Un rôle n'est pas une capacité

Pourquoi une architecture agentique doit justifier séparément la capacité qu'elle mobilise, l'autonomie qu'elle accorde et l'autorité qu'elle refuse

Introduction : ce que l'on décide en nommant une persona

Dans l'émulation d'un essai comparant les CAR-T au traitement standard dans les lymphomes agressifs, le choix des variables d'ajustement repose d'abord sur une représentation causale du processus clinique. L'émulation d'essai cible consiste à expliciter l'essai randomisé que l'on aurait voulu conduire, puis à l'imiter avec des données observationnelles (Hernán et Robins, 2016). Construire cette représentation relève du jugement scientifique : relations supposées entre variables, confusion non mesurée, mécanismes de sélection, définition du temps zéro, sans laquelle le délai entre éligibilité et administration effective du traitement suffit à produire un biais de temps immortel (Hernán et al., 2016). Une fois le graphe causal explicité et ses hypothèses examinées, une partie du travail change de nature. Des algorithmes identifient les ensembles d'ajustement qui satisfont les critères graphiques et vérifient les contraintes temporelles spécifiées, de façon reproductible et indépendamment de la formulation de la question. Ils ne valident ni la justesse du graphe, ni l'identifiabilité de l'effet dans les données, ni la pertinence statistique de l'estimateur, et ils ne choisissent pas entre plusieurs ensembles admissibles dont les propriétés en variance diffèrent.

Une pratique répandue consisterait à confier l'ensemble à un agent « épidémiologiste », doté d'une persona crédible et d'un droit de parole dans une délibération entre agents. Elle confondrait trois choses que l'exemple sépare : le jugement qui construit le problème, le calcul qui en traite la partie mécaniquement contrôlable, la validation qui porte sur les deux. Dans cette pratique, que l'on appellera *agentification par persona*, nommer une fonction revient à lui attribuer une instance de LLM, un rôle exprimé par prompt et une interface conversationnelle, avant même d'avoir établi si cette fonction exige un traitement génératif.

C'est cette pratique que vise l'article. Il ne conteste ni le principe de l'agent, qui peut reposer sur un solveur, un moteur de règles ou une combinaison de composants, ni celui d'une architecture multi-agent. Plusieurs agents spécialisés peuvent apporter des capacités, des droits ou des contextes distincts qui justifient leur séparation. Cette justification n'exonère pas de mesurer leurs coûts de coordination et de contrôler leurs

effets. Une seconde erreur accompagne souvent la première sans lui être équivalente : laisser au composant qui propose une action une autorité excessive sur son exécution. Une architecture à un seul LLM outillé peut présenter exactement les mêmes défauts de contrôle qu'une architecture à dix personas ; une architecture découpée par rôles peut disposer d'un contrôle solide. La première erreur motive l'article ; la seconde explique pourquoi sa correction, la spécialisation du calcul, doit s'accompagner d'une séparation de l'autorité qu'elle ne garantit pas.

La thèse tient en trois propositions. *Un rôle n'est pas une capacité* : chaque fonction doit être confiée au composant dont les propriétés évaluées satisfont ses exigences. *Une capacité ne justifie pas nécessairement un agent* : l'autonomie accordée à un composant doit être justifiée par la valeur qu'elle apporte. *Un agent ne détient pas automatiquement l'autorité sur ses effets* : les effets externes doivent être soumis à des mécanismes d'autorisation, d'allocation et de preuve indépendants du composant qui les propose, selon leur criticité et les garanties requises.

Ces trois propositions répondent à une seule question : à quel composant confier une fonction, à quel moment de son cycle de vie, et sous quelle autorité ? L'article défend un critère de conception et de justification. Il ne prétend pas que les architectures hybrides sont systématiquement moins coûteuses, que les composants déterministes sont toujours plus fiables, ni que le multi-agent est intrinsèquement moins performant. Il porte sur les systèmes agentiques dotés de capacités d'action sur des ressources, des données ou des systèmes externes, dans des environnements où la preuve a un coût et une valeur, au premier rang desquels la santé, la pharmacie et la défense.

Quelques termes suffisent. Un *agent* est un composant doté d'un objectif, d'un accès à des outils et d'une boucle de décision sur ses propres actions ; son *autonomie* est l'étendue des décisions que cette boucle prend sans validation extérieure. Un *rôle* est une spécification comportementale attribuée par prompt. Une *capacité* est une aptitude effective à réaliser une fonction, dont les propriétés et les limites sont évaluées sur un domaine d'emploi défini ; un rôle peut contribuer à cette aptitude, il n'en constitue ni la preuve ni la garantie. La *frontière d'autorité* sépare des responsabilités, des droits ou des compétences ; l'*unité de calcul* découpe un traitement selon la nature de l'opération. Une contrainte est *mécaniquement contrôlable* lorsqu'un mécanisme spécifié peut, à partir des informations accessibles, déterminer si elle est satisfaite, violée ou non évaluable ; seule la satisfaction permet de conclure positivement, et seulement si le niveau de preuve exigé est atteint. Une contrainte est *ouverte* lorsque l'on ne sait pas, aujourd'hui, spécifier un tel mécanisme, parce que son évaluation exige un jugement. Un *effet externe* est toute action qui consomme une ressource, divulgue une donnée ou modifie un état hors du contexte de raisonnement.

1. Pourquoi l'on découpe par personas

L'agentification par persona est pratiquée par des équipes compétentes, et il faut en rendre compte avant de la critiquer. Ce qui suit décrit des mécanismes plausibles ; aucune donnée ne permet à ce stade d'en établir la fréquence ni de les hiérarchiser.

Le premier mécanisme est organisationnel. Conway observait que les systèmes tendent à reproduire la structure de communication des organisations qui les conçoivent (Conway, 1968). Un agent « finance », un agent « conformité » et un agent « risques » reproduisent un organigramme, et avec lui des lignes de responsabilité et de budget lisibles. Le deuxième est cognitif : un rôle est une interface de compréhension, qui permet d'expliquer le système, de localiser un défaut, de répartir la maintenance. Ce bénéfice porte sur la représentation du système, non sur ses propriétés ; un organigramme d'agents lisible peut masquer un calcul illisible. Le troisième est outillé : certains frameworks définissent leurs agents par un rôle, un objectif et une histoire personnelle, le triptyque *role, goal, backstory* de CrewAI en étant l'exemple le plus explicite. Quand l'outil demande une persona avant un contrat d'interface, la persona devient une unité de conception par défaut. Le quatrième est protocolaire : le protocole A2A, lancé par Google en avril 2025 puis confié à la Linux Foundation, vise à permettre à des agents de découvrir leurs capacités respectives, d'échanger des informations et de coordonner des tâches. Lorsque l'interopérabilité se négocie d'agent à agent, envelopper une fonction dans un agent devient un moyen de la rendre adressable.

Il faut concéder ce que le rôle fait bien. En environnement régulé, la séparation entre production et validation est une exigence ; le principe des quatre yeux existe parce que celui qui produit ne doit pas être celui qui approuve. Un rôle qui matérialise une telle frontière a une valeur architecturale réelle. Le problème apparaît lorsque le rôle sert à découper le calcul sans correspondre à aucune frontière d'autorité, de droits ou de contexte. La persona devient alors *un artefact de gouvernance déguisé en artefact de calcul* : elle rassure sur qui répond de quoi, sans rien garantir sur ce qui est calculé.

2. Un rôle n'est pas une capacité

Le premier coût de la confusion est d'interprétation. Anthropic rapporte qu'un système composé d'un orchestrateur Claude Opus 4 et de sous-agents Claude Sonnet 4 surpasse de 90,2 % l'agent unique Opus 4 sur une évaluation interne de recherche documentaire, et que, sur le benchmark BrowseComp, trois facteurs expliquent 95 % de la variance de performance, la consommation de jetons en expliquant à elle seule 80 % ; ces systèmes consomment environ quinze fois plus de jetons qu'une conversation (Anthropic, 2025). Ces résultats montrent qu'une configuration multi-agent peut améliorer la performance sur certaines tâches de recherche en mobilisant davantage de calcul.

Il n'établit pas de relation causale : une décomposition de variance reste une association, et l'évaluation est interne. La conclusion prudente suffit : les gains attribués au multi-agent doivent être interprétés au regard des ressources supplémentaires mobilisées, et ne peuvent être attribués automatiquement à la décomposition en rôles.

Le deuxième coût est comptable, et il ne se lit pas sur un compteur de jetons. Chaque agent reconstruit une partie du contexte que d'autres possèdent déjà ; chaque transfert produit une sortie intermédiaire qu'il faut générer puis relire ; chaque étape séquentielle ajoute sa latence ; chaque vérification croisée consomme autant qu'une génération ; l'observabilité et le débogage d'une délibération multi-agent ont un coût que la facture d'inférence ne montre pas. En sens inverse, un composant spécialisé a des coûts de conception, d'intégration et de maintenance, et un LLM généraliste peut être la solution la moins chère pour une fonction rare, ambiguë ou changeante. La mesure pertinente est le coût complet par tâche correctement accomplie, reprises, contrôle et validation compris. Elle ne suffit pas seule : deux architectures peuvent ne pas avoir la même couverture fonctionnelle, le même taux d'abstention ni la même exposition aux effets interdits, et celle qui refuse correctement une tâche paraîtra plus chère que celle qui l'exécute à tort. La comparaison doit tenir séparés la qualité des tâches admissibles, la couverture et le taux d'abstention, le respect des contraintes, la latence et le coût complet. Le mot de *gabegie* n'est justifié que dans un cas démontré : lorsqu'une architecture mobilise des ressources supplémentaires sans apporter de propriété qui en justifie le coût.

Le troisième coût tient au format d'échange. Lorsque des composants échangent leurs résultats principalement sous forme de prose, ils imposent à leurs destinataires un nouveau travail d'interprétation : un montant calculé devient une phrase qu'il faut relire pour en extraire un montant, une incertitude devient un adverbe. Des contrats d'interface typés réduisent cette ambiguïté et rendent détectables certaines violations de structure, sans garantir la vérité, la complétude ou la pertinence de ce qui est transmis. L'enjeu n'est pas de bannir le langage naturel, mais de ne pas l'utiliser comme substitut implicite à un contrat d'interface.

Le quatrième coût est le plus structurel. Cemri et al. analysent sept frameworks multi-agents et établissent une taxonomie de quatorze modes de défaillance répartis en trois catégories : spécification et conception, désalignements entre agents, vérification (Cemri et al., 2025). Ils observent que les gains de ces systèmes restent souvent minimes face aux agents uniques sur les benchmarks courants. Ce résultat ne fournit pas de taux transposable à une architecture régulée ; il indique où regarder, la vérification. Une seconde instance du même modèle peut détecter certaines erreurs, notamment en suivant un raisonnement différent ; elle ne constitue pas, par sa seule existence, une vérification indépendante. Deux instances qui partagent un modèle fondation partagent ses biais et ses vulnérabilités ; leur accord ne prouve pas leur justesse. La sûreté de

fonctionnement connaît ce problème sous le nom de défaillance de cause commune : une redondance homogène protège mal contre ce qui affecte également ses branches.

On a passé une décennie à apprendre à ne pas découper un monolithe en cent services bavards. On s'apprête à le refaire avec des services qui facturent leur bavardage au jeton.

3. Allouer par propriétés, et au bon moment

Chaque fonction est caractérisée par un profil d'exigences, puis confiée à une classe de composant dont les propriétés évaluées couvrent ce profil. *Le rôle définit qui répond d'une décision ; la capacité définit ce que l'on peut en garantir.*

L'allocation procède en trois étapes. La première caractérise la fonction : entrées, sorties, variabilité, degré de spécification des règles, disponibilité et fiabilité des données, niveau de preuve exigé, risques, effets externes. La deuxième identifie les solutions admissibles, c'est-à-dire toutes celles qui satisfont les exigences éliminatoires, quelle que soit leur nature : fonction déterministe, solveur, moteur de règles, modèle spécialisé, LLM, architecture hybride, artefact généré puis validé. La troisième arbitre entre elles. Le coût complet y est un critère central, mais certaines propriétés ne se réduisent pas à des seuils : évolutivité, réversibilité, dépendance à un fournisseur, facilité d'investigation après incident. Un score composite masquerait ces choix derrière des pondérations arbitraires ; il vaut mieux documenter l'arbitrage, ses hypothèses de volume et de changement, les compromis acceptés, son responsable et les conditions de son réexamen. *Un arbitrage non documenté n'est pas une allocation ; c'est une habitude.*

Deux décisions doivent rester distinctes : la *qualification fonctionnelle*, qui demande si la générativité apporte une capacité nécessaire ou utile, et l'*arbitrage*, qui demande laquelle des solutions admissibles convient le mieux. La structure des données ne tranche pas la première. Une planification à entrées et sorties typées peut exiger un raisonnement génératif lorsque le travail consiste à interpréter une demande ou à proposer une formalisation. Un solveur convient quand le problème est posé ; un LLM peut convenir quand il reste à poser. Et un LLM peut être retenu pour une fonction qui ne l'exige pas, si l'arbitrage lui est favorable : l'allocation par propriétés n'est pas un dogme anti-génératif, c'est une discipline de justification.

Les décisions qui comptent en environnement régulé portent souvent sur des contraintes partiellement contrôlables. Une contre-indication médicamenteuse contextuelle contient une partie mécaniquement contrôlable (l'interaction répertoriée entre deux substances, le seuil de fonction rénale au-delà duquel une posologie doit être adaptée) et une partie ouverte (l'arbitrage bénéfice-risque chez un patient donné). Le seuil est parfaitement spécifié ; mais si la valeur biologique disponible est ancienne ou non représentative, le contrôle doit conclure « non évaluable » et renvoyer le cas. Un contrôle qui ne sait conclure qu'en vrai ou faux transformerait l'absence de donnée en autorisation

implicite. Confier la contrainte entière à un LLM laisserait la partie contrôlable dépendre d'un composant probabiliste ; la confier entière à un moteur de règles reviendrait à prétendre avoir spécifié le jugement. Les deux erreurs sont symétriques, et la seconde est la plus fréquente chez ceux qui découvrent les vertus du déterminisme.

La charge de la preuve croît avec la criticité de la fonction. Mais l'unité de preuve n'est pas la classe de composant : c'est la propriété revendiquée. Un moteur de règles peut appliquer exactement une règle erronée ; un solveur peut garantir l'optimalité d'un problème mal formulé. À l'inverse, certaines propriétés d'un composant probabiliste peuvent être garanties par son enveloppe d'exécution : permissions, bornes de ressources, formats acceptés, absence d'accès direct à une ressource protégée. *On ne prouve pas un composant ; on prouve une propriété d'un composant, sous des hypothèses*.

Le cycle de vie ajoute une dimension que l'allocation instantanée ignore. Une capacité se conçoit, se valide, s'exécute, et chaque étape peut appeler un composant différent. Le LLM peut générer une règle, une requête, un workflow ou un programme ; cet artefact est validé, puis exécuté sans LLM. *Concevoir avec un LLM n'oblige pas à exécuter avec un LLM*. L'intérêt de cette configuration doit être énoncé avec précision, car un artefact figé procure trois bénéfices distincts dont aucun n'implique les autres : la stabilité d'un objet versionné, la possibilité de l'inspecter, et la faisabilité d'en établir les propriétés requises. Une règle déclarative courte et un programme généré de plusieurs milliers de lignes sont tous deux figés ; leurs surfaces de validation n'ont rien de comparable. Le cas favorable est celui d'un artefact dont le domaine d'emploi, les dépendances et les propriétés attendues peuvent être bornés et qualifiés à un coût acceptable. Et lorsque l'artefact échoue, cet échec ne doit pas autoriser automatiquement une exécution générative de substitution : dans une fonction critique, ce recours est lui-même une décision d'allocation, à qualifier et à contrôler, sans quoi le chemin de secours devient la voie par laquelle la générativité non validée revient en production.

La caractérisation d'une fonction n'est pas fixe. Une pratique experte se codifie, une donnée devient disponible, un modèle progresse ; le prix de l'inférence baisse, les exigences de preuve montent. Une doctrine qui fixerait une frontière serait fausse dans deux ans. Celle-ci fixe un critère, qui s'applique à une frontière mobile.

4. Une capacité ne justifie pas un agent

Le découpage d'un système en composants distincts peut être motivé par six propriétés recherchées : le parallélisme, lorsque la tâche se décompose en branches indépendantes et que la latence compte ; l'isolation de contexte, lorsqu'un contexte unique serait saturé ou pollué ; la séparation des droits, qui limite l'étendue des dommages qu'un détournement peut causer ; la diversité de vérification, lorsque les erreurs des vérificateurs ne sont pas corrélées ; les frontières d'autorité, réglementaires

ou inter-organisationnelles ; l'exploration indépendante de stratégies différentes dans un même espace de recherche. Ce sont des motifs possibles, non des justifications suffisantes, et la liste n'est pas exhaustive. Une propriété recherchée ne justifie une séparation que si celle-ci l'apporte effectivement, à un coût acceptable : le parallélisme peut accroître la contention, l'isolation peut dégrader le partage d'informations, la multiplication des frontières complique la coordination. L'exploration de stratégies, en particulier, est plausible mais reste à démontrer à ressources comparables.

Surtout, ces motifs justifient une séparation, pas encore un agent. Une branche de calcul parallèle peut être une fonction ; une frontière de droits peut être matérialisée par un service classique ; une vérification indépendante peut être un test déterministe. Deux décisions successives doivent être justifiées : pourquoi séparer cette fonction, puis pourquoi le composant séparé doit disposer d'une autonomie de décision. La seconde se justifie lorsque la capacité du composant à choisir et réviser ses actions apporte, par rapport aux solutions admissibles sans cette autonomie, une propriété ou un gain qui en couvre les coûts et les risques. *Une séparation se justifie par la propriété qu'elle apporte ; l'autonomie, par la valeur qu'elle démontre*. Un agent qui ne sépare rien et dont l'autonomie n'apporte rien n'est qu'un prompt supplémentaire avec une identité.

L'orchestration est l'endroit où cette distinction produit ses effets les plus coûteux, parce que l'orchestrateur cumule souvent quatre responsabilités : proposer un plan, autoriser ses étapes, allouer les ressources, faire exécuter. Un processus connu se pilote par un workflow ou une machine à états, le LLM n'intervenant qu'aux points qui exigent une interprétation. Un orchestrateur génératif reste pertinent pour les tâches ouvertes, à condition de borner ce qu'il décide : il choisit parmi des capacités autorisées sans créer de droits, et son plan est une proposition soumise au contrôle. *Un plan n'est pas une autorisation*. L'état de la tâche (ce qui a été fait, autorisé, consommé) ne doit pas résider dans le seul contexte du modèle, qui se tronque, se résume et se perd ; il doit être externalisé dans une structure explicite que le LLM consulte sans en être le dépositaire. Les conditions d'arrêt, enfin, doivent être imposées de l'extérieur. L'autonomie d'un agent se mesure à ce qu'il peut décider ; sa sûreté, à ce qu'il ne peut pas empêcher.

Encapsuler chaque agent dans un microservice ramène l'agentique à des questions classiques d'architecture distribuée ; cela règle la question du déploiement, non celle du nombre, ni celle de savoir ce qui mérite d'être un agent. C'est précisément parce qu'elle banalise l'agent que cette encapsulation rend visibles les erreurs de granularité que l'architecture distribuée a mis quinze ans à documenter, et les réponses qu'elle y a apportées : contrats d'interface versionnés plutôt que conversations, compensations plutôt que reprises aveugles, découpage par capacité plutôt que par organigramme.

5. Un agent ne détient pas l'autorité sur ses effets

Cette section traite la seconde erreur, qui existe avec ou sans persona. Elle énonce des principes ; leur mise en œuvre détaillée relève d'un traitement technique distinct.

Un LLM traite une demande de virement et produit un objet structuré contenant un montant et un champ d'approbation positionné à vrai. Un moteur de règles vérifie que le montant est sous le plafond et que l'approbation est vraie. Le contrôle est déterministe, il fonctionne comme spécifié, et il autorise une opération qui n'aurait jamais dû l'être. Le montant n'est pas en cause : l'agent peut le proposer, un plafond peut le borner. L'approbation l'est, parce qu'une assertion du contrôlé, non attestée, a été acceptée comme preuve. *Un contrôle déterministe peut être captif*. Il faut distinguer les paramètres de l'action, que l'agent peut produire ; les attributs d'autorité (identité, droits, approbations), qui doivent être attestés par des sources indépendantes ; les faits métier, qui peuvent exiger une vérification externe ; les politiques, que l'agent ne doit pouvoir ni modifier ni négocier. Pour les effets concernés, le contrôle doit rester hors de l'autorité du composant contrôlé, couvrir les chemins d'exécution identifiés, vérifier les attributs d'autorité auprès de sources appropriées et produire des traces dont l'intégrité peut être établie, selon le modèle de menace retenu. Ces exigences s'inscrivent dans les principes classiques de médiation complète et de séparation des privilèges (Saltzer et Schroeder, 1975). L'indépendance a cependant une limite : une source attestée peut être erronée, une approbation authentique peut avoir été donnée sur la foi d'informations fausses. L'indépendance réduit la manipulation ; elle ne garantit pas la justesse.

Un agent qui lit un document contenant une instruction malveillante peut émettre un appel d'envoi de courriel parfaitement typé, vers un destinataire externe, avec une pièce confidentielle, dans le périmètre exact de ses permissions. Schéma respecté, droits respectés, effet illégitime : c'est le *confused deputy* décrit par Hardy (1988), dont l'injection de prompt est la version contemporaine. La difficulté est plus profonde qu'il n'y paraît. Si l'utilisateur a autorisé l'envoi d'un compte rendu à un partenaire externe, le contrôleur ne peut pas déduire des seuls paramètres de l'envoi quelle partie du contenu procède de la demande et quelle partie procède de l'injection. *Autoriser un type d'action n'est pas autoriser son contenu*. Pour les effets critiques, l'autorisation doit être liée à un objet précis : tel contenu, tel destinataire, ou une représentation vérifiable comme l'empreinte du document approuvé. Même alors, trois garanties restent distinctes : l'identité de l'objet exécuté, la légitimité de l'approbation, la conformité substantielle de l'objet à l'intention et aux contraintes. Les deux premières s'attestent souvent mécaniquement ; la troisième peut rester ouverte. *Un contrôle indépendant peut garantir l'application d'une décision sans garantir la justesse de cette décision*.

Le risque propre aux LLM est le blanchiment d'instructions : une instruction contenue dans une donnée sans autorité est reformulée, puis réintroduite dans le système comme instruction légitime, par exemple sous la forme d'une étape du plan proposé. Une sortie

générative peut légitimement devenir une instruction, lorsqu'un responsable humain ou un service métier l'examine et la valide ; le problème n'est pas l'élévation d'autorité, mais l'élévation sans acte d'autorisation distinct, attribuable et journalisé. *L'autorité vient de l'acte qui valide, non du texte qui propose.* Le même principe vaut pour la confidentialité : une synthèse peut révéler une information protégée sans la citer, et la déclassification d'une sortie exige une validation portant sur l'information produite, non un simple réétiquetage. CaMeL (Debenedetti et al., 2025) instancie la séparation entre flux de contrôle et données non fiables : l'intention est extraite de la requête de confiance, un interpréteur trace la provenance des valeurs et applique des politiques aux appels d'outils. Les auteurs rapportent 77 % de tâches AgentDojo résolues avec une sécurité qu'ils qualifient de prouvable, contre 84 % sans défense. Ce résultat porte sur leur modèle de sécurité et sur un benchmark ; il ne garantit pas la sécurité d'un système déployé, mais il illustre une différence de nature entre une contrainte imposée par l'architecture et une résistance améliorée du modèle.

Le contrôle ne vaut pas seulement au dernier appel d'outil. Une sortie d'agent n'est pas une action, mais elle peut devenir l'entrée décisionnelle d'un autre agent ; toute sortie doit donc porter un statut explicite (proposition, information ou action autorisée). La délégation obéit à la même logique, à condition de distinguer deux opérations. La *délégation d'autorité* transmet une partie des droits du délégant et ne doit jamais en créer. La *demande d'exécution auprès d'une autorité distincte* sollicite un composant qui dispose de ses propres droits : un service de paiement exécute un virement que l'agent demandeur n'a pas le droit de réaliser, après avoir vérifié selon sa propre politique que la demande est autorisée.

Deux distinctions complètent ces principes. *Une action permise n'est pas une action financée* : un agent autorisé peut engager des ressources qui ne le méritent pas, et plusieurs agents autorisés peuvent ensemble dépasser un budget partagé ; l'allocation des ressources relève d'un mécanisme propre, indépendant des agents. *Une autorisation n'est pas une transaction* : entre l'autorisation accordée, la commande émise, la commande acceptée et l'effet confirmé, le dernier événement peut rester inconnu, et une nouvelle tentative aveugle n'est pas une preuve d'échec de la précédente ; c'est exactement ainsi que l'on paie deux fois. L'architecture doit prévoir un état indéterminé et une politique explicite pour le traiter. *La gouvernance ne consiste pas à produire un journal rassurant, mais à rendre visibles les situations où la preuve manque.*

Restent les contraintes ouvertes, que ces mécanismes ne tranchent pas. La bonne question n'est pas « quel vérificateur ? » mais « sur quelle base vérifier ? ». D'abord, ce qui peut être confronté à une référence externe : donnée source, calcul indépendant, règle applicable, jugement expert documenté ; une synthèse clinique n'est pas mécaniquement contrôlable, la présence dans le dossier source des éléments qu'elle cite l'est souvent. Ensuite, ce qui ne peut qu'être soumis à un second jugement, en

sachant que la diversité des modèles ne remplace pas une référence externe et signale au mieux qu'elle est nécessaire. Enfin, ce qui impose l'abstention faute de preuve suffisante, à condition que le signal d'incertitude soit calibré sur un domaine défini. Le contrôle humain trouve là sa place : proportionné au risque, ciblé et suffisamment informé pour permettre un jugement effectif, il porte notamment sur les contraintes ouvertes à fort enjeu et sur la justesse des règles, plutôt que sur la répétition manuelle de contrôles mécaniques déjà qualifiés. Il suppose des humains exercés à juger ; sans quoi le contrôle mécanique devient le dernier dépositaire d'une expertise que plus personne ne sait vérifier.

6. Ce que l'exemple d'ouverture montre, et ce qu'il ne montre pas

Dans une émulation d'essai de ce type, trois familles d'opérations pourraient être confiées à une persona LLM : la vérification de la cohérence des données au regard d'une ontologie des variables, l'identification des ensembles d'ajustement sur le graphe causal, la vérification de l'admissibilité temporelle des variables. Toutes trois peuvent être confiées à des mécanismes exécutables. La cohérence des données relève typiquement d'une validation de formes, dont le résultat dépend des règles déclarées et non de la formulation d'une question ; les critères graphiques et l'admissibilité temporelle se vérifient par des procédures reproductibles, appliquées après la révision clinique du graphe. Ces vérifications identifient des ensembles d'ajustement admissibles relativement au graphe ; elles ne garantissent ni la qualité du graphe, ni l'identifiabilité de l'effet dans les données. L'exécutabilité du contrôle ne vaut pas validation de l'inférence causale. La frontière entre ce qui relève du jugement des cliniciens et ce qui relève du calcul est exactement celle que cet article propose de rendre explicite.

Conclusion : capacité, autonomie, autorité

Cet article propose un critère de conception dont certaines conséquences peuvent être garanties ; il ne démontre pas qu'une architecture qui l'applique est globalement préférable. Cette seconde question est expérimentale. Son programme principal découle directement de la thèse : à fonction, exigences et ressources comparables, quelle valeur apporte la substitution d'une capacité spécialisée à une persona LLM, et quelle valeur supplémentaire apporte l'autonomie ? Y répondre exige de définir, pour chaque comparaison, la population de tâches, le comparateur (dont un LLM unique doté des mêmes outils, comparateur nécessaire pour aider à distinguer l'effet de la spécialisation du calcul de celui de la multiplication des instances), l'estimand, l'horizon, le critère principal, et ce que signifie un budget égal lorsque jetons, temps de calcul, coût monétaire et énergie ne sont pas interchangeables. Les questions d'indépendance des

contrôles, de garanties budgétaires, de gestion des changements et de coût de validation en sont des programmes dérivés.

La thèse a des limites. La caractérisation d'une fonction évolue, et avec elle l'allocation qui en découle. La séparation des composants, qui facilite l'établissement de certaines preuves, multiplie aussi les interfaces et les éléments à qualifier ; son bilan n'est pas acquis d'avance. Les mécanismes d'attestation et de liaison à l'intention ont un coût d'ingénierie et d'utilité qui peut, dans certains usages, excéder la valeur des garanties apportées. Le traitement des contraintes ouvertes repose sur des dispositifs moins établis que les contrôles mécaniques. Et les mécanismes proposés en section 1 pour expliquer l'agentification par persona restent des hypothèses.

La proposition la plus solide n'est pas que le déterminisme coûte moins cher qu'un LLM : cet équilibre dépend des prix, des volumes et des modèles, et il se déplacera. C'est qu'une architecture agentique doit justifier séparément trois choses qu'une persona confond :

- La capacité qu'elle mobilise, évaluée sur un domaine d'emploi défini,
- L'autonomie qu'elle accorde, qui doit démontrer sa valeur,
- L'autorité qu'elle refuse au composant qui propose.

Cette séparation reste pertinente lorsque les modèles deviennent moins coûteux, plus performants ou plus fiables, parce qu'elle ne porte pas sur ce qu'ils savent faire, mais sur ce qu'on les laisse décider.

Le critère se vérifie sur toute architecture existante, en trois questions. Pour chaque composant :

- Quelle capacité apporte-t-il, et sur quel domaine d'emploi a-t-elle été évaluée ?
- Pour chaque boucle de décision autonome : quelle valeur démontrée justifie cette autonomie ?
- Pour chaque effet externe : quelle part des règles, des preuves, des ressources et des mécanismes qui l'autorisent reste hors de portée du composant qui le propose ?

Une architecture qui ne sait pas répondre à ces trois questions n'a pas seulement un problème de nombre d'agents. Elle n'a pas établi ce qu'elle calcule, pourquoi elle délègue et ce qu'elle autorise.

Références

Anthropic. « How we built our multi-agent research system ». Anthropic Engineering, juin 2025. https://www.anthropic.com/engineering/multi-agent-research-system

Cemri M., Pan M. Z., Yang S. et al. « Why Do Multi-Agent LLM Systems Fail? ». NeurIPS 2025, Datasets and Benchmarks Track. arXiv:2503.13657.

Conway M. E. « How Do Committees Invent? ». *Datamation*, 14(4), avril 1968, p. 28-31.

CrewAI. Documentation, définition des agents (attributs *role*, *goal*, *backstory*). https://docs.crewai.com

Debenedetti E., Shumailov I., Fan T. et al. « Defeating Prompt Injections by Design ». arXiv:2503.18813, v2, juin 2025.

Hardy N. « The Confused Deputy (or why capabilities might have been invented) ». *ACM SIGOPS Operating Systems Review*, 22(4), 1988, p. 36-38.

Hernán M. A., Robins J. M. « Using Big Data to Emulate a Target Trial When a Randomized Trial Is Not Available ». *American Journal of Epidemiology*, 183(8), 2016, p. 758-764. doi:10.1093/aje/kwv254.

Hernán M. A., Sauer B. C., Hernández-Díaz S., Platt R., Shrier I. « Specifying a target trial prevents immortal time bias and other self-inflicted injuries in observational analyses ». *Journal of Clinical Epidemiology*, 79, 2016, p. 70-75.

Linux Foundation. Annonce de la création du projet Agent2Agent (A2A), juin 2025 (protocole initié par Google en avril 2025).

Saltzer J. H., Schroeder M. D. « The Protection of Information in Computer Systems ». *Proceedings of the IEEE*, 63(9), 1975, p. 1278-1308.