Livre blanc « Gouvernance démontrable »

"Transformer les politiques organisationnelles en garanties vérifiables"

Conception et rédaction : Bruno URBAN
Contenu assisté par un LLM (IA)
Date : Septembre 2026
Contact : integration@sotead.com

Avant propos

Ce livre blanc est né de quatre interrogations qui, bien qu'elles concernent des domaines différents, posent finalement une même question : **comment construire un système d'information capable non seulement d'appliquer les décisions de l'organisation, mais également de démontrer qu'elles sont effectivement respectées ?**

Cyber-Défense

Les attaques automatisées se multiplient et l'utilisation de l'intelligence artificielle augmente progressivement leur capacité d'adaptation et leur efficacité. En parallèle, l'automatisation et l'IA prennent une place croissante dans les dispositifs de détection et de réponse.
Cette évolution conduit cependant à deux questions fondamentales :

- Que se passe-t-il lorsque la première ligne de défense est contournée ou dépassée ?
- Quelles garanties subsistent alors au sein même du système d'information ?

Une défense périmétrique, aussi performante soit-elle, ne peut constituer à elle seule la dernière ligne de protection. Les fonctions protégées doivent elles-mêmes disposer de mécanismes capables de contrôler les conditions dans lesquelles elles acceptent de s'exécuter, les composants qu'elles utilisent et les données qu'elles consomment.

Lorsqu'une garantie nécessaire à l'exécution ne peut plus être établie, une décision doit alors être prise : privilégier la continuité du traitement au risque de propager une compromission, ou refuser l'exécution au risque de provoquer une indisponibilité.

Cette décision n'est pas uniquement technique. **Elle relève de la gouvernance et doit pouvoir être définie explicitement par l'organisation.**

Cyber-Résilience

Aucune organisation ne peut considérer qu'elle sera toujours capable d'empêcher une cyberattaque. La cyber-résilience suppose donc de préparer non seulement la défense, mais également la capacité à maintenir, reconstruire ou restaurer les fonctions essentielles après un incident majeur.

Cela nécessite de pouvoir répondre à plusieurs questions :

- Quelles sont les fonctions critiques de l'organisation ?
- Quel est leur périmètre réel et de quelles ressources, données et autres fonctions dépendent-elles ?
- Comment peuvent-elles être reconstruites ou restaurées rapidement dans un environnement dont la confiance peut être établie ?
- Quelle perte de données, de capacité ou de fonctionnalité est acceptable ?
- Quel fonctionnement dégradé l'organisation est-elle prête à accepter ?

Répondre à ces questions nécessite notamment de disposer d'un inventaire à jour des fonctions critiques et de leurs dépendances, de mécanismes de déploiement maîtrisés, de données sauvegardées et restaurables ainsi que d'exercices périodiques permettant de démontrer la capacité réelle de restauration.

La résilience dépend donc fortement de la connaissance que l'organisation possède de son propre système d'information.

Or cette connaissance est souvent maintenue indépendamment du système qu'elle décrit et peut rapidement devenir incomplète ou obsolète.

Idéalement, le système d'information devrait contribuer lui-même à produire et maintenir une représentation exploitable de ses fonctions, de leurs dépendances et des garanties qui leur sont associées.

Une telle connaissance pourrait alors devenir non seulement documentaire, mais également exploitable pour l'analyse d'impact, la restauration, le contrôle et la prise de décision.

Gouvernance

Les organisations définissent de nombreuses politiques destinées à encadrer leur système d'information. Elles résultent de leurs obligations réglementaires, de leurs exigences de sécurité, de leurs politiques de gouvernance des données, de leurs contraintes opérationnelles mais également de leurs choix stratégiques, notamment en matière de résilience ou de souveraineté.

Définir ces politiques est une première difficulté. Les intégrer effectivement dans le système d'information et démontrer leur application en constitue une seconde, souvent beaucoup plus complexe.

Même lorsqu'une organisation développe de moins en moins directement les solutions constituant son système d'information, les services qu'elle utilise doivent être intégrés entre eux. Cette intégration nécessite nécessairement des mécanismes techniques : code, composants d'intégration, plateformes ou solutions d'orchestration.

La question devient alors de savoir comment utiliser ces points d'intégration pour appliquer les politiques de l'organisation et produire les éléments permettant d'en démontrer le respect.

Un décideur devrait ainsi pouvoir répondre aux questions suivantes :

- Comment les politiques de l'organisation sont-elles effectivement appliquées dans le système d'information ?
- Comment démontrer que les contrôles et actions correspondants ont réellement été exécutés ?
- Comment une décision stratégique ou une nouvelle politique peut-elle être répercutée rapidement lorsque de nombreuses fonctions métier sont concernées ?
- Comment conserver cette capacité d'évolution sans imposer aux différentes équipes des cycles de vie identiques ?

Répondre à ces questions nécessite de réduire la dépendance entre la logique métier et les mécanismes chargés d'appliquer la gouvernance.

Cela suppose également de passer d'une gouvernance principalement fondée sur des prescriptions et des contrôles a posteriori à une gouvernance capable de produire, au cours même de l'exécution du système d'information, les preuves permettant d'en démontrer l'application.

Souveraineté

Les systèmes d'information modernes reposent sur un nombre croissant de services, de technologies et de fournisseurs externes. Cette évolution apporte de l'agilité mais crée également des dépendances susceptibles de limiter la capacité d'une organisation à faire évoluer son système d'information selon ses propres choix.

La souveraineté ne peut donc être réduite à la localisation des données ou à la nationalité d'un fournisseur. Elle concerne également la capacité de l'organisation à conserver la maîtrise de ses traitements, de ses données, de ses politiques et des preuves nécessaires à leur gouvernance.

Plusieurs questions se posent alors :

- Dans quelle mesure une fonction métier dépend-elle de la technologie qui stocke ou transporte ses données ?
- Peut-on remplacer une infrastructure ou un fournisseur sans remettre en cause la logique métier ?
- L'organisation conserve-t-elle la maîtrise des éléments nécessaires pour comprendre, reconstruire et contrôler ses traitements ?
- Les preuves permettant d'établir la confiance restent-elles exploitables indépendamment des solutions qui les ont produites ?

La souveraineté ne signifie pas l'absence de dépendances. Elle suppose plutôt de **connaître ces dépendances, de les maîtriser et de préserver la capacité à les remplacer lorsque cela devient nécessaire**.

Conclusion de l’avant propos

Ces quatre problématiques sont généralement abordées séparément. Elles partagent pourtant une même difficulté : l'organisation doit connaître ce qu'elle exécute, maîtriser les conditions de cette exécution et disposer d'éléments objectifs lui permettant d'établir la confiance.

Défendre ne suffit pas si aucune garantie ne subsiste après le franchissement des défenses.

Restaurer ne suffit pas si la confiance dans ce qui est restauré ne peut être établie.

Gouverner ne suffit pas si l'application des décisions ne peut être démontrée.

Enfin, choisir ses technologies ne suffit pas à assurer sa souveraineté si les fonctions métier en deviennent indissociables.

Le présent livre blanc explore une approche visant à rendre ces garanties explicites, exécutables et démontrables, tout en préservant autant que possible l'indépendance des fonctions métier vis-à-vis des mécanismes techniques qui les supportent.

Chapitre 1 – Pourquoi avons-nous besoin d’un nouveau paradigme ?

1.1 Le constat

Depuis plusieurs décennies, le développement de logiciels s’est principalement concentré sur la fourniture de fonctionnalités métier.

La qualité d’un logiciel se mesurait principalement par sa capacité à réaliser correctement les opérations métiers attendues.

Aujourd’hui, cette perspective n’est plus suffisante.

Un logiciel moderne n’est plus simplement un ensemble de fonctionnalités métiers.

Il opère au sein d’un système d’information distribué, connecté à de nombreux services, traitant des données de toutes natures et soumis à des exigences croissantes en matière de sécurité, de conformité réglementaire, de résilience opérationnelle, de gouvernance des données et de souveraineté numérique.

Dès lors, la valeur d’un logiciel ne dépend plus uniquement de « ce qu’il fait », mais aussi de « comment il le fait ».

1.2 Un changement dans la nature, pas seulement dans la technologie

Les architectures distribuées, la virtualisation, le Cloud Computing, les API, les plateformes SaaS et le traitement de données à grande échelle ont progressivement fait disparaître les frontières traditionnelles des systèmes d'information d'entreprise.

Les logiciels ne contrôlent plus l’intégralité de leur environnement d’exécution.

Les données circulent en permanence entre les organisations, les régions géographiques, les fournisseurs de cloud et les plateformes techniques hétérogènes.

Dans un tel environnement, la responsabilité portée par un logiciel s’étend bien au-delà de son propre périmètre fonctionnel.

1.3 Un nombre croissant de responsabilités

Les logiciels modernes doivent désormais satisfaire simultanément plusieurs catégories d’exigences.

Les « exigences verticales » soutiennent directement les objectifs métier :

- gérer les clients
- calculer les prix
- émettre des factures

- planifier les opérations terrain

Les « exigences horizontales » concernent le système d'information dans son ensemble :

- sécurité
- gouvernance des données
- conformité règlementaire
- auditabilité
- observabilité
- résilience
- gouvernance opérationnelle
- stratégie d’entreprise

Contrairement aux fonctionnalités métier, ces exigences concernent l'ensemble du système d'information, indépendamment de la finalité de chaque application.

1.4 Le défi actuel

Aujourd’hui, ces exigences horizontales s’expriment principalement à travers :

- les politiques
- les recommandations
- les procédures
- les normes

Elles restent cependant largement déconnectées de leur mise en œuvre technique.

En pratique :

- Les équipes de sécurité définissent les politiques de sécurité
- Les équipes de gouvernance des données définissent les règles relatives aux données
- Les équipes de conformité établissent les exigences réglementaires
- Les architectes d'entreprise publient des directives architecturales
- Les équipes de développement tentent d'implémenter certaines de ces exigences dans leurs applications

Cette organisation présente plusieurs limites.

- Les responsabilités se fragmentent
- Le niveau réel de conformité est difficile à évaluer
- Les audits deviennent le principal mécanisme de vérification

En fin de compte, la mise en œuvre finale dépend en grande partie de la manière dont chaque équipe de développement interprète les exigences de l'organisation.

1.5 Les limites des approches traditionnelles

La plupart des organisations ont répondu à ces exigences croissantes en introduisant des processus supplémentaires.

- les méthodologies de développement ont évolué
- les jalons qualité se sont multipliés
- les revues de code sont devenues systématiques
- des outils d'analyse de code statique et dynamique ont été introduits
- le suivi opérationnel s'est considérablement amélioré

Bien que chacune de ces initiatives résolve une partie du problème, elles fonctionnent généralement de manière indépendante.

De ce fait, les organisations accumulent souvent des contrôles sans pouvoir démontrer que les garanties attendues sont effectivement respectées tout au long de l’exécution d’un processus métier.

Le problème n’est donc plus l’absence de contrôles.

C'est l'absence d'un modèle cohérent capable de relier les politiques organisationnelles, les contrôles techniques et les preuves vérifiables.

1.6 La confiance ne peut plus être assumée

Traditionnellement, la confiance s’établit au moyen d’hypothèses.

Un logiciel est considéré comme fiable car :

- il a été développé par une équipe interne ou gérée en interne
- il a réussi les tests de recette
- il a fait l’objet d’une revue
- il a été déployé selon des procédures approuvées

Ces hypothèses étaient généralement acceptables lorsque les systèmes évoluaient lentement et restaient relativement statiques.

Les systèmes d’information modernes évoluent continuellement.

Les logiciels sont déployés plusieurs fois par jour.

Les dépendances sont mises à jour automatiquement.

L’infrastructure est provisionnée de manière dynamique.

L'intelligence artificielle contribue de plus en plus à la génération de code.

Dans un tel environnement, la confiance ne peut plus reposer uniquement sur une validation préalable.

Elle doit être établie en permanence tout au long de l’exécution.

C’est l’un des principes fondamentaux d’une démarche « Zero Trust ».

Rien n’est considéré comme digne de confiance simplement parce qu’on lui a déjà fait confiance.

Chaque propriété importante doit être vérifiée lorsque cela compte.

1.7 De la confiance à la démonstration

Cette observation conduit naturellement à une question fondamentale.

Comment une organisation peut-elle garantir que ses politiques ne sont pas simplement documentées, mais qu’elles sont effectivement appliquées ?

La gouvernance traditionnelle répond généralement à cette question par le biais d’audits.

Les audits restent essentiels mais ils ont généralement lieu après le déploiement et évaluent des échantillons de preuves collectées au fil du temps.

L'architecture proposée dans ce document suit une philosophie différente.

Au lieu de supposer que les politiques organisationnelles ont été respectées, l’approche vérifie leur application avant, pendant et après chaque exécution métier.

La confiance ne repose donc plus sur des déclarations.

Elle est établie au moyen de « garanties démontrables ».

Chaque garantie est réalisée par au moins une action et/ou un contrôle technique.

Chaque action et/ou contrôle génère des preuves vérifiables.

Ensemble, ces éléments de preuve forment une chaîne vérifiable démontrant que les politiques attendues ont été effectivement appliquées.

1.8 Une perspective différente

Ce document ne présente pas une autre plateforme d'intégration.

Il ne propose pas non plus un autre moteur d'orchestration.

Son objectif est fondamentalement différent.

Il introduit un modèle d'exécution dans lequel les politiques organisationnelles deviennent des garanties exécutables, vérifiables et auditables.

Le traitement métier n'est plus exécuté simplement parce qu'il a été demandé.

Il n'est exécuté que lorsque les conditions requises pour établir la confiance ont été démontrées.

L’exécution devient alors la conséquence de garanties vérifiées plutôt que le point de départ du processus.

1.9 La souveraineté numérique : au-delà de l'infrastructure

La souveraineté numérique est souvent réduite au choix d'un fournisseur de cloud ou à la localisation des données.

Si ces aspects sont importants, ils ne constituent qu'une partie du problème.

Une organisation perd également une partie de sa souveraineté lorsque ses traitements métier deviennent dépendants d'une technologie particulière, d'un format de données propriétaire, d'un moteur d'intégration ou d'un fournisseur de services dont le remplacement nécessite de modifier les logiciels eux-mêmes.

L'architecture proposée dans ce document vise à réduire ces dépendances en séparant la logique métier des mécanismes techniques permettant d'accéder aux données, d'assurer leur transport, d'appliquer les politiques de gouvernance et de produire les preuves nécessaires à leur démonstration.

Le métier manipule des représentations logiques des données plutôt que leurs représentations physiques. Les interactions avec les infrastructures de stockage, de transport ou d'échange sont déléguées à des composants spécialisés.

Cette séparation réduit les dépendances technologiques et facilite l'évolution des infrastructures sans remettre en cause les traitements métier.

Les évolutions technologiques, comme le remplacement d'un système de fichiers par un stockage objet, l'introduction d'une API HTTP ou la migration vers une nouvelle plateforme, peuvent ainsi être réalisées avec un impact limité sur les traitements métier.

Cette approche ne supprime pas les dépendances aux infrastructures, qui demeurent nécessaires à l'exécution des traitements, mais elle les isole dans des composants spécialisés.

L'organisation conserve ainsi la maîtrise de sa logique métier, de ses politiques, de ses preuves d'exécution et des connaissances produites sur son système d'information.

La souveraineté ne résulte donc pas uniquement de la possession des infrastructures.

Elle repose également sur la capacité de l'organisation à faire évoluer ses traitements, ses données et ses mécanismes de gouvernance indépendamment des choix technologiques qui les supportent.

1.10 Conclusion du chapitre

La complexité croissante des systèmes d’information modernes nécessite une approche différente de l’architecture logicielle.

Les fonctionnalités métiers seules ne suffisent plus.

Les organisations doivent également être en mesure de démontrer que leurs logiciels sont conformes aux politiques de sécurité, aux exigences de gouvernance, aux obligations réglementaires et aux contraintes opérationnelles.

L'approche présentée dans ce livre blanc propose de passer d'une « gouvernance déclarative » à une « gouvernance démontrable ».

Au lieu de s'appuyer principalement sur des procédures et des audits périodiques, elle introduit un modèle dans lequel les politiques organisationnelles sont traduites en actions et/ou contrôles techniques qui produisent des preuves vérifiables tout au long de l'exécution.

Cette approche privilégie délibérément le « refus de service à l'exécution incontrôlée » : dès lors qu'une garantie requise ne peut pas être démontrée, l'exécution est interrompue.

Bien que cette posture de sécurité puisse ne pas convenir à toutes les organisations, elle reflète un principe fondamental : la confiance ne doit jamais être présumée lorsqu'elle ne peut pas être démontrée.

L'utilisation croissante de l'IA dans le développement du code et les cyberattaques renforce encore ce besoin.

À mesure que les cycles d'attaque s'accélèrent et s'automatisent, les mesures de prévention ne suffisent plus à elles seules.

Chaque couche de défense doit s'inscrire dans un cadre où la confiance est établie en continu plutôt qu'accordée implicitement.

Le chapitre suivant présente les fondements conceptuels de cette approche en expliquant comment les politiques organisationnelles peuvent être transformées en garanties démontrables.

Chapitre 2 – Transformer les politiques organisationnelles en garanties démontrables

2.1 Les fondations de l’architecture proposée

Lorsqu'une équipe métier conçoit un traitement, sa préoccupation est relativement simple : transformer une donnée d'entrée en une donnée de sortie.

Quel que soit le domaine d'activité, ce traitement peut généralement être résumé en trois étapes :

1. obtenir les données nécessaires au traitement
2. appliquer la logique métier
3. publier les résultats

Du point de vue du métier, cette représentation est suffisante. Elle décrit la finalité du traitement sans l'encombrer de préoccupations techniques ou organisationnelles.

Pourtant, ce même traitement est également concerné par de nombreuses exigences portées par d'autres équipes de l'entreprise.

- La gouvernance des données souhaite s'assurer que les données sont conformes aux contrats établis, qu'elles sont traçables et que leur qualité est maîtrisée.
- Les équipes de sécurité souhaitent vérifier l'intégrité des composants exécutés, contrôler l'origine des données, protéger les secrets et disposer de preuves permettant de détecter toute altération.
- Les équipes d'exploitation recherchent des mécanismes de reprise après incident, de journalisation et de supervision.
- Les architectes souhaitent maîtriser les dépendances entre les traitements et faciliter les évolutions du système d'information.
- Enfin, la Direction souhaite disposer d'une vision fiable des traitements exécutés et pouvoir démontrer que les politiques de l'entreprise sont effectivement appliquées.

Toutes ces attentes concernent un même traitement métier, mais chacune possède son propre cycle de vie, ses propres priorités et ses propres objectifs.

Traditionnellement, leur prise en compte repose principalement sur le développement spécifique de chaque logiciel. Les développeurs doivent intégrer les exigences de chaque domaine de gouvernance dans leur code, ce qui conduit progressivement à des traitements de plus en plus complexes, difficiles à maintenir et dont le comportement dépend largement des pratiques de chaque équipe.

Cette approche présente plusieurs limites :

- les mêmes mécanismes sont développés plusieurs fois
- les politiques sont interprétées différemment selon les projets
- l'évolution d'une politique nécessite souvent de modifier plusieurs applications
- les audits reposent largement sur des analyses documentaires ou des contrôles a posteriori
- les cycles de vie des différentes équipes deviennent fortement dépendants les uns des autres

Or, les organisations modernes recherchent exactement l'inverse.

Les traitements doivent pouvoir évoluer rapidement, être déployés indépendamment, changer d'infrastructure, intégrer de nouveaux partenaires ou répondre à de nouvelles exigences réglementaires sans remettre en cause l'ensemble du système.

L'objectif n'est donc plus d'intégrer les politiques organisationnelles dans les logiciels, mais de fournir un mécanisme permettant de les appliquer automatiquement, indépendamment du développement métier.

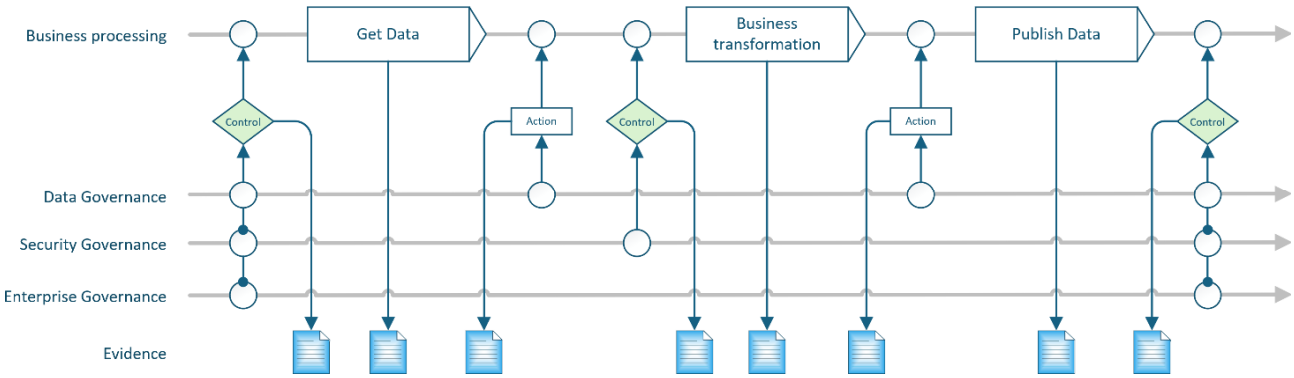
L'architecture proposée repose sur un principe simple : **Chaque domaine de gouvernance exprime ses objectifs sous la forme de « contrôles » et/ou « d'actions » pouvant être exécutés à des étapes précises d'un traitement métier.**

Ces contrôles et ces actions sont injectés dans le flux de traitement sans modifier la logique fonctionnelle elle-même.

Note : Le terme « injectés » désigne ici l'application de contrôles et d'actions au cycle d'exécution métier ; il n'implique pas une modification du code métier.

Le traitement métier continue ainsi à se concentrer exclusivement sur la transformation des données, tandis que les contrôles et/ou actions injectés répondent aux besoins de gouvernance en fournissant les preuves nécessaires.

Cette approche est illustrée par la figure ci-dessous.



Chaque domaine de gouvernance conserve son autonomie et son propre cycle de vie.

Les politiques qu'il définit sont traduites en mécanismes techniques réutilisables, indépendants des traitements métier.

Ces mécanismes peuvent être des contrôles, tels que la vérification de l'intégrité d'une donnée ou la validation d'un contrat de données, ou des actions, comme la signature d'un résultat, la journalisation d'un événement ou la conservation d'un état transactionnel.

Un même mécanisme peut simultanément répondre aux besoins de plusieurs domaines de gouvernance.

Par exemple, un contrôle d'intégrité des données contribue :

- à la gouvernance des données en garantissant la confiance dans les informations échangées
- à la sécurité en détectant toute altération
- à l'audit en fournissant une preuve vérifiable
- à la stratégie de l'entreprise en renforçant la confiance dans les décisions prises à partir de ces données

L'intérêt de cette approche ne réside donc pas uniquement dans l'exécution de contrôles supplémentaires.

Chaque contrôle et chaque action produisent des « preuves » qui deviennent les éléments objectifs permettant de démontrer que les politiques de l'entreprise ont effectivement été appliquées.

La gouvernance ne repose alors plus uniquement sur des procédures ou des déclarations, mais sur des faits vérifiables.

Cette séparation entre la logique métier, les mécanismes de gouvernance et les preuves produites constitue le fondement de l'approche présentée dans ce document.

Les chapitres suivants montreront comment ces mécanismes permettent de construire progressivement des garanties démontrables portant sur les données, les traitements et, plus largement, sur la gouvernance du système d'information.

2.2 Une interprétation différente du Zero Trust

Le terme « Zero Trust » est fréquemment associé à la gestion des identités, à la segmentation du réseau ou à l'authentification continue.

Même si ces aspects restent essentiels, ils ne répondent qu’à une partie du problème.

Ce document adopte une interprétation plus large.

Plutôt que de se demander s'il faut faire confiance à une identité ou à un système, elle pose une question différente : **La propriété exigée par l'organisation peut-elle être démontrée au moment où elle compte ?**

La confiance n’est plus attachée à une identité, un logiciel ou un composant d’infrastructure.
Au lieu de cela, elle est attachée à la démonstration de propriétés spécifiques.

Par exemple :

- les données saisies sont-elles authentiques ?
- le traitement métier a-t-il été validé ?
- les données sont-elles conformes à leur contrat ?
- la sortie a-t-elle été signée ?
- la transaction peut-elle être récupérée ?
- la « Définition du Processus Métier (Business Process Definition) » a-t-elle été approuvée ?

Chacune de ces questions peut recevoir une réponse objective.

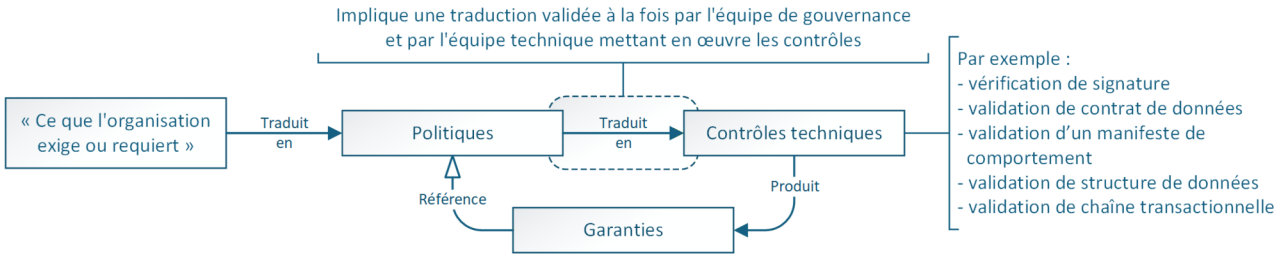
Chaque réponse peut être étayée par des preuves.

La confiance devient alors le résultat de garanties démontrées.

2.3 Le modèle de construction de la confiance

Nous avons vu que l'approche suit un modèle de raisonnement simple mais rigoureux :

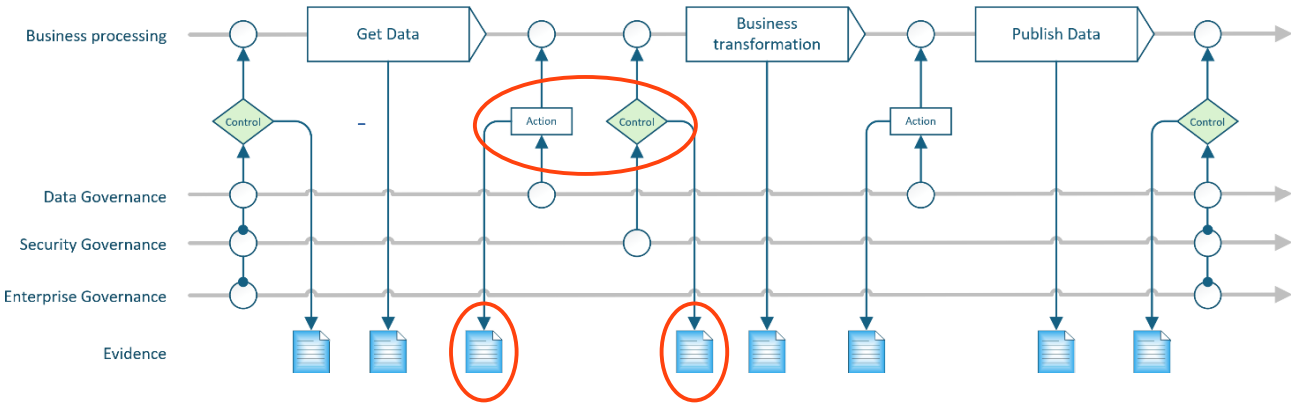
- chaque exigence organisationnelle est progressivement transformée en actions et/ou contrôles techniques.
- chaque action et/ou contrôle est mis en œuvre par un composant spécialisé qui produit des preuves qui apportent des garanties et, in fine, favorisent la confiance.



Modèle de construction de la confiance – Cycle de vie de la gouvernance opérationnelle – V1.0

Un « Orchestrateur de Garanties » pilote la chaîne d'actions et de contrôles lors d'une demande de traitement métier.

Les actions et les contrôles fournissent des preuves qui permettent à l'Orchestrateur de Garanties de décider si l'exécution peut continuer.



Ces séquences constituent la base de toute l'architecture.

Elles établissent une relation directe entre l'intention de l'organisation et le comportement de l'exécution.

Au lieu de s’appuyer sur des hypothèses, chaque décision d’exécution est étayée par des preuves vérifiables.

Une représentation simplifiée du modèle est alors :

Objectif Organisationnel → Politique → Action(s) et/ou Contrôle(s) → Preuve(s) → Garantie(s) → Confiance

2.4 Séparation des responsabilités

L'un des principes majeurs de l'architecture proposée est la séparation des responsabilités.

Chaque partie prenante contribue au modèle de confiance sans nécessiter une connaissance détaillée de l'ensemble du système.

Par exemple :

- les équipes métiers définissent les règles métier et la propriété des données
- les équipes de gouvernance des données définissent les « Contrats de Données (Data Contracts) »
- les équipes de sécurité définissent les politiques de sécurité et les contrôles associés
- la gestion des risques valide les capacités techniques acceptables
- les équipes opérationnelles définissent les politiques d'exécution
- les développeurs implémentent la logique métier
- l’architecture proposée applique les garanties qui en résultent

Chaque partie prenante signe ou approuve uniquement les artefacts relevant de sa zone de responsabilité.

L’architecture proposée combine ensuite ces engagements indépendants dans un modèle d'exécution cohérent.

La confiance n’est donc pas déléguée à une seule équipe, elle se construit par la combinaison de garanties validées de manière indépendante.

2.5 Gouvernance démontrable

Les paragraphes précédents ont montré que :

- les politiques sont traduites en actions et/ou contrôles
- les actions et/ou contrôles produisent des preuves
- les preuves construisent des garanties

La gouvernance ne repose donc plus uniquement sur :

- des procédures

- des audits documentaires
- des déclarations

Elle repose également sur des preuves techniques produites automatiquement pendant l'exécution.

Cette approche s’appelle la « **Gouvernance démontrable** ».

Toute organisation peut progressivement adopter cette approche sans remplacer ses logiciels existants.

La transition peut commencer par des pratiques simples telles que :

- décrire formellement les « Contrats de Données (Data Contracts) »
- signer des artefacts critiques
- valider le comportement du logiciel avant le déploiement
- produire des preuves vérifiables lors de l'exécution

Chaque garantie supplémentaire réduit l’incertitude.

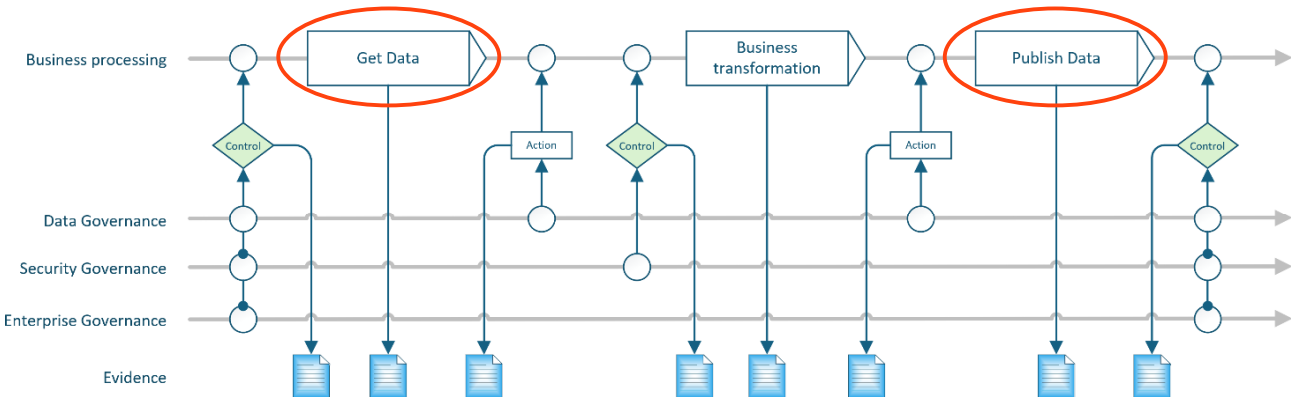
Chaque nouvel élément de preuve renforce la confiance.

L’objectif n’est donc pas d’instaurer la confiance en une seule fois, mais de l’établir de manière continue tout au long du cycle de vie de chaque processus métier.

Les chapitres suivants décrivent les garanties qui rendent possible cette confiance continue, à commencer par la confiance établie dans les données elles-mêmes.

Chapitre 3 – Établir la confiance dans les données

Dans ce chapitre nous allons nous intéresser aux actions et contrôles qui sont jugés nécessaires pour garantir la confiance dans les données utilisées par un traitement métier et dans les données créées par celui-ci.



3.1 Pourquoi la confiance dans la donnée est importante

Même l'algorithme métier le plus fiable ne peut produire des résultats dignes de confiance que si les données d'entrée sont elles-mêmes fiables.

Ce principe est bien connu, mais il est souvent sous-estimé.

Les organisations investissent généralement des efforts considérables pour sécuriser les logiciels tout en supposant que les données utilisées dans ces logiciels sont dignes de confiance.

En réalité, cette hypothèse est rarement démontrable.

Les données peuvent provenir de bases de données, de systèmes de fichiers, d'API, d'un intermédiaire (message brokers) ou d'organisations partenaires.

Quelle que soit leur origine, le traitement métier ne doit jamais supposer que les données utilisées sont authentiques, complètes et structurellement valides.

Au lieu de cela, la confiance doit être progressivement établie avant le début de tout processus métier.

3.2 Vérification de l'intégrité des données

La première propriété à établir est l’intégrité et vise à répondre à la question : les données ont-elles été modifiées depuis leur production ?

Un simple hachage ne suffit pas pour répondre à cette question.

Bien que le hachage détecte les modifications, un attaquant capable de modifier les données peut généralement également recalculer le hachage.

Par conséquent, l’intégrité ne peut pas reposer uniquement sur le hachage.

Cela nécessite une authentification cryptographique.

L'approche recommandée consiste donc à signer numériquement les données.

Une signature numérique fournit davantage d’informations qu’un hachage seul.

Elle permet à l'organisation de déterminer :

- qui a produit les données ?
- quand ont-elles été signé ?
- quel algorithme cryptographique a été utilisé ?
- quelle clé de signature était impliquée ?
- est ce que cette clé a ensuite été révoquée ou compromise ?

L’approche vérifie ces propriétés avant l'exécution chaque fois qu'une signature est disponible.

Lorsque les données d'entrée ne peuvent pas être signées (par exemple parce qu'elles proviennent de systèmes externes), l’approche ne peut pas établir de confiance cryptographique.

Dans de tels cas, la responsabilité est transférée au métier qui doit déterminer si les informations reçues restent acceptables malgré l’absence de garanties cryptographiques. Cette acceptabilité peut être associée à des contrôles spécifiques.

3.3 Les Contrats de Données (Data Contracts)

La gouvernance des données exige que les organisations comprennent :

- d'où proviennent les données
- où elles sont consommées
- à qui elles appartiennent
- comment elles évoluent
- quels processus métiers les transforment

Pour les organisations soumises à des réglementations telles que le « Règlement Général sur la Protection des Données (RGPD) », le maintien de ces connaissances n’est plus facultatif.

Un Contrat de Données répond donc simultanément à plusieurs objectifs :

- pour les développeurs, il définit les garanties structurelles requises avant exécution
- pour un traitement métier, il définit la représentation logique attendue des données
- pour les équipes de gouvernance, il documente la manière dont l’information circule dans le système d’information

Ces Contrats de Données ne sont valides que s’ils sont versionnés et signés.

Ils deviennent, alors, également des artefacts de gouvernance dont l'intégrité peut être vérifiée en permanence.

3.4 Validation structurelle

Des données authentiques ne sont pas nécessairement utilisables par un traitement métier.

Avant exécution, l’architecture proposée vérifie que les données reçues sont conformes à la structure attendue par le traitement métier.

Cette vérification est effectuée en s'appuyant sur des propriétés du « Contrat de Données (Data Contract) » associé.

Un Contrat de Données décrit formellement :

- le format attendu
- la structure des données
- attributs obligatoires (comme les en têtes par exemple)
- types de données
- règles de validation
- politiques de rigueur facultatives

Contrairement à la validation métier, la validation structurelle reste entièrement indépendante de la logique métier.

Plutôt que d'évaluer la validité métier, la validation structurelle garantit que le composant reçoit des données conformes au contrat convenu.

Cette séparation simplifie considérablement les implémentations métiers tout en améliorant la réutilisation et la souveraineté.

3.5 Abstraction de la représentation physique des données

Les logiciels métier traditionnels associent étroitement la logique métier aux préoccupations techniques.

Une fonction métier est fréquemment responsable de :

- localiser des données
- s'authentifier auprès de systèmes externes
- comprendre les formats de stockage
- analyser des fichiers
- convertir des encodages de caractères
- gérer des bibliothèques de sérialisation

Ces responsabilités n'ont rien à voir avec la logique métier elle-même et reflètent simplement des contraintes techniques.

Ce couplage réduit la réutilisabilité et la souveraineté.

Tout changement dans la technologie de stockage nécessite souvent des modifications du code métier, même si les règles métier restent inchangées.

L'architecture proposée suit une approche différente : **un traitement métier s'exécute dans un environnement de données logique plutôt que physique**.

Il ne manipule jamais directement les fichiers CSV, les jeux de résultats SQL, les documents JSON ou XML.

Au lieu de cela, l'architecture proposée convertit les représentations physiques en abstractions logiques standardisées.

Les résultats sont des objets de type :

- « iTableData » pour une représentation tabulaire des données
- « iTreeData » pour une représentation hiérarchique des données
- « iBinaryData » pour des données binaires telle qu’une image
- « iTextData » pour des données sous forme d’un texte non structuré
- « iStreamData » pour des flux de données

Les traitements métier deviennent ainsi indépendants des technologies de stockage et des formats de sérialisation simplifiant l’adaptation aux changements stratégiques (et donc la souveraineté).

Leur responsabilité se limite à la mise en œuvre des règles métier.

3.5.1 Normalisation et dénormalisation

La conversion de données physiques en abstractions logiques nécessite plus qu'une simple validation structurelle.

L’architecture proposée doit également normaliser les informations.

La normalisation peut inclure :

- conversion du codage des caractères
- analyse des nombres, indépendamment de leur culture
- représentations standardisées des dates et heures
- saisie de données logiques
- représentations canoniques

Cette normalisation garantit que chaque traitement métier reçoit des données cohérentes quelles que soient les technologies utilisées pour les produire.

A l’inverse, une fois le traitement terminé, l’architecture proposée effectue l’opération inverse.

Les données logiques sont reconverties dans leurs représentations physiques en appliquant les Contrats de Données de sortie avant publication.

3.5.2 Abstraction du transport de données

Savoir à quoi devraient ressembler les données n’est qu’une partie du problème.

L’architecture proposée doit également savoir comment les obtenir et les publier.

Cette responsabilité appartient à l'abstraction des « Transports de Données (Data Transport) ».

Les implémentations de transport fournissent des opérations de lecture et d'écriture spécifiques au stockage pour les systèmes de fichiers, les bases de données, les API, le stockage cloud et les intermédiaires (message brokers).

Les traitements métier ignorent totalement ces détails de mise en œuvre.

Cette séparation améliore la portabilité tout en empêchant les informations d'identification sensibles d'être exposées au code métier.

3.6 Données supplémentaires

Le traitement métier peut également nécessiter des :

- données de référence
- paramètres de configuration
- secrets

3.6.1 Données de référence

Les données de référence sont accessibles via des « Gestionnaires de Ressource (Resource Managers) » dédiés.

Ces données peuvent évoluer entre chaque exécution et sont généralement en lecture seule.

Les traitements métier sont chargés d’interpréter les éventuelles conséquences fonctionnelles résultant des modifications de ces données.

3.6.2 Les paramètres de configuration

Dans l’architecture proposée, les paramètres de configuration sont stockés dans la « Définition du Processus Métier ».

La définition elle-même étant signée, ces paramètres héritent des mêmes garanties d’intégrité.

3.6.3 Les secrets

Les secrets suivent un modèle différent.

Dans l’architecture proposée, les traitements métier ne stockent jamais directement les mots de passe ou les clés API.

Au lieu de cela, ils reçoivent les identifiants des secrets.

L’architecture proposée résout ces identifiants via un « iSecretProvider », récupérant les informations d'identification réelles à partir d'un mécanisme de stockage sécurisé tel qu'un coffre-fort numérique.

Dans la mesure du possible, les transports consomment directement ces secrets, minimisant ainsi leur exposition au sein du code métier en réduisant leur durée de vie en mémoire et favorisant leur indépendance et réutilisation.

3.7 Classification des données

La gouvernance des données s'appuie de plus en plus sur des modèles de classification basés sur des taxonomies d'entreprise.

Chaque processus métier et chaque jeu de données échangé peut donc être associé à une ou plusieurs classifications décrivant des propriétés telles que :

- confidentialité
- sensibilité
- champ d'application réglementaire
- les exigences de rétention
- criticité métier

Ces classifications font partie de la Définition des Processus Métier plutôt que du code métier lui-même.

Étant donné que l’architecture proposée analyse déjà chaque ensemble de données lors de la normalisation et dénormalisation, elle peut éventuellement effectuer une classification automatique.

Les expressions régulières, les dictionnaires ou les classificateurs spécialisés peuvent identifier des caractéristiques telles que des informations personnelles, des données de paiement, des identifiants de soins de santé ou tout autre contenu réglementé.

Les classifications qui en résultent deviennent des preuves supplémentaires enregistrées lors de l'exécution.

Elles peuvent ensuite être utilisées par des plateformes de gouvernance, des catalogues de données ou des outils de conformité sans que le traitement métier n'implémente lui-même une logique de classification.

3.8 Cycle de vie des données

L’architecture proposée vérifie l'intégrité des données au moment où elles sont consommées.

Cette vérification garantit que le traitement repose sur une information fiable.

Elle ne dispense toutefois pas d'une surveillance continue des données lorsqu'elles demeurent stockées sur des supports susceptibles d'être modifiés ou altérés.

Ces altérations peuvent être dues à :

- une corruption disque
- une erreur de sauvegarde
- une mauvaise restauration
- une suppression partielle
- une modification malveillante
- une erreur d'exploitation

L'intégrité d'une donnée ne constitue pas une propriété acquise au moment de sa production. Elle doit être préservée tout au long de son cycle de vie, depuis sa création jusqu'à son archivage ou sa destruction.

L’architecture proposée peut exécuter des contrôles d’intégrité périodiques via un traitement dédié afin d’identifier au plus tôt une altération et d’agir en conséquence.

On doit donc distinguer les contrôles liés à l'exécution d'un traitement des contrôles patrimoniaux, réalisés indépendamment de toute exécution métier. Les premiers garantissent la confiance nécessaire à la décision immédiate ; les seconds contribuent à préserver la confiance dans le patrimoine informationnel de l'organisation au fil du temps.

Les politiques de gouvernance des données peuvent également différentier les contrôles à effectuer et leur périodicité en fonction de la nature des données et si elles se trouvent sur un support de stockage immuable ou non.

Elles peuvent, par exemple, préciser :

- Si support modifiable : Contrôle périodique recommandé
- Si support immuable : Contrôle uniquement à la restitution

3.9 Conclusion du chapitre

Établir la confiance dans le traitement métier commence par établir la confiance dans les données.

L’architecture proposée sépare donc les préoccupations traditionnellement liées :

- validation de l'intégrité
- validation de la structure
- normalisation et dénormalisation de la représentation physique

- abstraction du transport
- protection des secrets
- classification

Les traitements métier ne se soucient plus de la provenance des données ni de la manière dont elles sont représentées.

Ils reçoivent simplement des données dont les garanties requises sont déjà établies.

Cette séparation améliore non seulement la réutilisation et la portabilité des traitements métier, mais transforme également la gouvernance des données en une capacité opérationnelle plutôt qu'en un exercice purement documentaire.

Le chapitre suivant étend la même philosophie en établissant la confiance dans les traitements métier.

Chapitre 4 – Établir la confiance dans l’exécution

La confiance dans les données constitue une condition nécessaire à la fiabilité d’un traitement, mais elle n’est pas suffisante. Une donnée dont l’intégrité, l’origine et la conformité ont été établies peut néanmoins être traitée par un composant différent de celui qui était attendu, par une version non autorisée, ou dans des conditions ne respectant plus les politiques définies par l’organisation.

Établir la confiance dans l’exécution consiste donc à répondre à une autre question : **comment démontrer qu’un traitement a été réalisé par les composants attendus, dans les conditions prévues et conformément aux garanties exigées ?**

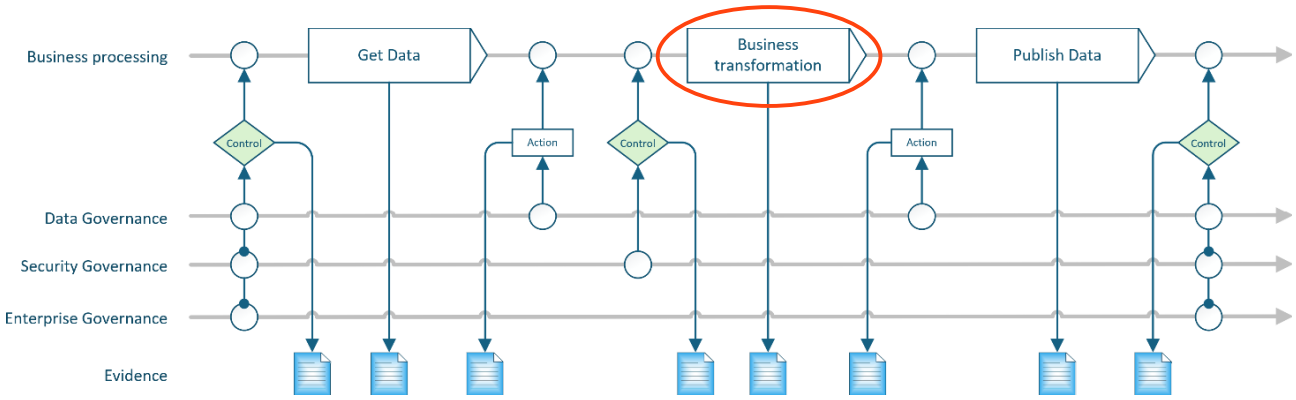
Cette question prend une importance particulière dans des systèmes d’information composés de nombreux services, bibliothèques et dépendances dont les cycles de vie évoluent indépendamment. Autoriser un traitement ne peut plus reposer uniquement sur le fait qu’un composant a été déployé ou qu’il appartient à un environnement considéré comme fiable. La confiance doit être établie au moment où l’exécution est demandée et maintenue pendant les étapes qui la composent.

Cela suppose de connaître précisément ce qui va être exécuté, d’en vérifier l’identité et l’intégrité, mais également de s’assurer que son comportement attendu est compatible avec les politiques applicables. Il faut ensuite maîtriser les conditions de l’exécution, contrôler ses différentes étapes et conserver les éléments permettant de démontrer a posteriori ce qui s’est réellement produit.

L’exécution devient ainsi elle-même un objet de gouvernance. Son autorisation ne résulte plus implicitement de la confiance accordée à un serveur, à une application ou à une équipe : elle résulte de **garanties vérifiables établies avant, pendant et après le traitement**.

Cette approche conduit également à une règle essentielle : lorsqu’une garantie indispensable ne peut être établie, l’exécution ne doit pas être considérée comme implicitement autorisée. Selon la politique définie par l’organisation, son interruption peut alors être préférable à la poursuite d’un traitement dont les conditions de confiance ne sont plus démontrables.

Ce chapitre présente les mécanismes permettant de construire cette confiance dans l’exécution : identification et contrôle des composants, maîtrise de leur comportement, contrôle de leurs dépendances, gestion transactionnelle, traçabilité des changements d’état et production des preuves permettant de reconstituer et de vérifier l’exécution.



4.1 Du code au manifeste de comportement

Plusieurs approches existent aujourd’hui pour établir la confiance dans l’exécution d’un logiciel.

Les organisations s'appuient généralement sur :

- l'analyse statique du code ("SAST")
- l’analyse dynamique du code (DAST)
- des revues de code manuelles ou assistées par IA
- des technologies de conteneurisation et de sandboxing

Chacune de ces approches répond à une partie du problème.

Les analyses statique et dynamique tentent d'identifier les vulnérabilités.

Note : Les analyses statique et dynamique ne sont généralement exécutées que lors des changements et/ou lors du déploiement de l’application mais plus durant son exécution.

Les conteneurs tentent d'isoler leurs conséquences.

4.1.1 Une approche fondée sur le comportement

L’architecture proposée suit une philosophie différente.

Son objectif n’est pas de prouver que le code métier est exempt de défauts.

Elle ne cherche pas non plus à remplacer les solutions SAST ou DAST.

Au lieu de cela, **elle vise à démontrer que le code en cours d'exécution présente uniquement des comportements préalablement identifiés, validés et acceptés**.

4.1.2 Génération du manifeste de comportement

Avant la publication, chaque traitement métier est analysé pour identifier ses capacités techniques.

L'analyse se concentre sur le comportement observable plutôt que sur la logique métier.

L’analyse inclue :

- les librairies référencés
- les espaces de noms utilisés
- l'utilisation de la réflexion
- les invocations OS natives (P/Invoke)
- la création de processus
- les accès au réseau
- les accès au système de fichiers
- les appels HTTP
- ou toute capacité jugée pertinente par l'organisation

Le résultat de cette analyse est un « Manifeste de Comportement (Behavior Manifest) ».

Le manifeste de comportement qui en résulte caractérise les capacités techniques du composant ; il s'agit d'un artefact descriptif, et non d'une évaluation de sécurité.

Une fois signé numériquement et référencé dans la « Définition du Processus Métier (Business Process Definition) », il devient la description faisant autorité du comportement d'exécution attendu.

4.1.3 Vérification à l'exécution

À chaque fois que l’architecture proposée doit charger un traitement métier, elle effectue à nouveau la même analyse comportementale.

Elle compare ensuite le manifeste nouvellement généré avec le manifeste de comportement publié.

Le chargement et l’exécution ne sont autorisés que si les deux manifestes sont identiques.

Par conséquent, la confiance n’est jamais placée uniquement dans une déclaration, elle est établie par la concordance entre une analyse indépendante et un manifeste préalablement validé et signé.

Toute divergence empêche immédiatement le chargement du traitement métier (tel que définit dans le paragraphe « 2.2 Une interprétation différente du Zero Trust »)

4.1.4 Aide à l'évaluation des risques

Le manifeste de comportement peut fournir également une aide précieuse à l’équipe en charge de la gestion des risques.

Deux versions d'un même traitement métier peuvent implémenter une logique métier différente tout en s'appuyant exactement sur les mêmes capacités techniques.

À l’inverse, des modifications apparemment mineures du code peuvent introduire des comportements techniques entièrement nouveaux.

Le manifeste de comportement peut permettre aux organisations de distinguer objectivement ces situations.

Si le manifeste reste inchangé, le périmètre de risque technique n’a pas évolué.

En fonction de la politique organisationnelle, l'approbation du déploiement peut donc suivre un processus de validation simplifié.

En revanche, si le manifeste change, les capacités techniques du composant ont évolué.

Cela représente une modification du périmètre de confiance et justifie une nouvelle évaluation des risques avant déploiement.

La génération automatique d'un rapport sur les différences de comportement entre les versions successives peut permettre aux équipes de gestion des risques et aux comités consultatifs sur les changements (CAB) de concentrer leurs examens exclusivement sur les modifications techniques significatives plutôt que d'inspecter l'intégralité du code source.

4.1.5 Avantages pour les parties prenantes

Le manifeste de comportement répond à trois objectifs complémentaires :

- Pour les développeurs, il fournit un retour immédiat sur les capacités techniques utilisées par leur implémentation. Ils s’engagent sur le comportement du traitement métier.
- Pour l’architecture proposée, il sert de référence comportementale fiable utilisée pour valider l'exécution
- Pour les équipes de gouvernance, il devient un artefact objectif soutenant la gestion du changement et l’analyse des risques

4.2 Les Gestionnaires de Ressource (Resource Managers)

L’exécution d’une logique métier nécessite plus que la simple lecture et écriture de données.

Le traitement métier s'effectue dans des environnements distribués où les pannes sont inévitables.

Les interruptions du réseau, les exécutions simultanées, les systèmes de stockage indisponibles, l'arrêt inattendu des processus et les publications partielles doivent tous être anticipés.

Pour cette raison, l’architecture proposée s’appuie sur des « Gestionnaires de Ressource (Resource Manager) » qui s’appuient sur des « Transport de Données (Data Transport) » en charge des spécificités liées aux environnements de stockage.

Un Gestionnaire de Ressource est chargé de maintenir la cohérence entre l’architecture proposée et une technologie de stockage associée.

Ses responsabilités comprennent :

- découvrir les éléments de travail disponibles
- réserver des ressources pour éviter les collisions lors de traitements simultanés
- lire des données d’entrée
- écrire des données de sortie
- participer à la coordination transactionnelle des ressources d'entrée et de sortie (dans la limite des garanties offertes par les systèmes concernés)
- restaurer les états précédents lorsque la récupération est requise
- libérer des ressources après une réussite
- archiver les éléments traités

L’architecture proposée délègue toutes les opérations spécifiques au stockage aux Gestionnaires de Ressource.

À l’inverse, les Gestionnaires de Ressource ne prennent jamais de décisions d’exécution.

Leur rôle se limite à mettre en œuvre les opérations requises par l’Orchestrateur de Garantie (définit au paragraphe suivant) tout en restant totalement indépendant de la logique métier.

Une fois les données lues, l’Orchestrateur de Garantie établit progressivement la confiance en vérifiant les signatures, en validant les contrats et en normalisant les informations avant d'invoquer le traitement métier.

Après exécution, le même principe s’applique à l’envers.

Les contrats de sortie sont validés, les données sont reconverties dans leur représentation physique, signées si nécessaire et enfin déléguées aux Gestionnaires de Ressource appropriés pour publication.

4.3 L'Orchestrateur de Garanties (Guarantee Orchestrator)

Avant d'examiner le modèle d'exécution lui-même, il est utile de définir la terminologie utilisée dans le reste de ce document.

- une « Garantie » est une propriété que l’architecture proposée s'engage à démontrer
- un « Contrôleur de Garanties (Guarantee Controller) » est chargé de vérifier une garantie spécifique et de produire les preuves liées
- l'« Orchestrateur de Garanties (Guarantee Orchestrator) » coordonne ces contrôleurs et détermine si l'exécution peut progresser
- enfin, le « Journal des Preuves (Evidence Journal) » enregistre toute garantie démontrée, permettant de reconstituer à tout moment l'exécution et l’auditer

L’architecture proposée n'exécute pas de traitement métier simplement parce qu'il a été demandé.

Elle exécute les traitements métier uniquement lorsque les garanties requises ont été établies avec succès.

L'Orchestrateur de Garanties maintient donc l’état d’exécution complet.

C’est le seul composant ayant une compréhension globale de :

- l'état d'exécution actuel
- les garanties déjà démontrées
- les garanties encore requises
- les stratégies de récupération possibles

Lorsqu’une étape est validée, il lance l’exécution de l’étape suivante.

Il peut s’agir :

- d’une acquisition de données via un Gestionnaire de Ressource,

- d’un contrôle via un Contrôleur de Garantie,
- du traitement métier,
- ou encore d’un convertisseur de données.

S’il ignore comment un Gestionnaire de Ressource accède aux données quel contrôle effectue un Contrôleur de Garantie et ce qu’effectue un traitement métier ou un convertisseur de données, il sait comment les invoquer et recevoir leurs résultats via des contrats d’implémentation établis.

Sa responsabilité relève exclusivement de l’orchestration.

L'exécution progresse via une machine à états déterministe.

Chaque transition nécessite :

- conditions préalables
- des preuves
- postconditions
- règles de récupération

Ce n’est qu’une fois que toutes les garanties requises ont été démontrées que l’Orchestrateur autorise la transition suivante.

Par exemple :

Lecture terminée (+ persistance de la Garantie)
▼
Signature vérifiée (+ persistance de la Garantie)
▼
Contrat vérifié (+ persistance de la Garantie)
▼
Normalisation terminée (+ persistance de la Garantie)
▼
Traitement métier
▼
(…)

Le flux d’exécution ne progresse donc jamais car une opération est terminée, il progresse parce que les garanties requises pour cette opération ont été démontrées et persistées.

4.4 Journal des Preuves (Evidence Journal)

Les sections précédentes ont présenté les éléments responsables de l’établissement de la confiance.

Nous avons vu que l'Orchestrateur de Garanties coordonne une machine à états dans laquelle chaque transition correspond à une garantie nouvellement établie et persistée.

Chaque transition réussie est immédiatement conservée sous la forme d'un « enregistrement de preuves » signé, lié cryptographiquement à la précédente.

Ensemble, ces enregistrements forment le Journal des Preuves d'une exécution métier.

Le Journal des Preuves répond à deux objectifs complémentaires :

- il fournit un historique vérifiable expliquant « ce qui s’est passé » et « pourquoi l’Orchestrateur de Garantie a jugé qu’il était sûr de continuer »
- il permet la reconstruction complète d'une exécution interrompue, permettant de reprendre ou de restaurer le contexte de traitement sans s'appuyer sur des hypothèses

Plutôt que de dépendre de la mémoire du processus, la continuité de l’exécution dépend entièrement de preuves persistées.

4.4.1 Exécution nominale

Dans des conditions idéales, l'exécution progresse selon la séquence suivante.

Étape	Action
01	Découvrir (les tâches à exécuter)
02	Verrouiller les données d’entrée (de la tâche à exécuter)
03	Lire les données d’entrée (le contenu de la tâche à exécuter)
04	Vérifier les données d’entrée (en termes de signature et de structure)
05	Normaliser les données d’entrée (convertir le contenu en un format logique et contrôler les valeurs)
06	Préparer (appeler l’action « prepare » du traitement métier)
07	Vérifier les données de sortie (en termes de structure)
08	Dénormaliser les données de sortie (convertir les sorties dans leur format physique et contrôler les valeurs)
09	Écrire les données de sortie (sur leur stockage final mais de manière verrouillée)
10	Signer les données de sortie
11	Engagement métier (appeler l’action « commit » du traitement métier)
12	Libération (des verrous des données de sortie)
13	Archiver la tâche et le Journal des Preuves lié à l’exécution

Chaque transition suit exactement le même schéma.

La vérification requise est effectuée par un Contrôleur de Garanties (Guarantee Controller) spécialisé.

Les preuves qui en résultent sont conservées par « l'enregistreur de preuves » (dans le Journal des Preuves).

L'Orchestrateur de Garanties valide que les exigences de transition ont été satisfaites.

Ce n’est qu’à ce moment-là que l’exécution pourra continuer.

L’Orchestrateur de Garantie n'avance donc jamais car une opération semble avoir réussi, il n'avance qu'après que la garantie correspondante ait été formellement démontrée et persistée.

4.4.2 Gestion des erreurs

Tous les échecs n’ont pas la même signification.

L’architecture proposée distingue trois catégories d'erreurs car elles nécessitent des réponses fondamentalement différentes.

Types d’erreur	Signification et exemples	Comportement
Technique	Une erreur technique indique que l’exécution est temporairement incapable de continuer. Exemples : • stockage indisponible • pannes de réseau • pannes d'infrastructures • conflit temporaire sur les ressources	Ces erreurs n’invalident pas la demande métier elle-même. L’exécution applique donc la politique de nouvelle tentative définie dans la Définition du Processus Métier. Les ressources d'entrée restent réservées pendant que les tentatives sont effectuées afin d'empêcher une exécution simultanée. Si la stratégie de nouvelle tentative est épuisée, l’exécution est transférée au traitement manuel.
Sécurité	Les erreurs de sécurité correspondent à des manquements dans l’établissement des garanties obligatoires. Exemples : • signatures numériques invalides • artefacts corrompus • composant applicatif modifié • incohérences manifestes de comportement • preuve d'exécution invalide	Chaque fois qu'une telle situation se produit, l’exécution est avortée. La demande est transférée pour enquête de sécurité. Contrairement aux pannes techniques, les erreurs de sécurité empêchent délibérément la relance automatique jusqu'à ce que la cause première soit comprise. Ce comportement reflète l'un des principes fondamentaux de l'architecture : « Chaque fois que la confiance ne peut être démontrée, l'exécution doit s'arrêter. »
Métier	Les erreurs métier indiquent que le traitement ne peut pas continuer car les exigences métier ne sont pas satisfaites. Exemples : • données de référence manquantes • violation les règles métier • informations métier incohérentes	L’exécution ne peut pas résoudre automatiquement une telle situation. L'exécution est suspendue et transférée en traitement manuel. Il est important de noter que le traitement manuel n’implique pas nécessairement l’annulation. La correction des informations métier sous-jacentes peut permettre à la demande d'être traitée avec succès.

Note : Un artefact est ici un produit ou document généré durant une exécution ou nécessaire à une exécution.

4.4.3 Récupération

L’arrêt inattendu d’un processus n’est pas en soi considéré comme un échec métier.

Cela interrompt simplement la démonstration des garanties.

Lorsque l'Orchestrateur de Garanties redémarre, il analyse le Journal des Preuves pour reconstruire le Contexte d'Exécution.

La reprise ne tente jamais de déduire ce qui aurait pu se produire.

Au lieu de cela, elle reprend à partir du dernier état démontré.

Selon la transition enregistrée, plusieurs situations peuvent se produire.

- Si l'exécution s'arrête avant que des effets secondaires externes ne se produisent, l'exécution reprend à partir de l'état de pré-exécution
- Si les ressources restent réservées, les gestionnaires de ressource appropriés rétablissent d'abord toutes les réservations en cours
- Si des sorties temporaires ont été écrites mais jamais validées :
  - Si le métier n'a pas été validé, elles sont supprimées selon les capacités de restauration fournies par les gestionnaires de ressource correspondants.
  - Si le métier a été validé, leur publication est finalisée par les gestionnaires de ressource correspondants.

La relance consiste donc à reconstruire la confiance plutôt qu'à rejouer aveuglément les opérations.

Le Journal des Preuves reste la seule source faisant autorité décrivant l'état de l'exécution.

4.4.4 Modèles de traitement métier

Les capacités de récupération dépendent également du comportement du traitement métier lui-même.

Le contrat d'implémentation d'un traitement métier comprend les trois actions distinctes « Prepare », « Commit », « Rollback » qui permettent au traitement métier de s'inscrire dans la transaction distribuée qui encapsule l'ensemble de l'exécution. Cette transaction vise à garantir que l'ensemble de l'exécution est soit finalisée correctement ou soit annulée sans impact sur les données.

En fonction du comportement du traitement métier dans les actions « Prepare », « Commit », « Rollback », ce traitement peut être classé dans les quatre modèles d'implémentation ci-dessous :

Modèle	Opérations			Commentaire
	1. « Prepare »	2. « Commit »	3. « Rollback »	
Fonction pure	Effectue uniquement un calcul. Elle ne produit aucun effet secondaire externe.	Applique les résultats	Ne fait rien	L'exécution peut être rejouée en toute sécurité chaque fois que nécessaire. C'est le modèle de mise en œuvre préféré.
Transaction compensable	Réserve des ressources	Valide ces réservations	Libère les ressources	De nombreuses API modernes suivent naturellement ce modèle.

	externes.			Tant que les réservations restent valables, les exécutions interrompues peuvent reprendre en toute sécurité.
Transaction idempotente	Produit déjà des effets observables.	Valide les effets	Annule les effets	Si « Prepare » produit des effets, répéter l'opération produit le même résultat final. L'exécution peut donc être donc rejouée sans introduire d'incohérences.
Transaction irréversible	Effectue immédiatement une action externe irréversible.	Ne fait rien	Ne fait rien	Dans cette situation, il n'est pas possible de démontrer de manière indépendante qu'une nouvelle exécution est sûre. La demande est donc transférée en traitement manuel. Il ne s'agit pas d'une limitation de l'architecture proposée mais d'une conséquence directe de sa philosophie. Lorsque la sécurité ne peut être démontrée, l'architecture proposée refuse délibérément de faire des hypothèses. L'exécution est redirigée vers le traitement manuel.

4.4.5 Limites

L'Orchestrateur de Garanties ne manipule jamais directement les données métier.

Au lieu de cela, il coordonne des « Contrôleurs de Garantie », des « Gestionnaires de Ressource », un convertisseur de données et le traitement métier.

Chaque acteur participe à une transaction distribuée via des contrats d'implémentation précis.

Nous avons vu que, quelle que soit la technologie de stockage sous-jacente, chaque Gestionnaires de Ressource doit être capable de :

- découvrir les tâches disponibles à accomplir
- réserver une tâche
- libérer les réservations
- marquer la réussite
- marquer l'échec
- archiver les tâches traités
- participer à la récupération

Ce contrat commun permet à des technologies de stockage hétérogènes de participer à un seul modèle d'exécution distribué tout en restant totalement indépendantes les unes des autres.

Une limite importante doit néanmoins être reconnue : l’Orchestrateur de Garanties ne peut pas inventer des garanties transactionnelles que les systèmes externes n’offrent pas.

L'environnement d'exécution ne peut coordonner les traitements métier que dans le cadre des garanties transactionnelles fournies par les systèmes externes.

Aussi, si un composant de traitement métier effectue plusieurs validations externes, ces opérations doivent elles-mêmes être protégées par un mécanisme transactionnel approprié.

Au sein d’une seule base de données relationnelle, cela est généralement simple.

Dans plusieurs bases de données – ou lors de l’interaction avec des services Web externes – cela peut devenir beaucoup plus complexe.

Lorsqu'aucun mécanisme de transaction distribué n'existe, la solution privilégiée consiste à décomposer le traitement en plusieurs traitements métier plus petits.

Chaque composant peut alors être exécuté indépendamment dans sa propre limite de garantie.

Plutôt que de tenter de coordonner des opérations irréversibles, l’architecture proposée coordonne une séquence de garanties démontrables.

Cette décomposition améliore considérablement la résilience tout en réduisant l’impact opérationnel des pannes.

4.5 Conclusion du chapitre

La confiance d’exécution ne peut pas se réduire à la vérification d’une signature numérique ou à la validation d’un composant logiciel.

Elle exige que chaque transition de l’exécution soit appuyée par des garanties démontrables.

L’architecture proposée n’orchestre donc pas simplement les traitements métiers, elle orchestre la construction de la confiance.

Chaque garantie s’appuie sur la précédente.

Chaque décision est appuyée par des preuves vérifiables.

Chaque interruption peut être analysée et récupérée en utilisant uniquement ces preuves.

L'exécution elle-même devient finalement la conséquence d'une chaîne ininterrompue de garanties démontrées.

Chapitre 5 – Établir la confiance grâce à des descriptions de processus déclaratifs

5.1 Pourquoi décrire un processus ?

Une description de processus n'est pas simplement un artefact de configuration.
Il s'agit d'une représentation déclarative d'un traitement métier dont le but premier est de permettre à l’architecture proposée de démontrer les garanties associées à une exécution.

Plutôt que de décrire uniquement « ce qui » doit être exécuté, elle décrit le « modèle de confiance » régissant cette exécution en spécifiant :

- les Contrats de Données
- les mécanismes de transport
- les secrets
- les garanties
- les signatures numériques requises
- le Manifeste de comportement
- les responsabilités
- les politiques de publication
- les politiques de récupération
- les politiques de rétention

La description du processus devient donc le fondement sur lequel la confiance peut être établie, vérifiée et finalement démontrée.

Dans la plupart des moteurs d'exécution, une description de processus répond à deux questions fondamentales :

- Que faut-il exécuter ?
- Où peut-on trouver les données requises ?

Même si ces informations sont suffisantes pour automatiser un traitement métier, elles ne suffisent pas à démontrer que l'exécution est conforme aux politiques de gouvernance de l'organisation.

Dans l’architecture proposée, la « Définition du Processus Métier » répond à un objectif plus large.

Elle décrit non seulement le processus métier lui-même, mais identifie également les garanties qui doivent être établies avant, pendant et après l'exécution.

L’objectif n’est donc pas simplement d’automatiser l’exécution, mais de rendre cette exécution « manifestement digne de confiance ».

5.2 Une description guidée par des garanties

La « Définition du Processus Métier » comporte toutes les informations utiles aux acteurs qui participent à une exécution afin que les garanties requises soient obtenues.

Ces acteurs sont des Gestionnaires de Ressource, des Contrôleurs de Garanties et un traitement métier.

L’architecture proposée fournit des capacités d'exécution génériques, mais elle n'a aucune connaissance intrinsèque des politiques métier, des règles de sécurité, des exigences réglementaires ou de la gouvernance organisationnelle.

Toutes ces décisions proviennent exclusivement de descriptions déclaratives.

Plutôt que d'intégrer une logique de gouvernance dans le moteur d'exécution, l’architecture proposée interprète les garanties définies par la Définition du Processus Métier.

En conséquence, les politiques de gouvernance peuvent évoluer librement.

5.3 Diviser la description en plusieurs documents

Tous les aspects d’une Définition du Processus Métier n’évoluent pas au même rythme.

Par exemple :

- L'identité d'un traitement change rarement
- Les paramètres « runtime » évoluent au fur et à mesure que l'infrastructure évolue
- Les « manifestes de comportement » évoluent à chaque fois que le comportement ou la portée d'un traitement métier change
- Les « échanges de données » évoluent au fur et à mesure de l'évolution des Contrats de Données et du traitement métier

Regrouper toutes ces préoccupations dans un seul document les forcerait artificiellement à partager le même cycle de vie, alors même que chacune appartient à un domaine de gouvernance différent.

En les séparant en artefacts indépendants, chaque discipline peut évoluer selon ses propres processus de gouvernance sans impacter les autres.

Cette indépendance est essentielle pour préserver l’agilité organisationnelle tout en maintenant la confiance.

Chaque fragment de la Définition du Processus Métier a son propre :

- propriétaire
- signature numérique
- cycle de vie
- processus de validation

L’architecture proposée assemble ensuite ces artefacts gouvernés de manière indépendante dans un environnement d'exécution fiable.

5.4 Signatures numériques

Chaque artefact participant au modèle d'exécution est signé numériquement à l'aide d'une cryptographie asymétrique.

Chaque autorité de gouvernance est responsable de la protection de sa propre clé de signature privée.

En pratique, les signatures sont généralement appliquées dans le cadre du processus de déploiement ou de publication de l'organisation.

Différents artefacts peuvent être signés par différentes autorités, selon le modèle de gouvernance de l'organisation.

Lorsque l’architecture proposée est exécutée, elle commence par vérifier la signature de chaque artefact à l'aide de la clé publique de confiance associée à son domaine de gouvernance.

Les domaines de confiance typiques incluent :

- Métiers (Business)
- Opérations (Operation)
- Développement (Development)
- Sécurité (Security)
- Plateforme (Platform)
- Données (Data)
- Entreprise (Enterprise)

Un mappage typique pourrait être :

Artefact	Responsabilité principale	Domaine de confiance
Identité	Équipe métier	Métiers (Business)
Runtime	Opérations	Opérations (Operation)
Traitement Métier	Équipe de Développement	Développement (Development)
Behavior Manifest	Équipe de Développement + Sécurité	Sécurité (Security)
Échanges de données	Gouvernance des données	Données (Data)

Une signature numérique sert bien au-delà de la protection de l’intégrité d’un fichier.

Elle représente un « engagement vérifiable » pris par l'autorité responsable de cet artefact.

L’architecture proposée ne vérifie pas simplement qu'un artefact n'a pas été modifié, elle vérifie également "quel domaine assume la responsabilité de son contenu".

La confiance devient donc une propriété organisationnelle, et non pas seulement une propriété cryptographique.

5.5 Implication de la publication d'artéfacts

Lorsque l’architecture proposée est exécutée, elle charge toujours la dernière version des artéfacts et vérifie l'intégrité de l'ensemble publié avant le début de toute transaction.

Cette intégrité garantit que le processus de publication s'est déroulé avec succès et qu'aucun artefact n'a été modifié après la publication.

Lors de la publication, la plateforme scelle la version complète en signant numériquement la collection d'artefacts avec la clé privée de la plateforme.

Cette signature de plateforme certifie que la version publiée est complète, cohérente avec les processus de déploiement et prête à être exécutée.

Une fois l'exécution démarrée, l’architecture proposée continue d'utiliser le même ensemble immuable d'artefacts tout au long de la durée de vie de son « Contexte d'Exécution ».

Si une nouvelle version d'un artéfact est publiée alors que les transactions sont encore en cours de traitement, l’architecture proposée détecte le changement, termine toutes les transactions en cours avant de s'arrêter correctement.

Au redémarrage, l’architecture proposée crée un nouveau Contexte d'Exécution à partir de l'ensemble des artéfacts à jour.

Ce comportement garantit que chaque transaction est exécutée dans un environnement d'exécution stable et entièrement vérifiable.

5.6 Le moteur d'exécution de la machine à états

L’architecture proposée ne dicte pas la manière dont une organisation gère ses processus métier.

Chaque organisation définit ses propres politiques de gouvernance, mécanismes de contrôle et garanties opérationnelles.

Celles-ci évoluent continuellement en réponse aux nouvelles réglementations, aux menaces émergentes, aux politiques internes et aux objectifs métier changeants.

Si ces règles de gouvernance étaient intégrées directement dans le moteur d’exécution, chaque changement de politique nécessiterait des modifications du moteur d’exécution.

Le moteur d’exécution deviendrait étroitement liée à la gouvernance organisationnelle et perdrait sa capacité à servir d’environnement d’exécution générique.

L’architecture proposée présentée dans cet article adopte une approche différente.

Elle ne contient aucune politique de gouvernance intégrée.

Au lieu de cela, elle interprète une machine à états déclarative qui décrit le cycle de vie des transactions, les garanties à établir et les contrôles à effectuer.

Cette description est appelée « Définition du pipeline de garanties ».

Étant donné que la gouvernance est exprimée de manière déclarative plutôt que mise en œuvre par programme, les politiques peuvent évoluer indépendamment du moteur d’exécution.

Le moteur d’exécution agit donc comme un « interprète d'une politique », et non comme une « implémentation d'une politique ».

5.6.1 La définition de la machine à états

Au démarrage, l’architecture proposée charge à la fois la « Définition du Processus Métier » et la « Définition du pipeline de garanties ».

Bien que ces deux descriptions soient interprétées ensemble, elles répondent à des objectifs fondamentalement différents.

La « Définition du Processus Métier » décrit « ce que fait le processus métier ».

La « Définition du pipeline de garanties » décrit « comment ce processus métier doit être régi ».

La « Définition du pipeline de garanties » précise entre autres :

- les étapes du cycle de vie des transactions
- les Gestionnaires de Ressource à invoquer
- les Contrôleurs de Garantie à exécuter
- la séquence d'exécution
- les Preuves qui doivent être produites

Cette définition appartient à l'équipe de gouvernance de l'organisation, qui traduit les politiques de gouvernance en une chaîne d’actions et de contrôles exécutables.

Son cycle de vie est donc indépendant de celui du processus métier lui-même.

Cette séparation permet aux politiques de gouvernance d'évoluer sans modifier la logique métier et, à l'inverse, permet aux processus métier d'évoluer sans nécessiter de changements dans les politiques de gouvernance.

5.6.2 Une chaîne plutôt qu'un graphe

Contrairement à un moteur de Workflow traditionnel, la machine à états ne décrit pas la logique métier.

Au lieu de cela, elle décrit les activités de gouvernance technique qui accompagnent chaque transaction.

Le modèle d'exécution est intentionnellement linéaire.

Une transaction progresse à travers une séquence déterministe d’opérations impliquant :

- Contrôleurs de Garantie
- Gestionnaires de Ressource
- Traitement métier
- Transformateurs de données, si nécessaire

L’absence de graphe d’exécution complexe est un choix architectural délibéré.

Un modèle d’exécution linéaire est nettement plus facile à comprendre, plus facile à auditer et considérablement plus simple à récupérer après une panne.

Plus important encore, il garantit que chaque transaction suit le même chemin d’exécution entièrement déterministe et entièrement traçable.

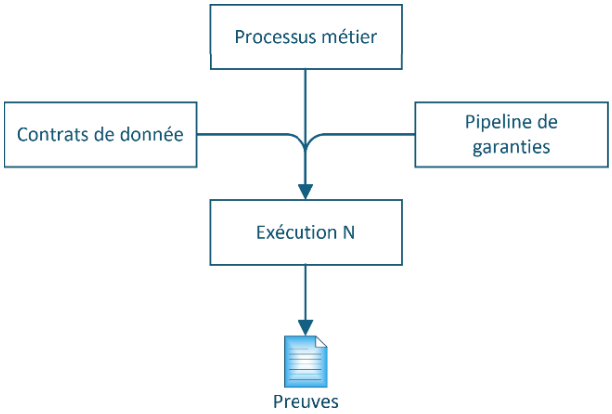
Les décisions métier restent entièrement sous la responsabilité de la composante métier.

La machine d'état est uniquement responsable des activités de gouvernance nécessaires pour établir et démontrer la confiance tout au long du cycle de vie des transactions.

5.6.3 Au moins quatre cycles de vie indépendants

Chaque transaction représente la convergence d’au moins les quatre cycles de vie indépendants suivants :

- la Définition des Processus Métiers
- la Définition du pipeline de garanties
- des Contrats de Données
- la demande métier en cours de traitement



Chacun évolue indépendamment.

Le même processus métier peut donc être exécuté sous des versions successives de la politique de gouvernance sans nécessiter aucune modification du processus métier lui-même.

A l'inverse, plusieurs versions d'un processus métier peuvent être régies par la même politique.

Le « Journal des Preuves » fait le lien entre ces dimensions.

Pour chaque transaction, il enregistre :

- la version de la Définition des Processus Métiers
- la version de la Définition du pipeline de garanties
- La version des Contrats de Données
- le Contexte d'Exécution associé
- les Preuves produites tout au long de l'exécution

Des années après son exécution, il reste donc possible de reconstituer l'environnement exact dans lequel une décision métier a été prise (à la condition, bien sûr, de conserver ces journaux).

Cette fonctionnalité offre un niveau de traçabilité difficile à atteindre lorsque les politiques de gouvernance sont intégrées directement dans le code de l'application.

5.7 Assurance qualité de l'exécution

L'architecture proposée ne crée pas de confiance.

Elle ne définit pas non plus les politiques de gouvernance, ne valide pas les décisions métier et ne garantit pas l'exactitude d'une mise en œuvre.

Sa responsabilité est plus fondamentale.

Elle vérifie que chaque acteur impliqué dans l'exécution d'une transaction a rempli les engagements qui relèvent de son propre domaine de responsabilité.

La confiance émerge donc de la convergence d'artefacts gouvernés indépendamment plutôt que d'hypothèses intégrées au moteur d'exécution.

5.7.1 Responsabilités indépendantes

Les systèmes d'entreprise modernes sont développés et exploités par plusieurs équipes spécialisées.

- Les experts métier définissent les processus métier
- Les équipes de développement implémentent les traitement métier
- Les équipes de sécurité autorisent le comportement des logiciels
- Les équipes de gouvernance des données publient des Contrats de Données
- Les équipes opérationnelles définissent les environnements d'exécution
- Les équipes de plateforme déploient des instances de l'architecture proposée

Chaque discipline possède un corpus de connaissances distinct et applique ses propres processus de gouvernance.

Tenter de centraliser ces responsabilités dans un seul artefact ou dans un seul processus d’approbation créerait non seulement des goulots d’étranglement organisationnels, mais affaiblirait également la responsabilisation.

Au lieu de cela, chaque discipline publie son propre artefact de confiance.

Chaque artefact suit son propre cycle de vie, processus de validation, flux de publication et signature numérique.

La propriété reste distribuée, tandis que la confiance devient composable.

5.7.2 Création d'un Contexte d'Exécution fiable

Avant le début d'une transaction, le moteur d’exécution rassemble tous les artefacts publiés dans un seul « Contexte d'Exécution » immuable.

Ce contexte représente l'ensemble complet d'hypothèses sous lesquelles la transaction sera exécutée.

Avant d'accepter le contexte, le moteur d’exécution vérifie que :

- chaque artefact requis est présent
- chaque signature numérique est valide
- chaque artefact provient d'une autorité de confiance
- toutes les versions déclarées sont mutuellement compatibles
- toutes les dépendances peuvent être résolues
- le contexte assemblé est cohérent en interne

Ce n’est qu’une fois ces vérifications réussies que l’exécution commence.

Si une vérification échoue, la transaction est rejetée avant toute traitement métier.

Ce principe garantit que la confiance est établie « avant » l’exécution plutôt que déduite après.

5.7.3 Assurance qualité de l'exécution

Le moteur d’exécution effectue ce que l'on peut décrire comme une « Assurance Qualité d’Exécution ».

Contrairement à l'Assurance Qualité traditionnelle qui évalue les logiciels avant leur déploiement, l'Assurance Qualité d’Exécution évalue les conditions dans lesquelles chaque transaction individuelle est exécutée.

Son objectif n’est pas de déterminer si une décision métier est correcte.

Elle ne tente pas non plus de valider la qualité de la conception des logiciels, de la modélisation des données ou des politiques de gouvernance.

Ces responsabilités incombent aux disciplines qui ont produit les artefacts correspondants.

Au lieu de cela, le moteur d'exécution vérifie que la transaction s'exécute dans un environnement qui satisfait à chaque engagement déclaré.

L'Assurance Qualité d'Exécution répond ainsi à des questions telles que :

- la version approuvée du traitement métier a-t-elle été exécutée ?
- les Contrats de Données applicables étaient-ils disponibles ?
- le pipeline de gouvernance requis a-t-il été appliqué ?
- tous les contrôles obligatoires ont-ils été effectués avec succès ?
- tous les éléments de preuve requis ont-ils été produits ?

Si la réponse à l'une de ces questions est négative, la transaction ne peut pas être considérée comme totalement fiable.

L'Assurance Qualité d'Exécution certifie que la transaction a été exécutée dans des conditions vérifiées et démontrables.

5.7.4 Un principe architectural

Cette approche reflète un principe architectural plus large.

La confiance d'entreprise ne peut pas être déléguée à un seul composant.

Elle doit émerger d'engagements gouvernés de manière indépendante et vérifiables objectivement.

L'architecture proposée n'est donc pas la source de confiance.

Elle est le mécanisme qui rassemble les artefacts fiables, vérifie leur intégrité et leur compatibilité, exécute leurs descriptions déclaratives et produit les preuves requises pour démontrer que chaque engagement déclaré a été honoré.

La confiance n'est pas intégrée à l'architecture proposée, elle est établie par des engagements vérifiables et démontrée par l'exécution.

5.8 Au-delà de l'exécution : la valeur des descriptions déclaratives

Les descriptions déclaratives sont souvent perçues simplement comme des artefacts de configuration consommés par un moteur d'exécution.

En réalité, ils représentent quelque chose de bien plus précieux parce qu'en décrivant formellement à la fois l'intention métier et la gouvernance opérationnelle, ils deviennent des actifs d'informations réutilisables qui s'étendent bien au-delà de l'exécution.

Le moteur d’exécution de l’architecture proposée n’est que leur premier consommateur.

Les mêmes descriptions peuvent également aider l'architecture, la gouvernance, la sécurité, la conformité, les opérations et l'analyse d'entreprise.

La documentation n'est plus produite séparément du système.

Elle devient un sous-produit naturel des descriptions nécessaires à l'exécution.

5.8.1 La Définition du Processus Métier en tant qu'actif de l'entreprise

La Définition du Processus Métier fournit une représentation structurée et faisant autorité de l'intention métier.

Parce qu'elle est à la fois exécutable et déclarative, elle devient une source d'informations fiable pour un large éventail de parties prenantes.

- Les architectes peuvent analyser les dépendances entre les capacités métier et les composants techniques
- Les équipes opérationnelles peuvent comprendre comment les activités métier sont exécutées sans examiner le code de l'application
- Les équipes de gouvernance des données peuvent identifier quelles informations sont consommées, produites ou référencées par chaque processus
- Les équipes de sécurité peuvent déterminer quels traitements métier participent aux opérations critiques

L’analyse d’impact devient considérablement plus simple.

Lorsqu'un traitement métier, un Contrat de Données ou une dépendance externe évolue, chaque processus métier affecté peut être identifié directement à partir de sa description déclarative.

La définition des processus métier sert donc non seulement d’artefact d’exécution, mais également de source vivante et continuellement entretenue de connaissances architecturales.

Contrairement à la documentation traditionnelle, elle reste synchronisée avec le système déployé car elle est nécessaire à l'exécution elle-même.

5.8.2 La Définition du Pipeline de Garantie en tant qu’actif de gouvernance

La Définition du Pipeline de Garantie offre le même niveau de visibilité pour la gouvernance opérationnelle.

Plutôt que de documenter les politiques de gouvernance dans des documents narratifs, l'organisation publie une description exécutable des contrôles qui régissent l'exécution des transactions.

Cela permet d’analyser la gouvernance avec la même rigueur que l’architecture logicielle.

Par exemple, les organisations peuvent :

- visualiser la chaîne complète des contrôles appliqués à chaque transaction
- comparer les politiques de gouvernance entre les versions
- identifier l'impact de l'introduction ou de la suppression d'un contrôle
- vérifier le respect des exigences internes ou réglementaires
- détecter les règles de gouvernance manquantes ou incohérentes
- simuler l'évolution de la politique avant le déploiement

Parce que la gouvernance est exprimée de manière déclarative, elle devient inspectable, analysable et versionnable.

La gouvernance opérationnelle évolue de la documentation à un actif d'entreprise qui peut être continuellement géré et amélioré.

5.9 De l'exécution à la connaissance de l'entreprise

À mesure que les descriptions déclaratives s’accumulent, elles forment naturellement un ensemble de connaissances chaînées.

Les contextes d'exécution lient tous ces artefacts dans des environnements de transaction immuables.

Les preuves relient chaque transaction exécutée aux versions exactes des artefacts qui la régissaient.

Ces relations existent déjà dans l’architecture proposée.

Les représenter explicitement transforme une collection d’artefacts techniques en un graphe de connaissances d’entreprise.

Un tel graphe permet aux organisations de répondre à des questions qui nécessitent traditionnellement une enquête manuelle approfondie.

Par exemple :

- quels processus métiers dépendent d'un Contrat de Données particulier ?
- quelles transactions ont été exécutées dans le cadre d'une politique de gouvernance spécifique ?
- quels traitements métier ont été autorisés par un manifeste de comportement donné ?
- quelles règles de gouvernance s'appliquaient à une décision métier particulière ?
- quels artefacts déployés seraient affectés par un changement proposé ?

Étant donné que chaque relation provient de descriptions déclaratives fiables, le graphe de connaissances qui en résulte reste objectif, traçable et continuellement synchronisé avec la plateforme opérationnelle.

Note : Le graphe ne peut être objectif que relativement à la complétude et à la justesse de ce qui a été déclaré et observé. Une dépendance extérieure non déclarée ou non observable peut manquer.

L’architecture proposée ne génère pas explicitement ces connaissances.

Elle interprète simplement des descriptions déclaratives dont les relations forment naturellement une représentation interconnectée de l'entreprise.

Le Graphe de Connaissances d’Entreprise n’est donc pas un produit distinct.

Il s’agit d’une propriété émergente d’une architecture déclarative fondée sur des descriptions fiables.

5.10 Conclusion du chapitre

Les plates-formes d'exécution traditionnelles se concentrent principalement sur l'automatisation.

L’architecture proposée dans cet article poursuit un objectif plus large.

En séparant l'intention métier de la gouvernance opérationnelle et en exprimant les deux au moyen de descriptions déclaratives gouvernées de manière indépendante, l’architecture proposée établit un environnement d'exécution fiable plutôt qu'un simple moteur d'exécution.

Chaque discipline participante apporte ses propres artefacts fiables.

L’architecture proposée les assemble, vérifie leur intégrité et leur compatibilité, interprète leur intention et enregistre les preuves nécessaires pour démontrer que chaque engagement déclaré a été rempli.

La confiance n’est donc pas intégrée à l’architecture proposée, elle résulte d’engagements vérifiables établis par des autorités indépendantes.

Le résultat est un environnement d'exécution qui non seulement automatise les processus métiers, mais offre également une confiance démontrable, une traçabilité complète et un ensemble de connaissances organisationnelles en constante évolution.

Dans ce modèle, les descriptions déclaratives deviennent bien plus que des artefacts d’exécution.

Ils deviennent la base sur laquelle repose une exécution fiable de l’entreprise et permettent l’émergence de capacités de plus haut niveau telle que le Graphe de Connaissances d’Entreprise.

Chapitre 6 – Adopter une gouvernance démontrable

6.1 Une transformation progressive

L'architecture proposée tout au long de ce livre blanc ne doit pas être considérée comme une solution tout ou rien.

L’objectif est de présenter une manière innovante de penser les systèmes d’information.

Les principes proposés peuvent être adoptés progressivement.

Chaque garantie introduite réduit l’incertitude.

Chaque élément de preuve renforce la confiance.

Les organisations peuvent donc évoluer à leur rythme en fonction de leurs priorités, des exigences réglementaires et de leur maturité technique.

La gouvernance démontrable n’est pas un produit.

C'est une approche architecturale.

6.2 Commencez par décrire

La plupart des organisations possèdent déjà des applications métiers.

Les remplacer est rarement réaliste.

La première étape consiste donc à décrire plutôt qu’à modifier.

Les processus métier doivent progressivement être documentés au moyen d'artefacts structurés et exploitables décrivant :

- les traitements métier
- les données échangées
- les contrats attendus
- les dépendances techniques
- les politiques d'exécution

À ce stade, aucun moteur d’exécution n’est requis.

Le simple fait de rendre explicites les connaissances implicites représente déjà une amélioration significative.

La Définition du Processus Métier devient le premier artefact de gouvernance.

6.3 Introduire les signatures numériques

Une fois les artefacts critiques identifiés, ils devraient progressivement devenir vérifiables.

Les signatures numériques représentent l'un des moyens les plus simples d'établir la confiance.

Les organisations peuvent commencer par signer :

- les Définitions du Traitement Métier
- les traitements métier (DLL et exécutables)
- les Contrats de Données

A ce stade, les signatures peuvent simplement être vérifiées lors du déploiement.

L'exécution elle-même reste inchangée.

Néanmoins, les engagements organisationnels deviennent objectivement vérifiables.

6.4 Caractériser le comportement

Les processus de déploiement traditionnels valident généralement les logiciels via des tests.

L'architecture proposée introduit une perspective supplémentaire.

Avant le déploiement, chaque Composant Applicatif est caractérisé.

Ses capacités techniques sont résumées dans un Manifeste de Comportement.

L’objectif n’est pas de rejeter automatiquement les logiciels, il s’agit de comprendre ce que le logiciel est techniquement capable de faire et de le formaliser dans un artéfact géré (le Manifeste de Comportement).

Les évaluations des risques deviennent donc nettement plus objectives.

Les efforts de révision peuvent se concentrer sur les changements qui modifient réellement le périmètre de confiance technique.

6.5 Produire des preuves

De nombreuses organisations collectent déjà des journaux opérationnels.

Cependant, les journaux opérationnels décrivent principalement les événements.

L'architecture proposée recommande plutôt de produire des « preuves ».

Au lieu d'enregistrer uniquement le fait qu'un composant s'est exécuté, l'environnement d'exécution consigne la preuve que les conditions d'exécution requises étaient remplies.

Cette distinction change fondamentalement la finalité des journaux opérationnels.

Les preuves deviennent le fondement de la gouvernance.

Les audits ne reposent plus exclusivement sur la documentation.

Ils s’appuient sur des preuves d’exécution vérifiables.

6.6 Orchestrer les garanties

Ce n’est qu’une fois les garanties, les actions, contrôles et les preuves introduits que le moteur d’exécution devient vraiment précieux.

Son rôle est de coordonner les garanties déjà définies.

Le moteur d’exécution ne remplace pas la gouvernance.

Il l’opérationnalise.

Il vérifie en permanence que toutes les garanties requises ont été démontrées avant d'autoriser la prochaine étape d'exécution.

L’exécution devient donc la conséquence de la gouvernance plutôt qu’une activité technique indépendante.

6.7 Gouverner les garanties plutôt que les applications

La gouvernance traditionnelle examine, audite et certifie les applications.

L'architecture proposée déplace cette perspective.

Au lieu de régir directement les applications, les organisations régissent les garanties attendues de ces applications.

Les traitements métier deviennent interchangeables.

Ce qui compte vraiment, c'est que chaque exécution démontre :

- l'intégrité des données
- l'intégrité du code
- respect des Contrats de Données
- comportement technique validé
- publication contrôlée
- exécution récupérable

L’objet de la gouvernance n’est donc plus le logiciel lui-même mais l’ensemble des garanties que les logiciels doivent continuellement démontrer.

6.8 Un modèle de maturité

Les organisations se transforment rarement du jour au lendemain.

Une gouvernance démontrable peut donc être adoptée progressivement.

Un modèle de maturité possible est présenté ci-dessous.

Niveau	Objectif
1	Décrire les processus métier
2	Introduire les Contrats de Données
3	Signer numériquement les artefacts critiques
4	Caractériser le comportement du logiciel
5	Produire des preuves d’exécution
6	Orchestrer automatiquement les garanties
7	Gouverner l’entreprise grâce à des preuves vérifiables

Chaque niveau fournit une valeur mesurable indépendante.

Les organisations sont donc libres de s’arrêter à tout moment ou de poursuivre vers une gouvernance complète et démontrable.

6.9 Au-delà de l’architecture proposée

L’architecture proposée est indépendante de la technologie.

Toute plateforme d’exécution pourrait progressivement adopter des concepts similaires.

- Plateformes de conteneurs
- Plateformes d'intégration
- Moteurs de flux de travail
- Orchestrateurs de microservices
- Environnements d'exécution cloud natifs

L'objectif essentiel reste inchangé : « Transformer les politiques organisationnelles en garanties démontrables appuyées par des preuves vérifiables ».

6.10 Conclusion du chapitre

Pendant des décennies, les systèmes d’information ont été principalement conçus pour exécuter des fonctionnalités métier.

Les exigences réglementaires croissantes, les cybermenaces de plus en plus sophistiquées et le rythme accéléré de la fourniture de logiciels nécessitent désormais un objectif différent.

Les organisations ne doivent plus seulement exécuter correctement les processus métier.

Elles doivent également être capables de démontrer que ces processus ont été exécutés conformément aux politiques qu'elles ont définies.

Ce livre blanc propose un modèle d’architecture dans lequel la gouvernance, l'architecture, la sécurité et l'exécution font partie d'une seule chaîne continue.

- Les politiques deviennent des garanties
- Les garanties sont vérifiées par des Contrôleurs de Garanties spécialisés
- Les Contrôleurs de Garanties produisent des preuves
- Les preuves guident l'exécution
- L'exécution, à son tour, démontre continuellement la gouvernance

En fin de compte, la confiance n’est plus assumée, elle est construite, vérifiée, enregistrée et démontrée en permanence.

Chapitre 7 – Les limites du modèle

Une propriété non observable ne peut pas être démontrée. Un « Behavior Manifest » peut caractériser des capacités observables, pas démontrer l'absence de toute vulnérabilité.

Une preuve démontre un fait dans un contexte donné. Elle ne garantit pas automatiquement la validité métier du résultat.

L'architecture ne crée pas les garanties que les infrastructures sous-jacentes ne fournissent pas.

La gouvernance démontrable complète la gouvernance organisationnelle ; elle ne la remplace pas. Certaines décisions restent humaines : acceptation du risque, qualification d'une politique, approbation d'un comportement, fonctionnement dégradé, etc.

Annexe A – Vocabulaire des concepts fondamentaux

A.1 Moteur d’exécution

Définition	Composant logiciel chargé d'exécuter un traitement métier dans un environnement contrôlé.
Description	Le moteur d’exécution fournit les ressources nécessaires au traitement (données, secrets, référentiels), applique les contrôles et les actions de gouvernance, assure la transaction, produit les preuves d'exécution et garantit la reprise après incident.
Responsabilités	• charger le traitement métier • fournir les données • fournir les secrets • gérer les transports • appliquer les politiques • produire les preuves

A.2 Orchestrateur de garanties

Définition	Composant chargé d'appliquer les contrôles et les actions nécessaires pour satisfaire les politiques de gouvernance avant, pendant et après l'exécution d'un traitement métier.
Responsabilité	Il orchestre les garanties (pas les traitements)

A.3 Contrôle

Définition	Vérification produisant une décision. Exemple : • intégrité valide • contrat respecté • signature correcte
Responsabilité	Répondre « oui » ou « non »

A.4 Action

Définition	Opération exécutée afin de produire une garantie ou une preuve. Exemple : • signer une donnée • journaliser un état

	• archiver un artefact
Responsabilité	Modifier l'état du système

A.5 Politique

Définition	Ensemble de règles définissant les objectifs de gouvernance de l'organisation. Exemple : Toutes les données personnelles doivent être signées Le moteur d’exécution n'interprète pas l'objectif, il applique les contrôles qui le matérialisent.

A.6 Garantie

Définition	Résultat démontrable obtenu grâce à un ou plusieurs contrôles et actions. Exemple : Garantie d'intégrité des données

A.7 Preuve

Définition	Élément objectif démontrant qu'une garantie a été obtenue. Exemple : • signature • journal • état transactionnel • résultat d'un contrôle

A.8 Confiance

Définition	Niveau d'assurance obtenu par l'accumulation de garanties démontrables. La confiance n'est pas un mécanisme. C'est une conséquence.

A.9 Manifeste

Définition	Description signée d'un artefact, d'un traitement ou d'une clé, utilisée pour vérifier sa conformité et son identité.

A.10 Contrat de données

Définition	Décrit la structure attendue des données.

A.11 Contrat métier

Définition	Décrit les services fournis par une application

A.12 Contrat d'exécution

Définition	Décrit les garanties apportées par le moteur d’exécution.

A.13 Domaine

Définition	Périmètre de responsabilité associé à une politique, des clés ou des artefacts. Exemples : • Enterprise • Platform • Business • Data

A.14 Chaîne de confiance

Définition	Ensemble des relations de certification permettant de démontrer qu'un artefact provient d'une autorité reconnue.

A.15 Gouvernance démontrable

Définition	Approche consistant à transformer des politiques organisationnelles en mécanismes techniques produisant automatiquement les preuves nécessaires pour démontrer leur application.