Le SaaS commodifié : où va la valeur dans la transition agentique

Pourquoi la transition agentique pourrait commodifier une partie du SaaS horizontal et déplacer la valeur vers l'orchestration, l'infrastructure de délégation et les capacités métier profondes.

2026-05-17 · Perspectives · SaaS · Agents IA · Produit · Orchestration

Résumé

La transition agentique fragilise le modèle SaaS horizontal en déplaçant l'usage de la manipulation directe vers la délégation gouvernée.

La valeur pourrait remonter vers les couches d'orchestration qui captent l'intention, et descendre vers les capacités métier profondes ou l'infrastructure de délégation.

Pour les éditeurs SaaS, l'enjeu n'est pas seulement d'ajouter de l'IA, mais de choisir explicitement leur position dans une chaîne de valeur en recomposition.

Sommaire

Contexte de lecture

Audience

Cet article s'adresse aux dirigeants, équipes produit, éditeurs SaaS et responsables innovation qui veulent anticiper les effets stratégiques des agents IA sur le logiciel B2B.

Ce que cet article couvre

Il couvre la commodification possible du SaaS horizontal, les dynamiques agentiques, la couche d'orchestration, le pricing, la gouvernance, les modes de défaillance et les décisions produit à prendre.

Ce que cet article ne couvre pas

Il ne propose pas une prédiction certaine, un benchmark de solutions, ni une recette universelle applicable à tous les éditeurs SaaS.

Le présupposé qui se fissure

Pendant vingt ans, le SaaS a reposé sur un présupposé tellement évident qu'il n'a presque jamais été formulé : **l'utilisateur principal du logiciel est un humain**. Tout ce qui en découle (parcours, écrans, dashboards, notifications, modèles de pricing à l'utilisateur, métriques d'adoption, démarches de support) suppose qu'à l'autre bout du logiciel se trouve un humain qui observe, comprend, clique et valide.

Ce présupposé ne s'effondre pas. Il se relativise. Une part croissante de l'usage du logiciel passe désormais par des systèmes capables d'agir au nom de l'utilisateur : lire des documents, remplir des champs, appeler des API, rapprocher des données, préparer des décisions. Le centre de gravité se déplace de la manipulation directe vers la délégation d'exécution.

La formule « Service as Software » ainsi que l'idée de « vendre le travail plutôt que le logiciel » désigne cette inversion. Elle s'est imposée dans le débat VC à partir de 2023-2024, notamment via Sarah Tavel puis Foundation Capital. Elle n'est pas neuve, mais elle est utile : elle force à se demander si l'on vend un outil que l'utilisateur manipule ou une capacité d'exécution qui rend un service à la place de l'utilisateur.

Cette question paraît rhétorique. Elle ne l'est pas. Elle gouverne la manière dont on conçoit le produit, comment on facture, à qui revient la responsabilité du résultat, et, point que peu d'éditeurs SaaS regardent en face, **où ira la capture de valeur dans les cinq prochaines années**.

Ma thèse, en une phrase : la transition agentique va commodifier une grande partie du SaaS horizontal existant en le réduisant à un fournisseur d'API derrière une couche d'orchestration. Les gagnants ne seront pas ceux qui ajoutent un assistant dans un coin de leur produit, mais ceux qui contrôlent la couche d'orchestration, possèdent un avantage métier difficilement substituable, ou se positionnent comme infrastructure de la délégation, permissions, audit, primitives d'action.

Ce que les LLM changent… et ce qu'ils ne changent pas

Première précaution : ne pas confondre nouveauté et révolution.

L'automatisation des workflows existe depuis longtemps. Le RPA a vendu cette promesse pendant une décennie, avec un succès réel sur des tâches très structurées. Les iPaaS connectent des systèmes hétérogènes depuis quinze ans. Les moteurs de règles, les bus d'intégration, les architectures API-first ont déjà déplacé une partie significative de l'exécution loin de l'écran humain.

Les LLM n'inventent pas la délégation logicielle. Ils en élargissent considérablement le périmètre.

Trois apports spécifiques méritent d'être nommés précisément.

Interpréter du non-structuré. Là où le RPA exigeait des entrées prévisibles ou des screen scrapers fragiles, un agent fondé sur un LLM peut traiter un e-mail mal formulé, extraire l'information pertinente d'un PDF inconsistant, ou comprendre qu'une pièce jointe se rapporte probablement à tel contrat. Ce n'est pas magique, ce n'est pas fiable à 100 %, mais cela rend automatisables des zones jusqu'ici réfractaires.

S'adapter localement. Un workflow rigide casse dès qu'un cas sort du scénario prévu. Un agent peut absorber une variation, reformuler une demande, demander une clarification. Cela ne supprime ni les erreurs ni le besoin de garde-fous, mais cela change le rapport entre coût de spécification et couverture des cas.

Abaisser le coût d'entrée. Le langage naturel devient un point d'accès utilisable sans avoir à écrire un script ou configurer un scénario. Un responsable opérationnel peut déclencher une analyse sans passer par une équipe data.

Ces trois apports sont réels. Ils ne sont pas symétriques.

Le coût d'entrée baisse fortement : il est facile de produire un prototype impressionnant en une après-midi. Le coût d'industrialisation, lui, reste élevé. Garantir fiabilité, sécurité, auditabilité, conformité, performance sous charge, gestion des cas limites : c'est aussi coûteux qu'avant, parfois plus, parce que les modes de défaillance d'un système probabiliste sont plus difficiles à anticiper que ceux d'un workflow déterministe.

Cette asymétrie explique l'écart actuel entre la profusion de démos convaincantes et la rareté des déploiements en production sur des tâches sensibles. Il ne se résorbera pas par les seuls progrès des modèles.

Trois dynamiques simultanées

La transition ne se fait pas en trois étapes successives. Elle se fait par trois dynamiques qui coexistent, à des intensités variables selon les secteurs, la tolérance au risque, la qualité des données et la maturité technique.

Greffer des agents sur l'existant. Un assistant qui lit des PDF, remplit des formulaires dans un CRM, prépare des tickets, synthétise des dossiers. Le produit ne change pas en profondeur ; l'agent vient s'y ajouter. C'est la dynamique la plus visible aujourd'hui, et la plus accessible : elle ne demande pas de réécrire le système d'information.

Sa limite apparaît dès que le volume monte. Les interfaces conçues pour des humains deviennent un goulot pour des systèmes qui doivent agir de manière répétable et tracée. Les permissions pensées pour des rôles humains ne couvrent pas les besoins d'un agent. Les logs applicatifs ne disent pas pourquoi tel agent a pris telle décision. La greffe tient, mais le coût de friction augmente avec l'usage.

Éroder certaines interfaces. À mesure que les agents traitent les cas répétitifs, certaines surfaces d'interface perdent leur centralité. L'écran de saisie d'une facture, le formulaire de qualification d'un lead, la grille de tri d'un ticket : ces objets restent nécessaires pour la supervision, mais cessent d'être le lieu où le travail se produit.

Cette dynamique est plus lente et plus discrète. Elle n'efface pas les interfaces ; elle les redirige vers la supervision, la validation, l'arbitrage, l'audit. C'est probablement le scénario dominant à moyen terme pour le SaaS B2B installé.

Concevoir des logiciels agent-native. La troisième dynamique part de la fin : on conçoit le système en supposant qu'il sera appelé par des agents, possiblement externes. Les capacités métier sont exposées explicitement, avec une sémantique exploitable, des permissions fines, des journaux d'action lisibles. L'interface humaine se resserre autour d'un cockpit de supervision.

L'émergence du Model Context Protocol (MCP), des standards de tool use, et plus largement des protocoles d'exposition de capacités à des agents tiers va dans ce sens. C'est la dynamique la plus structurante, mais aussi la plus longue à se déployer, et elle suppose de réarchitecturer, pas de greffer.

Ces trois dynamiques cohabitent. Une même entreprise peut en vivre les trois simultanément sur des périmètres différents. Aucune ne s'imposera mécaniquement à toutes.

L'intention devient un objet de design

L'un des déplacements les moins discutés est aussi le plus profond. Dans le SaaS classique, le design d'expérience optimise un parcours : comment rendre cette action facile à effectuer ? Dans un environnement agentique, une question supplémentaire s'impose : comment aider l'utilisateur à **formuler ce qu'il veut obtenir** ?

C'est une vraie difficulté, et elle a souvent été sous-estimée par les vagues précédentes d'IA conversationnelle.

L'intention n'est pas un objet stable qui préexiste à l'interaction. Trente ans d'UX ont montré que l'utilisateur construit sa demande en manipulant l'outil. Il explore, filtre, trie, voit apparaître une information inattendue, modifie son objectif. Un commercial qui analyse son portefeuille découvre souvent que le problème n'est pas celui qu'il imaginait. Un responsable support qui veut « réduire le nombre de tickets » comprend en cours de route que le vrai sujet est un incident produit mal documenté.

La délégation à un agent ne supprime pas cette boucle de découverte. Elle la déplace. L'utilisateur itère moins sur des écrans et davantage sur la formulation : précision du contexte, contraintes, hypothèses acceptables, niveau d'autonomie consenti, format de restitution. La demande se construit toujours dans l'interaction, mais l'interaction change de nature.

Conséquence pratique : dans un système agentique, certaines frictions deviennent productives. Un bon assistant ralentit avant une action irréversible. Il reformule pour vérifier ce qu'il a compris. Il expose ses incertitudes plutôt que de les masquer. Il demande une confirmation quand l'enjeu est élevé.

La meilleure UX agentique n'est donc pas l'UX la plus fluide. C'est celle qui dose correctement fluidité et contrôle. Un assistant qui exécute sans broncher est un assistant qui produira tôt ou tard une erreur coûteuse et silencieuse. La fluidité maximale est, dans ce contexte, un défaut de conception, pas une vertu.

Cette idée est encore largement absente des produits actuels, qui héritent du dogme SaaS selon lequel toute friction est mauvaise.

Le vrai enjeu : qui possède la couche d'orchestration

Si l'utilisateur final interagit de plus en plus avec un orchestrateur agentique capable de mobiliser plusieurs outils, et de moins en moins avec chaque application métier prise isolément, alors le centre de gravité de la relation client se déplace. L'application continue d'exister, elle stocke les données, applique les règles, exécute les opérations, mais elle devient une infrastructure appelée par d'autres, plutôt qu'une destination où l'utilisateur passe du temps.

Cette désintermédiation est le risque stratégique majeur du SaaS horizontal des dix prochaines années. Et il est largement sous-estimé, ou en tout cas fort peu documenté, par les éditeurs concernés.

Trois positions stables émergeront probablement.

L'orchestrateur capte l'intention, dialogue avec l'utilisateur, coordonne les outils, présente la synthèse. Il observe le plus finement l'usage réel et capture la relation. C'est la position la plus précieuse et la plus disputée. Les hyperscalers et les fournisseurs de modèles sont positionnés pour la prendre ; certains éditeurs verticaux profonds aussi, sur leur domaine.

L'infrastructure de la délégation fournit les primitives qui rendent la délégation gouvernable : protocoles d'exposition de capacités (MCP et successeurs), couches de permissions fines, audit des actions d'agents, gestion d'identité et de consentement, simulation et rollback. Cette couche est en train de naître. Elle sera

probablement disputée par des acteurs spécialisés et par les grands fournisseurs cloud.

Le fournisseur de capacités métier profondes, dans ce contexte, ne survit que s'il offre quelque chose qui ne peut pas être facilement remplacé : expertise réglementaire, données propriétaires, intégrations stratégiques, profondeur fonctionnelle dans un domaine étroit. Un éditeur dans la conformité financière, la santé spécialisée ou la gestion documentaire contractuelle peut tenir cette position. Un CRM horizontal généraliste, beaucoup moins.

Au milieu, la zone à risque : le SaaS horizontal sans avantage métier différenciant. Les outils de productivité génériques, les CRM standard, les outils de support de base, les plateformes de gestion de projet. Ces produits vont continuer d'exister, mais leur surface visible va se rétrécir. Ils deviendront des fournisseurs d'API derrière une interface qu'ils ne contrôlent plus. Le pouvoir de négociation se déplacera vers la couche qui possède la relation utilisateur.

Cette thèse est contestable. Voici ce qui me ferait revenir dessus :

- Si les éditeurs SaaS dominants parviennent à construire des orchestrateurs propres qui restent le point d'entrée privilégié de leurs utilisateurs sur leur périmètre (scénario Salesforce / ServiceNow / Microsoft, plausible mais pas garanti).
- Si les protocoles d'exposition de capacités restent trop fragmentés pour qu'un orchestrateur tiers puisse réellement composer des actions cross-vendor de manière fiable.
- Si les exigences de conformité et de responsabilité fragmentent suffisamment le marché pour que la délégation reste largement intra-application.

Aucun de ces scénarios n'est exclu. Mais aucun n'est non plus le scénario par défaut.

Le mirage de l'outcome pricing

Le pricing au résultat est devenu un lieu commun du discours sur les agents : on facturerait non plus à l'utilisateur, mais au ticket résolu, à la facture traitée, au dossier préparé. C'est séduisant. C'est largement irréaliste, et il vaut la peine de dire pourquoi.

Le pricing à l'outcome se heurte à trois problèmes que les modèles SaaS classiques évitent.

Le problème d'attribution. Si le temps de traitement d'un dossier baisse de 40 %, quelle part revient à l'agent, à la simplification du processus, à la meilleure qualité des données, à la réorganisation interne ? Aucune méthodologie ne tranche cette question de manière convaincante. Or tant qu'elle n'est pas tranchée, le prix est négocié dans l'opacité.

Le problème de définition. Une facture simple, une facture avec anomalie de TVA, une facture internationale avec rapprochement multi-bons-de-commande ne représentent pas la même charge ni le même risque. Définir l'unité tarifaire suppose de spécifier le périmètre, ce qui ramène à la complexité contractuelle qu'on prétendait éviter.

Le problème d'incitation. Si l'éditeur est payé au volume d'actions, il est incité à multiplier les actions. S'il est payé au résultat positif, il est incité à choisir les cas faciles. Aucun de ces alignements n'est trivial à concevoir.

En pratique, les modèles qui s'installeront seront hybrides : une base d'abonnement, des volumes inclus, une tarification additionnelle sur certaines actions, des paliers d'autonomie différenciés, des engagements de performance sur des périmètres très cadrés. Le per-seat ne disparaîtra pas, il survivra notamment pour les rôles de supervision, mais il cessera d'être la métrique centrale sur les usages les plus délégués.

Le vrai enjeu de pricing n'est pas de trouver une formule. C'est de **clarifier ce qui est vendu** : un accès, une capacité, une action exécutée, un volume traité, un niveau d'autonomie consenti, un résultat partiel garanti. Tant que cette clarification reste floue dans les contrats, les négociations seront difficiles et les promesses fragiles.

Le goulot organisationnel

Même en supposant les modèles techniques et économiques résolus, la transition restera lente pour une raison rarement traitée à sa juste valeur : déléguer à un agent transforme la répartition des rôles, des responsabilités et du pouvoir dans l'entreprise.

Qui décide du niveau d'autonomie accordé ? Qui reste comptable des erreurs ? Qui valide les exceptions ? Est-ce qu'une décision prise par un agent suite à une demande d'un employé peut contredire la stratégie du CEO ? Qui explique une décision contestée à un client, à un régulateur, à un juge ? Qui forme les équipes à superviser plutôt qu'à exécuter ? Qui décide qu'une compétence métier peut disparaître de la pratique quotidienne sans devenir inaccessible ?

Ces questions ne sont pas annexes. Elles conditionnent les déploiements bien plus que la performance des modèles.

Un agent techniquement excellent peut rester inemployable s'il n'existe pas de doctrine claire sur sa latitude d'action. À l'inverse, une organisation qui a cadré ses processus, défini ses garde-fous, formé ses superviseurs et accepté ses zones de responsabilité peut obtenir des gains importants avec des modèles imparfaits.

Il existe un risque corollaire encore peu discuté mais qu'on observe déjà : **l'érosion silencieuse des compétences**. Si certaines tâches disparaissent de la pratique quotidienne, les équipes perdent progressivement la capacité à les évaluer. Une entreprise peut automatiser pour gagner en efficacité, puis découvrir trois ans plus tard qu'elle n'a plus en interne assez de personnes capables de superviser correctement le système. Superviser un agent ne demande pas moins de compétence métier que l'exécuter à la main, il en demande même souvent davantage, parce qu'il faut reconnaître les cas limites et les erreurs subtiles, plutôt que d'exécuter une routine.

La délégation à grande échelle exige donc une stratégie explicite de maintien des compétences critiques. Cette stratégie est presque toujours absente des plans de déploiement actuels.

Les modes de défaillance comme limites potentiellement structurelles

Les articles sur les agents traitent généralement les failles connues (hallucinations, injection de prompt, dérive, fragilité hors distribution) comme des « défis à résoudre ». Il est plus prudent de les traiter comme des contraintes potentiellement durables, qui définiront ce qui peut et ne peut pas être délégué.

Les hallucinations ne se résolvent pas, elles se contournent. Aucune amélioration de modèle observée à ce jour ne supprime la possibilité qu'un système probabiliste produise une affirmation plausible mais fausse. Le RAG, la vérification, les contraintes structurelles réduisent la fréquence, pas le phénomène. Cela signifie que tout périmètre où une seule erreur factuelle est catastrophique restera difficile à automatiser sans supervision rapprochée.

L'injection de prompt est un problème de sécurité non résolu. Un agent qui ingère du contenu non maîtrisé ou non fiable (et la liste est longue : e-mail, document fournisseur, page web…) peut être manipulé pour

exécuter des actions non voulues. Les protections existantes ne sont que partielles. Plus un agent a d'accès et d'autonomie, plus la surface d'attaque grandit. Cette contrainte limitera durablement les périmètres où l'on peut déléguer sans cloisonnement strict.

L'observabilité métier reste embryonnaire. Savoir ce qu'un agent a fait est techniquement faisable. Comprendre pourquoi il l'a fait, dans quel cadre interprétatif, avec quelles hypothèses implicites, est beaucoup plus difficile. L'audit d'agents n'est pas un produit mature. Tant qu'il ne le sera pas, les secteurs régulés avanceront lentement, à raison.

La dérive silencieuse. Un agent dont les performances se dégradent, parce que les données d'entrée évoluent, parce qu'un système amont a changé, parce que les utilisateurs reformulent différemment, produit des erreurs qui ne se voient pas dans les métriques classiques. La détection de dérive n'est pas un problème nouveau, mais il est aggravé par la nature probabiliste et faiblement spécifiée des systèmes agentiques.

Aucune de ces limites n'interdit la délégation. Toutes en délimitent le périmètre. Elles expliquent pourquoi le Service as Software se déploiera d'abord sur les tâches où l'erreur est tolérable, détectable et réversible, et plus lentement, plus partiellement, sur les autres.

Ce qu'il faut faire

Pour un éditeur SaaS qui prend cette analyse au sérieux, six décisions structurent les dix-huit prochains mois.

D'abord, **cartographier ses capacités métier réelles**, indépendamment des écrans qui les exposent. Quelles actions le produit permet-il d'accomplir ? Quelles règles applique-t-il ? Quelles décisions prépare-t-il ? Cette cartographie est la fondation de tout le reste.

Ensuite, **exposer ces capacités via des protocoles agent-friendly**, MCP ou équivalent, avec une sémantique métier explicite et pas seulement une API technique. Un agent ne doit pas seulement savoir qu'il peut appeler **updateStatus** ; il doit comprendre ce que ce changement signifie dans le processus métier, quelles conséquences il déclenche, quels contrôles s'appliquent.

Définir **des niveaux d'autonomie graduels** : lecture seule, suggestion, préparation, exécution sous validation, exécution automatique sous contraintes, exécution avec audit renforcé. Ce découpage compte plus que le choix du modèle sous-jacent. Un mauvais périmètre d'autonomie produit plus de problèmes qu'un modèle moins performant.

Construire **une observabilité métier**, pas seulement technique. Quelle demande a été reçue, comment elle a été interprétée, quelles données ont été utilisées, quelles actions ont été proposées, lesquelles exécutées, lesquelles refusées, par qui. C'est une condition d'adoption dans tout environnement sensible.

Repenser **les métriques produit** : moins de « temps passé dans l'application », plus de « taux de reprise humaine », « taux d'exception », « qualité du résultat », « conformité des actions ».

Enfin, et c'est le choix le plus stratégique, **se positionner explicitement dans la chaîne de valeur**. Application visible directement utilisée par les humains ? Plateforme de capacités orchestrable par des tiers ? Propre couche d'orchestration métier ? Cockpit de supervision ? Ce choix détermine la roadmap, le pricing, la relation client, la défense concurrentielle. Le faire par défaut, c'est presque toujours choisir la position la plus exposée à la commodification.

Un pari, pas une certitude

Le SaaS ne meurt pas. L'interface graphique ne disparaît pas. L'humain ne sort pas de la boucle. Tout cela est faux et il faut le dire clairement, parce que la version caricaturale du discours sur les agents finit par discréditer l'analyse sérieuse.

Ce qui change est plus précis et plus intéressant : une part croissante de la valeur du logiciel se déporte de la manipulation directe vers la délégation gouvernée. Cette translation s'accompagne d'une redistribution de la valeur entre couches : l'orchestration capte la relation, l'infrastructure de délégation capte la gouvernance, les capacités métier profondes survivent, le SaaS horizontal sans différenciation se commodifie.

Cette thèse peut être fausse, c'est vrai. Si dans cinq ans les éditeurs SaaS dominants ont absorbé la couche d'orchestration sur leur périmètre, si les protocoles inter-vendor n'ont pas mûri, si la pression réglementaire a fragmenté le marché en silos applicatifs, alors j'aurai surestimé la portée de la désintermédiation.

Mais le scénario par défaut, en l'absence d'action stratégique délibérée, est que la valeur remonte vers les couches qui possèdent la relation et redescende vers les couches qui possèdent la donnée, en laissant au milieu une zone que les éditeurs SaaS horizontaux occupent aujourd'hui sans toujours mesurer combien elle est fragile.

La vraie question, pour un éditeur, n'est pas « comment ajouter de l'IA à mon produit ». C'est : **où vais-je me retrouver dans la chaîne de valeur quand l'utilisateur final aura cessé d'entrer chez moi par la porte que j'ai conçue pour lui ?**

Cette question a des réponses. Aucune n'est confortable. Aucune ne se résout en ajoutant un chatbot dans un coin de l'écran.

Références

1. « AI startups: Sell work, not software ».
https://www.sarahtavel.com/p/ai-startups-sell-work-not-software

2. « A few “Sell Work, Not Software” updated thoughts ».
https://www.sarahtavel.com/p/a-few-sell-work-not-software-updated

3. « AI leads a service as software paradigm shift ».
https://foundationcapital.com/ideas/ai-leads-a-service-as-software-paradigm-shift

4. « The $4.6T Services-as-Software opportunity: Lessons from the first year ».
https://foundationcapital.com/ideas/the-4-6t-services-as-software-opportunity-lessons-from-the-first-year

5. « AI Is Driving A Shift Towards Outcome-Based Pricing ».
https://a16z.com/newsletter/december-2024-enterprise-newsletter-ai-is-driving-a-shift-towards-outcome-based-pricing/

6. « Introducing the Model Context Protocol ».
https://www.anthropic.com/news/model-context-protocol

7. « Specification - 2025-06-18 ».
https://modelcontextprotocol.io/specification/2025-06-18

8. « Function calling ».
https://developers.openai.com/api/docs/guides/function-calling

9. « Tools ».
https://openai.github.io/openai-agents-python/tools/

10. « Top 10 for Large Language Model Applications ».
https://owasp.org/www-project-top-10-for-large-language-model-applications/

11. « LLM01:2025 Prompt Injection ».
https://genai.owasp.org/llmrisk/llm01-prompt-injection/

12. « Not what you've signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection ».
https://arxiv.org/abs/2302.12173

13. « Prompt Injection attack against LLM-integrated Applications ».
https://arxiv.org/abs/2306.05499

14. « Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile ».
https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-generative-artificial-intelligence

15. « AI Act - Regulatory framework ».
https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

16. « Hallucination Mitigation for Retrieval-Augmented Large Language Models: A Review ».
https://www.mdpi.com/2227-7390/13/5/856