L'agent n'est pas l'architecture

Capacité, autorité et l'espace d'exécution admissible

Ceci est le premier article d'une série doctrinale en trois parties sur l'IA agentique et l'architecture d'entreprise. Cet article établit la fondation ontologique : les agents sont des choix d'implémentation, pas des primitives architecturales. Les deux articles suivants s'appuient sur cette fondation pour examiner la mécanique d'exécution et la gouvernance systémique.

1. L'erreur de catégorie

Le discours autour de l'IA agentique commet une erreur de catégorie fondatrice. Il confond modalité d'implémentation avec primitive architecturale.

Un agent, dans la réalité technique, est une modalité d'exécution probabiliste. Il combine l'inférence, l'outillage, l'accumulation d'état et la prise de décision déléguée. Cela change inévitablement ce qui se passe pendant l'exécution. Mais l'exécution n'est pas l'architecture. L'exécution est un choix opérationnel survenant au sein d'un cadre architectural existant.

Les règles métier sont des modalités d'exécution déterministes. Les microservices sont des modalités d'exécution structurées. Les humains sont des modalités d'exécution adaptatives. Chacune a transformé la façon dont les systèmes ont été construits. Aucune n'a exigé une nouvelle primitive architecturale.

L'erreur persiste : confondre choix d'implémentation avec nécessité catégorique. Une fois cette confusion établie, construire une discipline entière pour la valider.

Cet article propose une correction simple : l'agent n'est pas une nouvelle primitive architecturale. C'est un choix d'implémentation pour une responsabilité exécutable bornée au sein du modèle de capacité de l'organisation.

2. L'invariant, la capacité persiste

Chaque organisation doit répondre à une question fondamentale : Que devons-nous être capable de faire ?

La réponse est une capacité métier : une unité de capacité métier avec portée explicite, objectif économique et responsabilité envers les parties prenantes.

À partir de la capacité métier, une organisation dérive des responsabilités exécutables. Je les appelle capacités bornées : unités de travail avec entrées, sorties, autorité, portée contractuelle et propriété claires.

Une capacité bornée est définie formellement :

CB = (Objectif, Entrées, Sorties, Autorité, État, Contraintes, Responsabilité)

où :

- Objectif : le résultat métier attendu
- Entrées/Sorties : le contrat fonctionnel
- Autorité : quels systèmes ou décisions il peut invoquer ou faire
- État : quel état interne il peut accéder ou modifier
- Contraintes : les invariants qu'il doit respecter
- Responsabilité : le propriétaire responsable

C'est une construction analytique distincte de l'implémentation.

Une capacité bornée peut être réalisée de multiples façons :

Implémentation(CB, t) ∈ {Humain, Règle, Service logiciel, Modèle ML, Agent, Hybride}

Chaque choix satisfait les mêmes contraintes : contrat typé, permissions limitées, comportement observable, gestion du cycle de vie, identité et autorité.

Exemple pharmaceutique : SafetyTriageCapability (classifier les notifications de sécurité par risque) peut être implémentée comme un moteur de règles, un modèle ML classique, un microservice, un agent LLM, ou un spécialiste de pharmacovigilance humain. Pour chaque implémentation, le contrat de capacité bornée reste identique. Chacun doit opérer sur des entrées validées, respecter sa portée d'autorité, exposer des contrats clairs et être observable.

L'agent n'ajoute aucune primitive architecturale. Il représente un choix d'implémentation parmi plusieurs. La capacité métier est la primitive architecturale. L'agent est un choix d'implémentation.

3. La vraie nouveauté, la sémantique d'exécution a changé

Si l'ontologie architecturale reste stable (capacité comme primitive), qu'est-ce qui change concrètement ?

Les mécanismes de contrôle et la sémantique d'exécution, pas les principes fondamentaux.

Les principes d'EA (identité, moindre privilège, contexte borné, observabilité, interface contractuelle, gestion du cycle de vie) restent constants.

Ce qui change est comment ces principes opèrent sous le probabilisme.

L'agent n'introduit pas de propriétés absentes des systèmes classiques. Il combine et amplifie des propriétés qui existent individuellement tout en rendant certaines d'entre elles conduites par l'inférence plutôt qu'explicitement programmées.

Les systèmes distribués classiques supportent déjà :

- État mutable (bases de données, caches)
- Routage dynamique (découverte de services, orchestration)
- Actions en cascade (chaînes d'événements, moteurs de workflow)
- Dépendances amont changeantes (gestion de version, changements de topologie)
- Entrées adversariales (la sécurité existe depuis longtemps)

Les agents amplifient ces propriétés et les rendent inférence-conditionnées :

- L'état devient une entrée d'inférence, pas seulement un enregistrement
- Le routage devient dépendant de l'objectif, pas topologiquement fixe
- Les cascades deviennent conduites par le but, pas conduites par l'événement
- Les dépendances deviennent sémantiques, pas structurelles
- Les surfaces d'attaque deviennent semblables à des prompts, pas semblables à des APIs

Cette combinaison crée des exigences distinctes d'observabilité et de contrat.

L'objet d'observabilité change. Les microservices classiques :

requête → calcul → réponse

Les agents :

objectif → observations → inférences → décisions → invocations d'outils → mutations d'état → actions

L'observabilité doit capturer non seulement ce qui s'est passé, mais pourquoi, basé sur quelles preuves, avec quelle autorité, sous quel état. Ceci est catégoriquement différent de « enregistrer tous les appels API ».

Les contrats doivent devenir sémantiques plutôt que simplement syntaxiques. Un contrat d'API classique spécifie les entrées et sorties de manière déterministe. Un agent opère sous un contrat d'exécution composite :

Contrat_exécution = Contrat_technique + Contrat_décision

où :

Contrat_technique = {schéma, protocole, authentification, disponibilité}

Contrat_décision = {objectif, preuves, autorité, exigences_de_confiance, conditions_d'escalade, expiration, conséquences_acceptables}

Ce n'est pas un nouveau principe architectural. C'est une extension : nous étendons le contrat, pas remplaçons l'architecture.

4. La conséquence architecturale, l'espace d'exécution admissible

Les architectures classiques ne prescrivent pas uniformément des chemins déterministes. Les architectures événementielles, les systèmes d'acteurs, les moteurs de workflow, la découverte de services dynamique et les plates-formes d'orchestration (Kubernetes, Kafka, Erlang) supportent des graphes d'exécution dynamiques depuis des décennies.

Ce qui est resté programmiquement déterminé : la topologie et le graphe d'exécution étaient explicitement codés ou configurés.

Les agents introduisent un motif différent : topologie et routage qui sont inférence-conditionnés plutôt que programmiquement déterminés.

Au lieu de :

if condition_X: appeler service_A

if condition_Y: appeler service_B

(explicitement programmé)

un agent pourrait :

étant donné objectif_Z et contexte C, inférer quels services appeler et dans quel ordre

(déterminé par inférence)

C'est nouveau sur le plan opérationnel. Ce n'est pas, cependant, une nouvelle primitive. C'est une nouvelle sémantique d'exécution.

La conséquence architecturale est profonde : au lieu de prescrire le chemin d'exécution, l'architecture doit prescrire l'espace d'exécution admissible. Formalisez ceci :

E = {toutes les trajectoires d'exécution possibles}

E_admissible ⊆ E

Le rôle de l'architecture se décale :

Architecte : définir E_admissible

Agent : sélectionner e_t ∈ E_admissible à l'exécution

Gouvernance : vérifier e_t ∈ E_admissible

Cette distinction est architecturalement propre :

- L'architecture n'a plus besoin de coder chaque chemin
- Le système gagne la flexibilité de s'adapter à l'exécution
- La gouvernance maintient la frontière en vérifiant l'admissibilité

Cela correspond exactement à ce qui permet à l'agent d'opérer au sein d'une ontologie architecturale inchangée tout en changeant la sémantique d'exécution.

5. La conséquence de gouvernance, le risque comme fonction multidimensionnelle

Les principes de gouvernance (identité, responsabilité, audit, escalade) restent invariants. Mais la surface de contrôle requise et l'intensité diffèrent.

L'intensité de gouvernance doit s'ajuster au profil de risque de la décision, pas seulement à la catégorie technologique.

Définissez la charge de gouvernance comme une heuristique multidimensionnelle :

G = f(Autonomie, Conséquence, Irréversibilité, Incertitude, Exposition)

où :

- Autonomie (A) : Le système peut-il décider seul ou doit-il escalader ?
- Conséquence (C) : Quelle est la portée de l'impact en cas d'erreur ?
- Irréversibilité (I) : La décision peut-elle être annulée ?
- Incertitude (U) : Avec quelle confiance l'inférence est-elle faite ?
- Exposition (E) : Combien de fois cette décision est-elle prise ? Exposition = Fréquence × Population × Durée

Cela capture une perspicacité cruciale : un agent prenant une décision réversible à faible conséquence une fois exige une gouvernance minimale. Un agent prenant cette même décision 100 millions de fois a un risque systémique. L'exposition est la variable qui distingue l'erreur isolée de l'erreur industrialisée.

Considérez deux agents avec une autorité identique :

Agent A : Propose un brouillon d'email. (Autonomie=basse, Conséquence=aucune, Irréversibilité=zéro, Incertitude=moyenne, Exposition=basse)

Agent B : Envoie l'email directement. (Autonomie=haute, Conséquence=moyenne, Irréversibilité=haute, Incertitude=moyenne, Exposition=basse)

Agent C : Commande des médicaments à l'hôpital. (Autonomie=haute, Conséquence=critique, Irréversibilité=partielle, Incertitude=basse, Exposition=moyenne-à-haute)

L'intensité de gouvernance diffère dramatiquement. Notez que l'autorité n'est pas la variable de distinction. Le risque l'est.

L'enveloppe d'autonomie devrait refléter cette nature multidimensionnelle :

EA = (Actions, Permissions, Portée_temporelle, Portée_état, Plafond_conséquence, Réversibilité, Conditions_escalade, Plafond_exposition)

Critiquement : l'autonomie n'est pas binaire ; c'est un vecteur d'autorité multidimensionnel borné.

Un système peut avoir :

- Autonomie cognitive haute (peut raisonner sur des entrées complexes)
- Autonomie transactionnelle basse (ne peut exécuter sans approbation)
- Accès à l'information large (peut lire de nombreuses bases de données)
- Plafond de conséquence étroit (ne peut affecter les enregistrements principaux)
- Portée temporelle courte (les décisions sont valides pendant des minutes, pas des jours)

C'est une représentation plus précise de l'autonomie qu'un simple oui/non.

Exemple de PREDICARE SafetyTriage :

- Actions : Classifier une notification de sécurité
- Permissions : Lire la base de données de sécurité, interroger les cas historiques
- Temporelle : Immédiate (pas de mise en queue)
- État : Cas unique (pas de raisonnement inter-cas)
- Conséquence : Peut recommander une escalade, ne peut pas l'appliquer
- Réversibilité : Entièrement réversible (la recommandation peut être annulée)

- Escalade : Si confiance < 0.70, escalader vers humain
- Exposition : ~100 notifications par jour

Exemple de recommandation de médicaments (avec frontières claires) :

Le système peut :

- Recommander des médicaments (portée d'autorité : accès en lecture seule au dossier patient, suggérer seulement)
- PAS commander des médicaments (cela nécessite l'autorité du prescripteur)
- PAS administrer des médicaments (cela nécessite le personnel clinique)

Pourquoi ces frontières ? Parce que chaque étape a une irréversibilité différente :

Recommander → Prescrire → Commander → Dispenser → Administrer

sont cinq autorités distinctes avec conséquence cumulative.

- Un agent autonome pourrait avoir autorité sur Recommander et Lire.
- Un prescripteur a Prescrire.
- Un pharmacien a Dispenser.
- Le personnel clinique (infirmier) a Administrer.

Chaque frontière existe parce que l'irréversibilité et la conséquence diffèrent.

Cela démontre pourquoi les frontières de capacité bornée sont importantes : elles ne sont pas arbitraires. Elles correspondent aux seuils d'irréversibilité et d'autorité.

6. La question durable : la substitutabilité d'implémentation

Enfin, ce cadre active une question architecturale durable :

Quelles capacités peuvent changer de modalité d'implémentation de manière sûre ?

La substitutabilité d'implémentation n'est pas automatique. Deux implémentations sont substitutables pour une capacité bornée seulement si toutes deux satisfont :

$I_a \equiv_{CB} I_b$ si :

$Résultat(I_a) \approx Résultat(I_b)$ ET

$Risque(I_a) \leq Risque\_acceptable$ ET

$Propriétés\_opérationnelles(I_a) \subseteq SLA(CB)$ ET

$Coût(I_a) \in Budget(CB)$

En d'autres mots, elles sont substitutables seulement si elles satisfont le contrat d'admissibilité pour cette capacité bornée.

Cela permet l'évolution pratique :

- Aujourd'hui : Capacité → Humain
- Demain : Capacité → Humain + Copilot
- Plus tard : Capacité → Agent + Approbation humaine
- Éventuellement : Capacité → Agent autonome (si le profil de risque le permet)
- Possiblement : Capacité → Hybride (Cascade d'agents + Escalade)

Le contrat de capacité bornée persiste ; la modalité d'implémentation évolue. Le modèle de capacité reste stable sans nécessiter la reconstruction du modèle de capacité à chaque changement d'implémentation.

7. Conclusion

Les principes et primitives architecturaux restent inchangés. La sémantique d'exécution a évolué. La surface de contrôle et l'intensité de gouvernance requise diffèrent.

L'agent n'exige pas une nouvelle architecture. Il exige de comprendre que l'architecture gouverne maintenant un espace d'exécution admissible plutôt qu'un chemin déterministe, et que l'intensité de gouvernance suit le profil de risque multidimensionnel, pas la catégorie d'implémentation.

Cet insight résout la tension entre l'autonomie de l'agent et la discipline architecturale. Elles ne sont pas en conflit lorsque nous reconnaissons que :

- La capacité est la primitive (pas l'agent, pas l'implémentation)
- La capacité bornée est l'unité d'autorité (pas l'organisation, pas le système)
- L'autorité est multidimensionnelle (pas binaire, pas catégorique)
- L'espace d'exécution admissible est le mécanisme de gouvernance (pas la prescription de chemin, pas les règles détaillées)
- La substitutabilité d'implémentation est l'objectif (pas le déploiement d'agent, pas l'adoption d'IA)