EDA est précieuse parce qu'elle permet à des services découplés de communiquer via événements en maintenant l'asynchronie et en limitant les dépendances directes (Microsoft, 2026 ; Hohpe & Woolf, 2003 ; Richardson, 2018).

Pour l'IA agentique, cette propriété est décisive. Un agent connecté uniquement par des appels API synchrones reste fortement couplé à des services spécifiques ou à des modèles conversationnels. Un agent qui consomme des événements métier et publie les siens devient insérable dans une topologie plus large. Il n'a pas besoin d'être le centre du système. Il devient un composant participant.

Cela a une conséquence stratégique majeure : l'EDA rend l'IA agentique adoptable par incréments. Il devient possible d'insérer des capacités agentiques le long de flux existants, de les instrumenter progressivement, de les gouverner, de les observer, puis d'étendre leur périmètre. C'est nettement plus crédible qu'une refonte centrée sur un méta-orchestrateur supposé tout comprendre. Cependant, le découplage event-driven n'est pas exempt de ses propres formes de couplage : les schémas d'événements partagés créent des dépendances de contrat, les flux d'événements partagés introduisent un couplage temporel, et l'infrastructure partagée crée un couplage

opérationnel. Ces compromis sont réels et bien documentés (voir section 7.2). L'argument ici n'est pas que l'EDA élimine le couplage, mais qu'elle transforme un couplage séquentiel étroit en formes de coordination plus souples, plus gouvernables et plus observables.

3.4. Auditabilité et reconstruisibilité causale

La question de l'auditabilité est l'une des faiblesses structurelles des discours naïfs sur l'IA agentique. Dans un système purement conversationnel ou fondé sur des appels ponctuels, il est difficile de reconstituer de manière robuste pourquoi l'agent a agi à tel moment, sur la base de quels signaux, avec quel contexte, en déclenchant quelles actions aval. Les exigences de gouvernance poussent précisément dans le sens inverse.

Le règlement européen sur l'IA exige, pour les systèmes à haut risque, que les systèmes permettent l'enregistrement automatique des événements tout au long de leur durée de vie (article 12), que les déployeurs conservent les journaux générés par le système (article 26(1)), et que les mesures de contrôle humain incluent la capacité de comprendre les capacités et limitations du système et de surveiller son fonctionnement (article 14(4)). Le NIST AI RMF met l'accent sur la surveillance continue, le suivi des configurations humain-IA et la capacité de surveiller les impacts et comportements du système en déploiement (NIST, 2024, §§ MG-3.2, MS-2.6, MS-2.7).

L'EDA fournit ici une brique particulièrement adaptée. Parce qu'elle formalise la circulation de signaux discrets, datés et corrélables, elle permet de reconstruire une chaîne causale : événement source, enrichissement, inférence, décision, action, échec éventuel, intervention humaine, correction, reprise. Avec des patterns appropriés (notamment inspirés de l'event sourcing (Fowler, 2005)) il devient possible non seulement de savoir ce que l'agent a fait, mais de reconstruire comment il en est arrivé là. Cette capacité est de plus en plus reconnue non comme un luxe de gouvernance, mais comme un prérequis à l'industrialisation des systèmes agentiques en environnement régulé.

3.5. Écologie multi-agents distribuée

Les systèmes agentiques sont souvent décrits sous la forme d'un agent principal, parfois assisté d'outils ou d'agents subordonnés. Les guides pratiques reconnaissent cependant que, lorsque le nombre d'outils et de tâches croît, il devient pertinent de répartir le travail entre plusieurs agents et de raisonner en termes d'orchestration (OpenAI, 2025). Cette direction fait écho à des principes anciens de la recherche sur les systèmes multi-agents, où la décomposition de problèmes complexes en agents spécialisés coopérants a été préconisée comme stratégie d'ingénierie fondamentale (Wooldridge & Jennings, 1995 ; Jennings, 2000). Des cadres récents comme AutoGen (Wu et al., 2023), CrewAI et LangGraph transposent ce paradigme à l'ère des LLM, permettant des configurations multi-agents coopératives et compétitives.

L'EDA fournit le milieu naturel de cette distribution. Des agents spécialisés peuvent consommer des événements différents, publier les leurs, enrichir un contexte partagé, collaborer ou se concurrencer sous arbitrage de règles de gouvernance. Plutôt qu'un super-agent central supposé omniscient, on obtient une écologie de capacités spécialisées. Cette perspective est conceptuellement plus robuste et plus compatible avec l'architecture d'entreprise réelle (Newman, 2021 ; Richardson, 2018).

Elle permet aussi d'éviter un défaut fréquent des systèmes agentiques centralisés : la concentration de la décision dans un point unique difficile à auditer, à faire évoluer et à sécuriser. En architecture distribuée, le découplage n'est pas un fétiche. C'est un moyen de maintenir l'évolutivité, la résilience et la lisibilité des responsabilités.

3.6. Synthèse du cadre

Fonction	Problème adressé	Capacité EDA	Conditions d'implémentation	Pertinence réglementaire
Ancrage temporel	L'agent ne perçoit que des clichés	Les événements capturent les transitions d'état avec horodatage	Politique de rétention des événements ; log en ajout seul ; garanties d'ordonnancement temporel	NIST MG-3.2 (surveillance continue)
Délégation graduée	L'agent est soit pleinement autonome, soit pleinement contraint	Politiques d'action par type d'événement	Routage par contenu ; métadonnées de sévérité ; règles d'escalade ; application de politiques côté consommateur	AI Act art. 14 (contrôle humain) ; NIST GV-1
Découplage inter-systèmes	Agent fortement couplé à des API/services spécifiques	Pub-sub et streaming avec consommateurs asynchrones	Gouvernance des schémas ; versioning des contrats ; isolation des consommateurs	
Auditabilité	Chaîne causale de l'action agentique non reconstruisible	Flux d'événements discrets, datés et corrélables	Identifiants de corrélation ; journalisation en ajout seul ; capacité de rejeu ; stratégie de	AI Act art. 12, 26(1) (journalisation) ; NIST MS-2.6

Fonction	Problème adressé	Capacité EDA	Conditions d'implémentation	Pertinence réglementaire
			rétention de piste d'audit	
Écologie distribuée	Super-agent centralisé non scalable et non auditable	Coordination agentique découplée par événements	Registre d'agents ; limites de responsabilité ; routage d'événements arbitré par la gouvernance	AI Act art. 9 (indirectement : la distribution augmente les besoins de gestion des risques sur le cycle de vie)

Tableau 1. Cadre : cinq fonctions structurantes de l'EDA pour l'IA agentique, avec conditions d'implémentation.

4. Cœur conceptuel : la double insuffisance

Les cinq fonctions formalisées ci-dessus reposent sur une observation conceptuelle plus fondamentale qui mérite d'être explicitée.

Réagir n'est pas interpréter. Une architecture event-driven sans couche agentique reste un système capable de propager et traiter des signaux, mais pas nécessairement d'en produire une interprétation contextuellement riche. Elle peut automatiser, router, transformer, déclencher. Elle ne saisit pas nécessairement l'ambiguïté d'une situation, le conflit entre objectifs, la nécessité d'une escalade humaine ou le caractère inédit d'un cas. L'IA agentique apporte précisément cette couche d'interprétation, de décision contextuelle, de sélection d'outils et de reformulation du problème. La distinction entre simple automatisation, exécution outillée et comportements plus généraux orientés objectif, soulignée dans les taxonomies récentes (Sapkota et al., 2025) se projette sur ce gradient.

Comprendre n'est pas participer. Inversement, un agent doté de raisonnement et d'outils ne participe pas automatiquement au système. Il peut rester périphérique, invoqué à la demande, sans exposition continue aux changements significatifs, sans inscription dans le tissu des interactions asynchrones, sans véritable matérialité opérationnelle.

La proposition théorique centrale de cet article est donc la suivante :

L'IA agentique apporte la capacité d'interprétation et de décision. L'architecture event-driven apporte la capacité de perception située, d'inscription dans le temps et de raccordement au système réel. Sans la première, le système réagit sans comprendre. Sans la seconde, il comprend sans réellement participer.

Cette proposition fournit un critère d'évaluation architecturale. Lorsqu'un projet se revendique « agentique », il faut examiner non seulement les capacités du modèle, mais aussi sa relation aux événements, aux transitions métier, à la journalisation, aux droits d'action, aux mécanismes d'escalade et aux contrats de responsabilité.

4.1. Un spectre de maturité architecturale

Le prisme de la double insuffisance suggère une classification des systèmes agentiques le long d'un spectre de maturité architecturale, résumé dans le Tableau 2.

Niveau	Type de système	Capacité cognitive	Perception située	Caractérisation architecturale
1	Agent conversationnel	Élevée (raisonnement LLM)	Aucune	Requête/réponse ; structurellement déconnecté de la dynamique du système
2	Agent outillé	Élevée (raisonnement + outils)	Aucune (invoqué à la demande)	Appels API ; pas de perception continue des changements d'état
3	Automatisation située	Faible (règles)	Élevée (event-driven)	Perçoit les transitions, réagit aux signaux ; manque de raisonnement contextuel
4	Système agentique pleinement articulé	Élevée	Élevée	Consomme et produit des événements ; raisonne contextuellement ; délégation gouvernée ; traces causalement reconstruisibles

Tableau 2. Spectre de maturité architecturale des systèmes agentiques.

Au Niveau 1, les *agents conversationnels* opèrent en mode requête/réponse pur, cognitivement capables mais structurellement déconnectés de la dynamique d'entreprise. Au Niveau 2, les *agents outillés* peuvent invoquer des API et exécuter des procédures bornées, mais restent invoqués à la demande. Au Niveau 3, les *systèmes situés mais non interprétants* (les automatisations event-driven traditionnelles)

perçoivent les transitions et réagissent aux signaux, mais n'ont pas le raisonnement contextuel nécessaire pour traiter l'ambiguïté, la nouveauté ou les objectifs concurrents. Au Niveau 4, les *systèmes agentiques pleinement articulés* combinent capacité cognitive et perception située.

Ce spectre n'est pas seulement descriptif. Il fournit un outil diagnostique : la maturité architecturale d'un système agentique peut être évaluée en examinant laquelle de ces quatre positions il occupe effectivement, indépendamment des déclarations marketing. Le cadre proposé en section 3 spécifie ce qui est architecturalement requis pour passer du Niveau 2 au Niveau 4, la transition que la plupart des projets d'IA agentique d'entreprise devront accomplir.

5. Étude de cas : du pipeline AutoML à la componentisation event-driven dans la génération de jumeaux numériques cliniques

5.1. Contexte

L'étude de cas est issue de l'évolution architecturale de TweenMe®, notre plateforme de génération de jumeaux numériques pour la synthèse d'évidence clinique, opérant dans le domaine de l'oncologie des tumeurs solides avancées. Le concept de jumeau numérique (une représentation virtuelle d'une entité physique maintenue en synchronisation tout au long de son cycle de vie (Grieves & Vickers, 2017)) est ici appliqué à l'évidence clinique au niveau patient. La plateforme traite des données cliniques en vie réelle (environ 180 patients répartis sur plusieurs dizaines de jeux de données sources conformes à la structure de domaines CDISC/SDTM) pour générer des cohortes synthétiques de patients permettant une analyse d'efficacité comparative entre séquences thérapeutiques. Le système a été validé sur plusieurs critères statistiques, incluant la concordance entre distributions de survie synthétiques et observées et une précision train-on-synthetic-test-on-real supérieure à 95 %.

La plateforme opère dans un contexte régulé où les exigences du règlement européen sur les dispositifs médicaux (règlement (UE) 2017/745) (MDR), du règlement européen sur l'IA et des standards établis de gouvernance des données cliniques imposent des contraintes spécifiques de traçabilité, de reproductibilité et de contrôle humain. La voie de qualification précise , que le système relève du règlement sur l'IA comme système à haut risque au titre de l'annexe III, du MDR comme composant d'un dispositif médical, ou des deux par leur interaction, dépend de la finalité prévue du système, de la qualification produit et de la voie de conformité, et n'est pas tranchée ici. Ce qui importe pour l'argument architectural est que toutes les voies de qualification plausibles imposent des exigences substantivement similaires en matière de journalisation, de traçabilité et de contrôle humain. Cette intersection réglementaire est caractéristique des défis

architecturaux auxquels font face les systèmes cliniques pilotés par l'IA (Topol, 2019 ; Rajkomar et al., 2019).

5.2. Architecture initiale : pipeline AutoML monolithique

L'architecture initiale suit un schéma classique de pipeline séquentiel : ingestion et extraction depuis les jeux de données sources, harmonisation ontologique des données, augmentation synthétique, génération de cohortes et modélisation statistique aval (régression en risques compétitifs sur de multiples critères cliniques par séquence thérapeutique). Ce pipeline est fonctionnel et a produit des résultats statistiquement validés.

Cependant, il présente les propriétés structurelles d'un monolithe. Chaque ajout fonctionnel (un nouveau critère clinique, une nouvelle source de données, un nouveau modèle de risque, une nouvelle contrainte réglementaire) impose de revoir le flux de bout en bout. Le couplage est fort : les modifications amont se propagent de manière imprévisible aux composants aval. La traçabilité est implicite, dépendante d'une journalisation ad hoc plutôt que de traces événementielles architecturalement imposées. Plus critiquement, le pipeline n'offre pas de point d'insertion naturel pour des composants intelligents opérant orthogonalement au flux principal, comme un agent surveillant la qualité des cohortes synthétiques, détectant des incohérences inter-bras, vérifiant en continu les contraintes réglementaires, ou déclenchant des alertes lorsque des seuils de validité statistique sont franchis (voir Figure 1a).

5.3. Pressions fonctionnelles révélant les limites architecturales

Avec l'extension du périmètre opérationnel de la plateforme, plusieurs catégories d'exigences fonctionnelles ont émergé que l'architecture pipeline ne pouvait pas accueillir proprement :

Surveillance continue de la qualité. Le besoin de valider les propriétés des cohortes synthétiques non seulement au moment de la génération mais tout au long de l'analyse aval, en détectant la dérive ou la dégradation lorsque de nouvelles données ou de nouveaux paramètres sont introduits.

Vérification réglementaire transversale. Le besoin de composants qui vérifient les contraintes de conformité (provenance des données, statut du consentement, exigences d'explicabilité des modèles) indépendamment du flux analytique principal, sans être embarqués dans la logique séquentielle du pipeline.

Détection d'anomalies et alertes. Le besoin de détecter des patterns inattendus (anomalies statistiques dans les comparaisons inter-bras de traitement, patients aberrants, problèmes d'intégrité des données) et de les router vers les gestionnaires appropriés (correction automatique, revue humaine, ou journalisation seule) selon la sévérité.

Extensibilité modulaire. Le besoin d'ajouter de nouvelles capacités analytiques (nouveaux types de modèles, nouvelles couches de visualisation, nouvelles fonctions de reporting) sans disruption architecturale des composants existants validés.

Dans l'architecture pipeline, chacune de ces exigences ne peut être adressée qu'en modifiant le flux principal (augmentant le couplage et le risque de régression) ou en construisant des canaux latéraux ad hoc (accumulant dette technique et opacité). Aucune de ces voies n'est compatible avec les exigences de gouvernance et de maintenabilité d'un système clinique régulé.

Il convient de noter que ces pressions ne sont pas de poids égal. La première (surveillance continue de la qualité) et la troisième (détection d'anomalies et alertes) ont été les déclencheurs primaires de la reconsidération architecturale, parce qu'elles affectent directement la validité et la sécurité des productions cliniques et parce qu'elles exigent une observation continue et transversale qu'un pipeline séquentiel ne peut structurellement fournir. La deuxième (vérification réglementaire) et la quatrième (extensibilité modulaire) sont des bénéfices anticipés de l'évolution architecturale plutôt que des déclencheurs isolés. Elles deviennent réalisables comme conséquences du découplage que les deux premières pressions nécessitent.

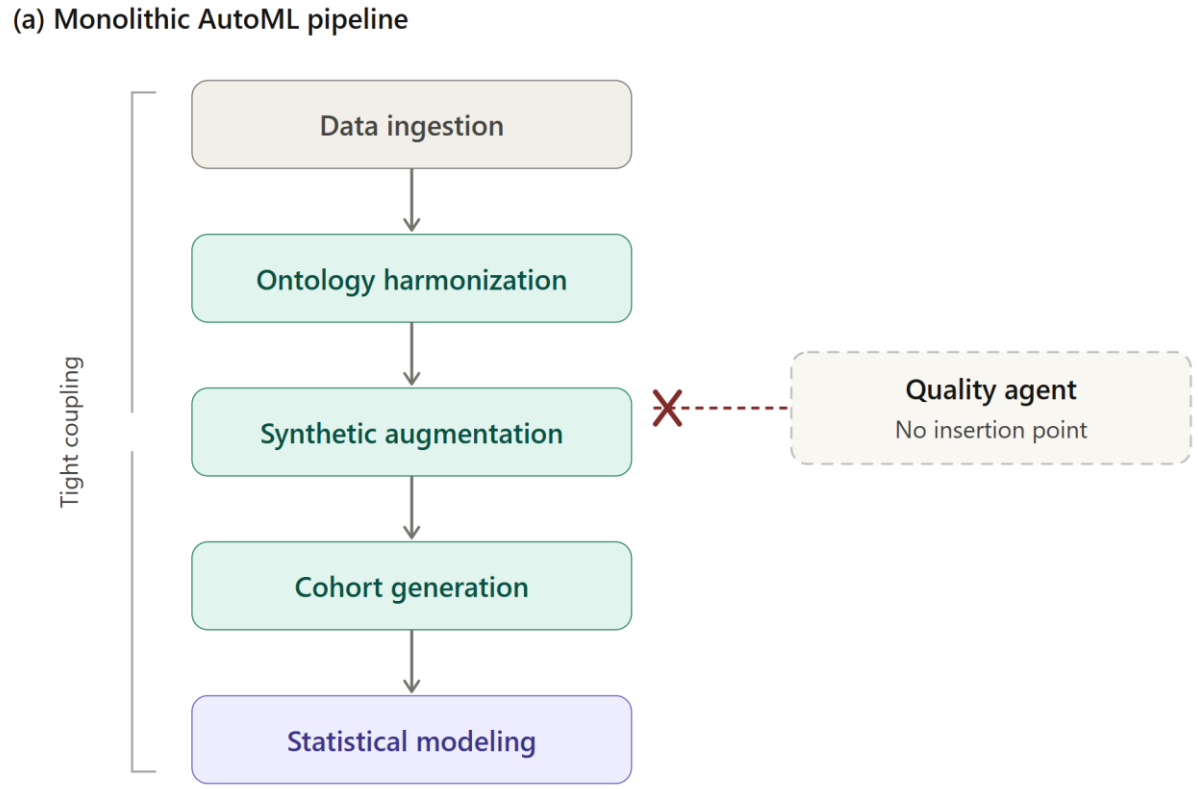
5.4. Architecture cible : componentisation event-driven

La réponse architecturale, actuellement en phase de conception, est une componentisation du pipeline existant par découplage event-driven. Le terme « componentisation » est employé ici au sens de Szyperski (2002) : la décomposition d'un système en unités indépendamment déployables et remplaçables, avec des interfaces bien définies, où la composition est gouvernée par des contrats plutôt que par un couplage séquentiel. Cette transformation est analogue, dans sa logique architecturale, à ce que le mouvement des microservices a opéré sur les applications monolithiques (Dragoni et al., 2017 ; Newman, 2021) non pas simplement l'ajout de nouveaux composants, mais la restructuration fondamentale de la forme du couplage, du déploiement, de l'observabilité et de la responsabilité (Bass et al., 2021). Le principe central est que les modules de traitement existants deviennent producteurs et consommateurs d'événements métier typés (*cohorte générée*, *validation statistique réussie/échouée*, *seuil de risque franchi*, *anomalie détectée*, *contrainte réglementaire vérifiée/violée*) plutôt qu'étapes d'un flux séquentiel fixe.

Les nouveaux composants, y compris des agents cognitifs, s'insèrent en souscrivant aux événements pertinents pour leur fonction, sans modifier les producteurs amont (voir Figure 1b). Un agent responsable de la surveillance de la qualité des cohortes souscrit aux événements *cohorte générée* et *validation statistique*. Un agent de conformité réglementaire souscrit aux événements *provenance des données* et *statut du consentement*. Un agent de détection d'anomalies souscrit aux événements

comparaison inter-bras et publie des événements *anomalie détectée* qui sont à leur tour consommés par un gestionnaire d'escalade.

Au niveau des patterns, l'architecture cible repose sur une infrastructure d'event streaming fondée sur un log (par distinction avec un courtier de messages transitoire), choisie pour sa capacité à supporter le rejeu, les consommateurs à liaison tardive et un historique d'événements durable, propriétés directement pertinentes pour les fonctions d'auditabilité et d'ancrage temporel du cadre. Les événements se conforment à un schéma typé avec versioning explicite et un identifiant de corrélation qui lie tous les événements provenant de la même exécution analytique, permettant la reconstruction causale sur l'ensemble de la chaîne de traitement. L'idempotence côté consommateur est imposée pour gérer les re-livraisons sans effets de bord. Le routage par contenu dirige les événements vers l'agent ou le gestionnaire approprié en fonction du type d'événement et des métadonnées de sévérité, implémentant les politiques de délégation graduée décrites en section 3.2. Ce niveau de spécificité de pattern est délibérément contraint aux décisions de phase de conception ; les choix d'implémentation (plateforme de streaming spécifique, technologie de registre de schémas, format de sérialisation) restent ouverts et ne sont pas architecturalement structurants.



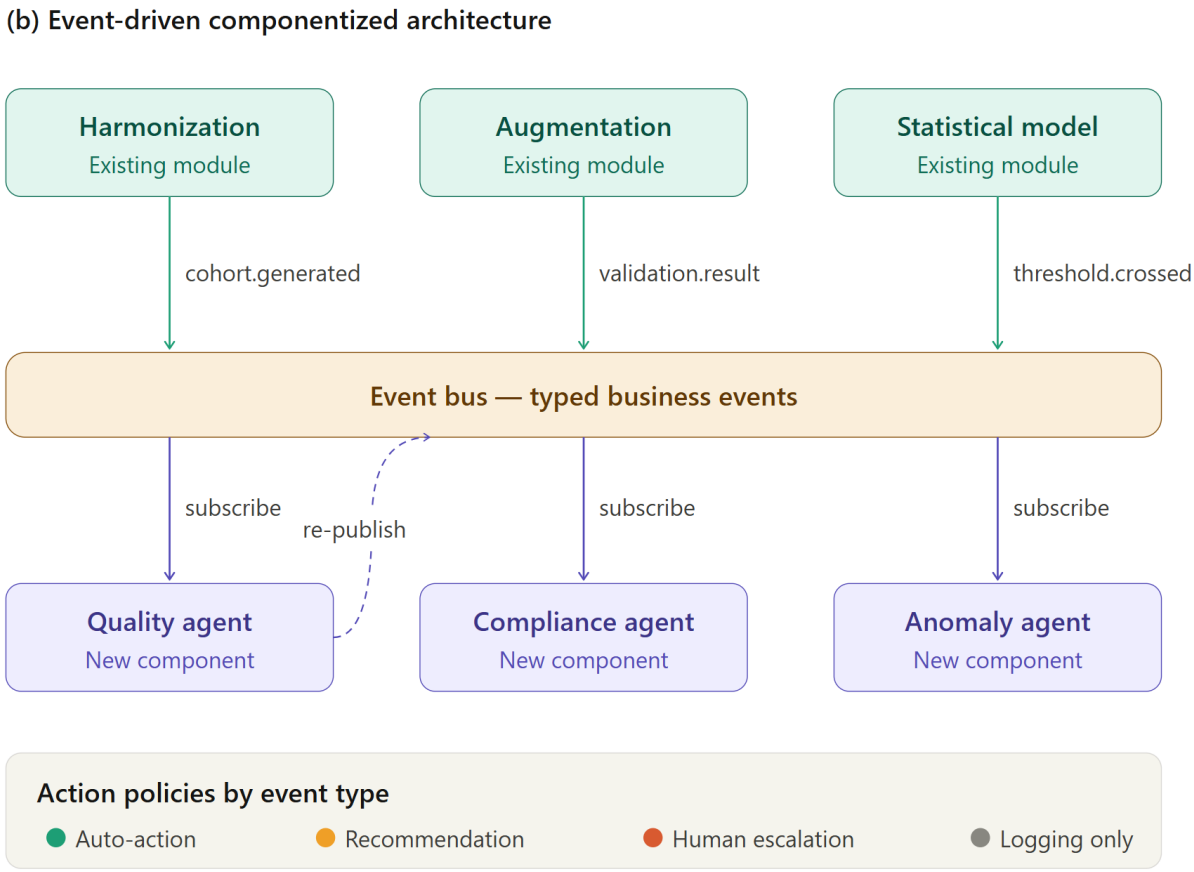


Figure 1. Évolution architecturale du pipeline monolithique à la componentisation event-driven. *(a)* Le pipeline AutoML initial : étapes de traitement séquentielles avec couplage fort et absence de point d'insertion naturel pour de nouveaux composants intelligents. *(b)* L'architecture event-driven cible : les modules existants publient des événements métier typés sur un bus d'événements partagé ; les nouveaux agents (qualité, conformité, détection d'anomalies) souscrivent aux événements pertinents sans modifier les producteurs amont. Les politiques d'action gouvernent le type de réponse (action automatique, recommandation, escalade humaine ou journalisation seule) par classe d'événement.

Cette architecture instancie directement les cinq fonctions identifiées dans le cadre (Tableau 1). Les événements sont horodatés et séquencés, fournissant l'ancrage temporel. Des politiques par type d'événement implémentent la délégation graduée : un événement *seuil de risque franchi* sur un critère principal déclenche une revue humaine, tandis qu'un événement routinier *validation statistique réussie* est journalisé sans escalade. De nouveaux agents sont ajoutés sans modifier les modules amont validés, réalisant le découplage inter-systèmes. Le flux d'événements fournit une piste d'audit causalement reconstruisible depuis l'ingestion des données jusqu'à la décision et l'action. De multiples agents spécialisés opèrent concurremment sur différents flux d'événements sous des politiques de gouvernance explicites, constituant une écologie distribuée plutôt qu'un pipeline centralisé.

Jérôme Vetillard
CTO | VP R&D | Chief Product Officer | AI-Powered Healthcare & Life Sciences Products | Compliance by Design | PhD AgroParisTech | CPO MIT Sloan | Exec MBA IE Business School & Brown University
Twingital-institute / Twingital-ventures : twingital-ventures.com

Ce que le cas révèle au-delà du cadre lui-même est l'asymétrie de difficulté d'implémentation entre les cinq fonctions. L'ancrage temporel et le découplage sont des conséquences relativement directes de l'adoption d'une infrastructure d'event streaming fondée sur un log. L'auditabilité exige un effort de conception supplémentaire : spécifiquement, le schéma d'identifiants de corrélation et la stratégie de rétention doivent être conçus avec l'audit réglementaire en tête, et non simplement le débogage opérationnel. La délégation graduée est la fonction architecturalement la plus exigeante : elle nécessite non seulement le routage et le filtrage d'événements, mais une couche de politique explicite qui mappe les types d'événements et les métadonnées de sévérité sur des classes de réponse autorisées, une couche qui n'existe nativement dans aucune plateforme d'event streaming actuelle et doit être ingéniérée comme une préoccupation applicative.

5.5. Note méthodologique

Cette étude de cas est un *cas explicatif orienté design* : elle utilise l'évolution architecturale d'un système réel pour illustrer et concrétiser un cadre conceptuel, non pour le valider empiriquement. L'évolution architecturale est actuellement en phase de conception, motivée par les limitations opérationnelles rencontrées durant la phase de validation du pipeline initial. Elle est présentée ici non comme un système déployé, mais comme un raisonnement de conception (*design rationale*) fondé sur l'expérience de première main des contraintes des pipelines d'IA clinique monolithiques en environnement régulé.

Le cas est de fait *favorable* au cadre : les cinq fonctions se projettent proprement sur les besoins documentés. Ce n'est pas une coïncidence, le cadre a été développé en partie à partir de la réflexion sur cette expérience. L'honnêteté intellectuelle de l'article dépend de la reconnaissance de ce fait : le cas illustre le cadre plutôt qu'il ne le teste indépendamment. Le cas n'est pas utilisé pour valider le cadre au sens positiviste, mais pour éprouver son adéquation explicative face à une transition architecturale concrète observée de l'intérieur. Sa valeur réside dans la démonstration que les cinq fonctions structurantes correspondent à des besoins architecturaux concrets et documentés dans un système réel opérant sur des données cliniques réelles sous des contraintes de gouvernance réelles et non dans la fourniture d'une comparaison contrôlée avec des approches architecturales alternatives.

Une note sur les contraintes temporelles s'impose. La plateforme opère sur des données rétrospectives en vie réelle et non sur un suivi patient en temps réel, ce qui signifie que les exigences de latence pour la propagation d'événements et la réponse des agents se mesurent en secondes à minutes plutôt qu'en millisecondes. C'est un régime matériellement différent des systèmes d'alertes cliniques en temps réel où des budgets de réponse sub-seconde s'appliquent. Cependant, la contrainte temporelle qui importe pour le cas présent n'est pas la latence brute mais *l'ordonnancement causal et la*

complétude : les événements doivent arriver aux consommateurs dans une séquence qui préserve leurs dépendances logiques, et aucun événement dans la chaîne analytique ne peut être silencieusement perdu sans détection. Ces contraintes (garanties d'ordonnancement et livraison au-moins-une-fois avec idempotence côté consommateur) sont bien dans les capacités des architectures d'event streaming fondées sur un log (Kleppmann, 2017), mais elles doivent être explicitement adressées dans la conception plutôt qu'assumées.

6. Correspondance avec les cadres de gouvernance

L'articulation architecturale proposée dans cet article n'est pas motivée uniquement par des considérations d'ingénierie. Elle répond à un ensemble de plus en plus explicite d'exigences de gouvernance qui s'appliquent aux systèmes d'IA opérant en environnement d'entreprise et régulé.

6.1. Le règlement européen sur l'intelligence artificielle

Le règlement (UE) 2024/1689 établit un cadre fondé sur le risque pour les systèmes d'IA au sein de l'Union européenne. Pour les systèmes d'IA à haut risque (une catégorie qui peut englober les systèmes d'aide à la décision clinique et de diagnostic, selon la finalité prévue et la voie de conformité) le règlement impose plusieurs exigences d'une pertinence architecturale directe.

L'article 9 exige l'établissement, la mise en œuvre, la documentation et la maintenance d'un système de gestion des risques tout au long du cycle de vie du système d'IA à haut risque. Une architecture event-driven supporte cette exigence en fournissant des flux continus et observables de comportement du système pouvant alimenter les processus de surveillance et d'évaluation des risques.

L'article 12 impose que les systèmes d'IA à haut risque incluent la capacité technique d'enregistrement automatique des événements (journaux) tout au long de leur durée de vie. La disposition précise que la journalisation doit permettre la surveillance du fonctionnement du système et la surveillance après mise sur le marché. Une architecture event-driven, par sa conception fondamentale, produit précisément un tel flux d'événements discrets, horodatés et enregistrables.

L'article 14 exige des mesures permettant le contrôle humain, incluant la capacité pour des personnes physiques de comprendre les capacités et limitations du système, de surveiller correctement son fonctionnement et, le cas échéant, d'intervenir dans son fonctionnement ou de l'interrompre. La délégation graduée par politiques d'action par type d'événement, telle que décrite dans le cadre, fournit un mécanisme structurel pour implémenter un tel contrôle.

L'article 26(1) exige des déployeurs de systèmes d'IA à haut risque qu'ils conservent les journaux générés automatiquement par le système, dans la mesure où ces journaux sont sous leur contrôle. Cette exigence présuppose une architecture capable de produire et de conserver des journaux opérationnels significatifs, une propriété que les systèmes event-driven fournissent nativement.

6.2. Le NIST AI Risk Management Framework

Le NIST AI RMF (NIST, 2023) et son profil IA générative (NIST, 2024) fournissent un cadre volontaire mais largement référencé pour la gestion des risques associés aux systèmes d'IA. Plusieurs dispositions sont directement pertinentes.

La fonction Gouverner (GV-1) appelle à établir des politiques, processus et procédures pour cartographier, mesurer et gérer les risques de l'IA, incluant la définition des rôles et responsabilités pour les configurations humain-IA. La délégation graduée event-driven supporte cette exigence en rendant les limites de l'autorité de l'agent architecturalement explicites.

La fonction Gérer (MG-3.2) appelle à la surveillance continue des systèmes d'IA en déploiement, incluant le suivi des propriétés émergentes et des comportements inattendus. Une architecture event-driven fournit le substrat d'observabilité pour une telle surveillance.

La fonction Mesurer (MS-2.6, MS-2.7) appelle à l'évaluation de l'interaction humain-IA et des impacts des systèmes d'IA dans les contextes de déploiement. La reconstruisibilité causale permise par la journalisation event-driven supporte directement une telle évaluation.

6.3. Implications

La convergence entre les cinq fonctions structurantes de l'EDA pour l'IA agentique (Tableau 1) et les exigences des cadres de gouvernance actuels n'est ni accidentelle ni superficielle. Elle reflète une préoccupation sous-jacente partagée : les systèmes d'IA qui agissent dans le monde réel doivent être observables, contrôlables, auditables et gouvernables. Ces propriétés ne sont pas de simples surcouches réglementaires. Ce sont des exigences architecturales. Lorsqu'elle est correctement implémentée (avec une gouvernance des schémas appropriée, une rétention des événements, une corrélation de traces et une supervision graduée) l'architecture event-driven fournit une base structurelle pour les satisfaire. Cet alignement n'est pas automatique : une architecture event-driven mal gouvernée, mal corrélée, mal historisée ne satisfait rien. Mais la forme architecturale est nativement compatible avec les exigences de gouvernance d'une manière que les architectures requête/réponse ou orientées batch ne le sont pas.

---

7. Limites et directions de recherche future

7.1. Périmètre d'applicabilité

Tous les cas d'usage d'IA agentique ne nécessitent pas une architecture event-driven. Un assistant de rédaction, un système de recherche documentaire, un copilote individuel ou un agent opérant dans une interaction brève et bornée peuvent fonctionner adéquatement dans un modèle requête/réponse enrichi. Les guides pratiques recommandent à juste titre de commencer par des systèmes simples avant d'ajouter de la complexité d'orchestration (OpenAI, 2025 ; Anthropic, 2025). La thèse défendue ici est plus précise : dans les environnements d'entreprise caractérisés par l'hétérogénéité des systèmes et le besoin d'action causalement reconstruisible sous contraintes de gouvernance, l'EDA devient le complément architectural essentiel.

7.2. Complexité de l'EDA

Il serait naïf de penser qu'ajouter un bus d'événements suffit à transformer un prototype agentique en système robuste. Les systèmes distribués asynchrones introduisent une complexité bien documentée : contrôle de flux, gestion des doublons, idempotence, ordonnancement, versioning, corrélation, reprise, files de lettres mortes et débogage distribué (Hohpe & Woolf, 2003 ; Kleppmann, 2017). Une architecture agentique event-driven sérieuse présuppose au minimum : des contrats d'événements stables et sémantiquement clairs, une gouvernance des schémas et des versions, des mécanismes d'idempotence et de corrélation, une journalisation exploitable, une politique explicite de droits d'action, une supervision humaine graduée et une observabilité transverse.

7.3. Rôle complémentaire des API

Affirmer le caractère essentiel de l'EDA ne revient pas à abolir les API. Tupe et al. (2025) montrent que les API doivent évoluer pour mieux supporter les workflows agentiques. Mais dans une architecture mature, les API deviennent des points d'action, des interfaces d'interrogation ou des points de mutation exposés à des agents déjà situés dans une topologie de signaux plus large. L'opposition n'est pas « API ou événements ». Elle est entre une vision pauvre de l'agent connecté uniquement par des appels synchrones et une vision riche où événements, API, règles, garde-fous et supervision humaine composent un système cohérent.

7.4. Validation empirique

L'étude de cas présentée en section 5 se limite à un raisonnement de conception fondé sur l'expérience opérationnelle. Elle n'inclut pas de données de performance empiriques sur l'architecture cible, qui est encore en phase de conception. Les travaux futurs devraient inclure une évaluation quantitative de l'architecture proposée selon les dimensions de complétude de la traçabilité, de surcoût de gouvernance et de coût

d'extensibilité, ainsi qu'une analyse comparative avec des approches architecturales alternatives (orchestrateur centralisé, chorégraphie seule, patterns hybrides).

8. Conclusion : une proposition doctrinale pour l'architecture d'entreprise

Le débat sur l'IA agentique reste trop souvent capturé par la fascination pour les capacités de raisonnement, les outils et l'autonomie apparente. Ce débat demeure superficiel tant qu'il n'interroge pas le milieu architectural dans lequel cette intelligence doit opérer. L'entreprise réelle n'est pas un dialogue continu avec un assistant. C'est une dynamique distribuée de transitions d'état, de dépendances temporelles, de signaux métier, de validations, d'alertes, d'exceptions et de contraintes de gouvernance.

Dans ce contexte, l'architecture event-driven apparaît non comme une option esthétique, mais comme le complément essentiel de l'IA agentique, dans les environnements d'entreprise caractérisés par l'hétérogénéité des systèmes et le besoin d'action causalement reconstruisible sous contraintes de gouvernance. Elle donne à l'agent son ancrage temporel, permet une délégation graduée, facilite l'insertion dans des systèmes hétérogènes, rend l'action causalement reconstruisible et ouvre la voie à des écologies distribuées d'agents spécialisés. Les cadres de gouvernance actuels, du NIST AI RMF au règlement européen sur l'IA, renforcent cette direction en mettant l'accent sur la surveillance, la journalisation, le contrôle humain et la gestion continue des risques.

Le champ de l'IA agentique en entreprise est encore dans sa phase initiale de structuration. La proposition offerte ici n'est pas une doctrine définitive, mais une contribution à la stabilisation d'un domaine émergent. Elle peut être condensée comme suit :

L'IA agentique traite de cognition déléguée : interpréter, planifier, choisir, exécuter. L'architecture event-driven traite de sensibilité située au changement : capter des transitions, diffuser des signaux, coordonner des réactions asynchrones. Les systèmes d'entreprise ont besoin des deux. Sans IA agentique, l'EDA reste un système réactif mais cognitivement limité. Sans EDA, l'IA agentique reste un raisonneur périphérique souvent brillant sur sollicitation, mais peu situé et difficilement industrialisable. Dans les environnements d'entreprise caractérisés par l'hétérogénéité des systèmes et le besoin d'action causalement reconstruisible sous contraintes de gouvernance, leur articulation disciplinée permet de concevoir des systèmes capables non seulement de répondre, mais de participer, arbitrer, escalader, documenter et être supervisés.

La question stratégique n'est donc plus seulement : « Comment faire raisonner un agent ? » La vraie question est : dans quelle architecture de signaux, de responsabilités, de garde-fous et de traçabilité ce raisonnement peut-il devenir action fiable ?

Jérôme Vetillard
CTO | VP R&D | Chief Product Officer | AI-Powered Healthcare & Life Sciences Products | Compliance by Design | PhD AgroParisTech | CPO MIT Sloan | Exec MBA IE Business School & Brown University
Twingital-institute / Twingital-ventures : twingital-ventures.com

Cet article a soutenu que, dans une proportion croissante des cas d'usage d'entreprise, la réponse réside dans une articulation disciplinée entre IA agentique et architecture event-driven. La question de savoir si cette proposition tient à travers un éventail plus large de domaines, de traditions architecturales et de régimes de gouvernance est une question pour de futurs travaux empiriques et comparatifs.

Références

Anthropic. (2025). Building effective agents. Anthropic Research Blog.

AWS. (s.d.). Event-driven architecture. Amazon Web Services Documentation.

Bass, L., Clements, P., & Kazman, R. (2021). *Software Architecture in Practice* (4e éd.). Addison-Wesley.

Chandy, K. M., & Schulte, W. R. (2010). *Event Processing: Designing IT Systems for Agile Companies*. McGraw-Hill.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering* (pp. 195–216). Springer.

Fowler, M. (2005). Event sourcing. martinfowler.com.

Grieves, M., & Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary Perspectives on Complex Systems* (pp. 85–113). Springer.

Hohpe, G., & Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley.

Hollnagel, E., & Woods, D. D. (2005). *Joint Cognitive Systems: Foundations of Cognitive Systems Engineering*. CRC Press.

Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence*, 117(2), 277–296.

Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media.

Luckham, D. (2002). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley.

Michelson, B. M. (2006). Event-driven architecture overview. *Patricia Seybold Group*, 2(12), 10–1571.

Microsoft. (2026). Event-driven architecture style. Microsoft Learn. Mis à jour le 7 mars 2026.

Jérôme Vetillard
CTO | VP R&D | Chief Product Officer | AI-Powered Healthcare & Life Sciences Products | Compliance by Design | PhD AgroParisTech | CPO MIT Sloan | Exec MBA IE Business School & Brown University
Twingital-institute / Twingital-ventures : twingital-ventures.com

Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2e éd.). O'Reilly Media.

NIST. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1.

NIST. (2024). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. NIST AI 600-1.

OpenAI. (2025). A practical guide to building agents. OpenAI Documentation.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics Part A*, 30(3), 286–297.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, 39(2), 230–253.

Rajkomar, A., Dean, J., & Kohane, I. (2019). Machine learning in medicine. *New England Journal of Medicine*, 380(14), 1347–1358.

Red Hat. (s.d.). What is event-driven architecture? Red Hat Topics.

Règlement (UE) 2017/745 du Parlement européen et du Conseil du 5 avril 2017 relatif aux dispositifs médicaux (règlement sur les dispositifs médicaux). *Journal officiel de l'Union européenne*, L 117/1.

Règlement (UE) 2024/1689 du Parlement européen et du Conseil du 13 juin 2024 établissant des règles harmonisées concernant l'intelligence artificielle (règlement sur l'intelligence artificielle). *Journal officiel de l'Union européenne*, L 2024/1689.

Richardson, C. (2018). *Microservices Patterns: With Examples in Java*. Manning Publications.

Santoni de Sio, F., & van den Hoven, J. (2018). Meaningful human control over autonomous systems: A philosophical account. *Frontiers in Robotics and AI*, 5, 15.

Sapkota, R., et al. (2025). AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. *arXiv preprint*.

Sheridan, T. B. (1992). *Telerobotics, Automation, and Human Supervisory Control*. MIT Press.

Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming* (2e éd.). Addison-Wesley.

Topol, E. J. (2019). *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. Basic Books.

Tupe, V., et al. (2025). Adapting API architectures for the age of AI agents. *arXiv preprint*.

Wang, L., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 186345.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2), 115–152.

Wu, Q., et al. (2023). AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.

Xi, Z., et al. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Jérôme Vetillard
CTO | VP R&D | Chief Product Officer | AI-Powered Healthcare & Life Sciences Products | Compliance by Design | PhD AgroParisTech | CPO MIT Sloan | Exec MBA IE Business School & Brown University
Twingital-institute / Twingital-ventures : twingital-ventures.com