Replay n'est plus Replay

De l'observabilité des événements à l'architecture épistémique

Ceci est le deuxième article d'une série doctrinale en trois parties sur la gouvernance de l'IA. L'article 1 a établi que les agents sont des choix d'implémentation pour les capacités bornées. Cet article examine ce qui change lorsque ces implémentations sont probabilistes. L'article 3 traite de la manière de gouverner les conséquences des décisions au sein d'un système d'agents probabilistes.

La proposition centrale : les systèmes probabilistes exigent une nouvelle architecture pour les systèmes distribués, où chaque assertion computationnelle porte les conditions épistémiques de sa propre production et est accompagnée de frontières d'autorité explicites.

1. Le problème fondamental : Replay(Événement) n'est pas Replay(Décision)

Dans les systèmes déterministes classiques, le replay semble simple. Un événement entre dans un service déterministe. Le service le traite. La sortie est reproductible.

Mais cette reproductibilité repose sur des hypothèses qui s'avèrent fragiles même dans les systèmes classiques. Une base de données change. Une API distante se met à jour. La configuration se décale. L'ordre des événements dans une queue diffère. Le rejeu fidèle est souvent impossible, même dans les systèmes déterministes.

La différence structurelle dans les systèmes probabilistes n'est pas que le replay devient impossible. C'est que la dimensionnalité de ce qui doit être préservé pour un replay significativement fidèle augmente.

Formellement :

$$D_t = f(E_t, S_t, C_t, M_t, R_t, T_t, \xi_t)$$

où :

- E = Événement
- S = État d'exécution
- C = Contexte
- M = Modèle
- R = Preuves récupérées
- T = Outils invoqués
- ξ_t = Variance d'exécution incontrôlée ou non enregistrée

Le « replay » exact n'est pas intrinsèquement impossible. Mais elle exige de geler suffisamment de l'environnement d'exécution (graines aléatoires, versions runtime, poids du modèle, état du corpus récupéré, réponses des outils, configuration matérielle) pour que la variance résiduelle devienne négligeable. Cette préservation a un coût.

La réalité architecturale : le coût de la reproductibilité augmente avec la dimensionnalité.

Coût_reproductibilité = f(ProfondeurÉtat, ProfondeurEnvironnement, ProfondeurÉpistémique, TolérranceVariance)

La dimension nouvelle est la Profondeur Épistémique. C'est ce que les systèmes probabilistes ajoutent. Vous devez maintenant préserver non seulement l'état d'exécution et l'environnement, mais aussi les conditions épistémiques sous lesquelles l'assertion a été produite.

Par conséquent : Le « replay fidèle » est un choix économique, pas un binaire technique. La question n'est pas « Pouvons-nous faire un replay ? » mais « Quel niveau de reproductibilité la conséquence de cette décision justifie-t-elle ? »

2. Les deux couches d'observabilité : événements et assertions

Dans les systèmes classiques, l'unité d'observabilité est l'événement. Les journaux d'événements construisent l'historique du système.

Dans les systèmes probabilistes, une nouvelle couche émerge : l'assertion — le résultat d'un calcul qui porte l'incertitude épistémique.

Nous devons maintenant distinguer soigneusement :

- Événement : Ce qui s'est passé (entrée, déclencheur, message)
- Exécution : Comment cela a été traité (conditions techniques, chemin computationnel)
- Assertion : Ce qui a été produit (résultat computationnel avec statut épistémique)
- Décision : Quelle action a été choisie (basée sur assertion plus politique plus autorité)

Chacun répond à une question distincte et porte des métadonnées distinctes :

Enveloppe d'événement (Quoi ?) :

```
{ event_id, timestamp, producer, causal_parent, correlation_id }
```

Enveloppe d'exécution (Comment ?) :

```
{ model_version, prompt_version, parameters, tools_invoked,
retrieved_sources, runtime_config, state_snapshot }
```

Enveloppe d'assertion (Quel résultat, avec quel statut épistémique ?) :

```
{ value: "Severity=HIGH", epistemic_status: INFERRED, confidence: 0.78, evidence: [Case_001, Case_045], validity_domain: "outpatient_notifications", expiration: T+90_days }
```

Enveloppe de décision (Quelle action, avec quelle autorité ?) :

```
{ decision: "Escalate", policy_applied: escalation_policy_v2, authority: read_only, decision_basis: "assertion severity HIGH plus policy rule 3.2" }
```

Cette progression reproduit désormais l'évolution classique de l'observabilité : monitoring (Événement) -> observabilité (Assertion) -> auditabilité (Décision avec justification).

L'insight critique : l'assertion est l'unité épistémique. La décision est l'unité opérationnelle. Elles ne doivent pas être confondues.

3. Gouvernance d'état : le contexte comme dossier d'audit

L'exécution probabiliste dépend fondamentalement du contexte. Le comportement au temps T dépend des conditions au temps T moins 1.

Comportement_t = f(Événement_t, Contexte_t-1)

Le contexte devient partie du dossier d'audit. Mais la préservation doit être proportionnelle à la conséquence :

Profondeur_persistance = f(Criticité_décision, Réversibilité, Exigence_audit, Risque_confidentialité, Coût_stockage)

Les assertions à faible risque exigent seulement l'événement plus l'assertion plus la version du modèle. Les décisions à haut risque exigent l'enveloppe d'exécution complète plus les snapshots d'état. Les décisions réglementées exigent des chaînes de provenance immuables.

Le principe architectural : ne demandez pas la préservation exhaustive d'état partout. Achetez le niveau de rétention de contexte justifié par la conséquence de la décision.

4. Typage épistémique : une nouvelle discipline architecturale

Les systèmes de type classiques codent la structure :

Typage classique : « Ceci est une chaîne de caractères »

Le typage sémantique a ajouté du sens :

Typage sémantique : « Ceci est une classification de sévérité »

Le typage épistémique code maintenant comment nous savons :

Typage épistémique : « Ceci est une sévérité inférée par SafetyClassifier_V3.0 à partir des cas [001,045] avec confiance 0.78, valide pour le domaine ambulatoire, expirant à T+90 jours »

Le typage épistémique est une discipline architecturale (pas nécessairement un système de type formel de langage de programmation) qui gouverne :

- Quels statuts épistémiques sont permis
- Quelles transformations sont autorisées entre statuts
- Quelles opérations exigent des portes de politique explicites
- Quand les assertions expirent ou perdent leur validité

La taxonomie du statut épistémique

Les types épistémiques sont multidimensionnels :

Epistemic_type = Origin × Method × TemporalStatus

Origin : Sensor | Human | Model | Policy | ExternalSource

Method : Observed | Derived | Inferred

TemporalStatus : Current | Predicted | Historical

Combinaisons exemples :

- (Model, Inferred, Predicted) -> Sortie du modèle pour les états futurs
- (Human, Observed, Current) -> Observation du clinicien maintenant
- (Policy, Derived, Current) -> Dérivation basée sur règles maintenant
- (Sensor, Observed, Historical) -> Données de capteur enregistrées du passé

Cette structure multidimensionnelle évite de forcer de faux choix entre INFERRED et PREDICTED — ce sont des axes indépendants.

L'algèbre épistémique

Le typage épistémique active les règles formelles de transformation :

(Model, Inferred, Current) + [validation_by: Human] -> (Human, Observed, Current)

(Model, Inferred, Predicted) + [policy_gate: X] -> (Policy, Derived, Predicted)

Expired(assertion) -> InvalidForDecision(assertion)

Critiquement : il n'y a PAS de transformation implicite :

(Model, Inferred, Current) -/-> (Human, Observed, Current) [SANS validation explicite]

Cette règle prévient le blanchiment épistémique — l'effondrement non autorisé des types épistémiques.

Le blanchiment épistémique est précisément :

transformation_non_autorisée(type_assertion) = violation de l'algèbre épistémique

Une architecture avec vérification de type épistémique peut appliquer ces règles aux frontières de contrat, prévenant la perte en cascade de précision épistémique.

5. Autorité : la deuxième dimension orthogonale

Le statut épistémique répond « Comment savons-nous cela ? »

L'autorité répond « À quoi cette information est-elle autorisée à causer ? »

Ces deux dimensions sont indépendantes :

- (Model, Inferred, Current) + authority: advisory -> peut informer les recommandations
- (Model, Inferred, Current) + authority: executable -> peut déclencher des actions (risque plus élevé)
- (Human, Observed, Current) + authority: read_only -> information seulement
- (Human, Observed, Current) + authority: executable -> action autorisée

Le risque est donc multifactoriel :

Risk(assertion) = EpistemicRisk(assertion_type) × AuthorityRisk(authority_level) × IrreversibilityRisk(effect)

Une assertion épistémique avec autorité zéro ne pose aucun risque opérationnel, indépendamment de son incertitude.

La même assertion avec autorité exécutable devient systémique.

6. L'échelle de reproductibilité

La reproductibilité n'est pas binaire. C'est une échelle graduée de garanties :

Idempotence d'effet : une livraison dupliquée crée-t-elle des effets involontaires dupliqués ? C'est un problème classique de systèmes distribués. Bon marché. Généralement requis sauf si l'opération est intentionnellement non-idempotente (par exemple, « enregistrer une nouvelle mesure »).

Stabilité d'assertion : avec le même contexte gelé (même version du modèle, mêmes preuves, même état de politique), la catégorie d'assertion reste-t-elle stable ? L'amplitude peut se décaler (confiance 0.78 à 0.82 du fait de la variance résiduelle), mais la classification (HIGH vs. LOW) reste. Coût modéré. Requis pour les décisions de risque moyen.

Reproductibilité d'exécution : pouvez-vous obtenir exactement la même assertion si vous rejouez tous les (E, S, C, M, R, T) ? Dépend de ξ_t. Cher ou impossible sans mesures extraordinaires. Requis seulement pour les décisions à haut risque ou réglementées.

La gouvernance s'ajuste à la conséquence :

Reproductibilité_requise = f(Impact, Irréversibilité, Exposition_réglementaire, Auditabilité)

Le principe architectural : achetez le niveau de reproductibilité justifié par la conséquence de la décision. L'exactitude coûteuse n'est pas requise partout.

7. Complexité EDA : le produit cartésien des incertitudes

L'architecture événementielle apporte des bénéfices (asynchronie, résilience) et des coûts (incertitude d'ordre, cohérence éventuelle).

Les systèmes probabilistes ajoutent une dimension structurelle :

Ω_système = Ω_distribué × Ω_épistémique

où :

Ω_distribué = espace des ordres d'événements possibles, états de cohérence

Ω_épistémique = espace des états épistémiques (versions de modèle, confiances d'assertion, états de politique)

Le produit cartésien est multiplicativement plus grand. La décision résultante ne peut pas être dérivée du seul ensemble d'événements ; elle dépend à la fois de l'ordre d'événements réalisé et de l'état épistémique au moment de l'exécution.

La solution : enveloppe d'assertion plus snapshots d'état versionnés à chaque agent. Cela permet la reconstruction de causalité temporelle — vous pouvez avancer dans la séquence et comprendre pourquoi chaque agent a produit son assertion.

8. Deux plans : Architecture vs. Runtime

Nous avons maintenant identifié deux structures distinctes :

Plan de gouvernance (Architecture d'autorité)

Capacité -> Droits de décision -> Autorité

C'est l'architecture de qui est autorisé à décider quoi. Elle est statique ou se change lentement.

Plan d'exécution (Séquence runtime)

Événement -> Exécution -> Assertion -> Décision -> Effet

Ceci est ce qui se passe à runtime. Les événements entrent, les assertions émergent (portant le statut épistémique), les politiques transforment les assertions en décisions (vérifiant l'autorité), et les effets sortent.

L'insight critique : ces structures sont orthogonales et ont été précédemment confondues.

La gouvernance répond « Qui décide ? » L'exécution répond « Qu'arrive-t-il ? »

Les assertions habitent les deux :

Dans le plan de gouvernance : « Ce type d'assertion est assigné à ce niveau d'autorité »

Dans le plan d'exécution : « Ce calcul a produit une assertion avec ce statut épistémique »

Cette séparation résout l'ambiguïté architecturale et prépare le cadre de gouvernance de l'article 3.

9. Contrats épistémiques : le point d'intégration

Les contrats épistémiques portent quatre dimensions :

EpistemicContract = Assertion_value + Epistemic_type(origin, method, temporal) + Authority_boundary + Validity_conditions(domain, expiration, reversibility)

Ces contrats se déplacent entre les services. Un système aval voit non seulement une valeur (« Severity=HIGH ») mais son contexte épistémique complet.

Cela transforme les systèmes distribués de « services qui échangent des données » à « services qui échangent des assertions avec provenance traçable et autorité bornée ».

Cela s'applique bien au-delà des agents :

- Machine learning classique (sorties de modèle avec incertitude)
- Jumeaux numériques (simulations avec statut prédit)
- Moteurs de recommandation (scores avec statut inféré)
- Systèmes autonomes (actions avec niveaux d'autorité)
- Systèmes hybrides humain-IA (mélange d'assertions humaines avec inférences machine)

Le motif commun : le calcul probabiliste produit des assertions. Les assertions doivent porter leur statut épistémique et leurs frontières d'autorité. Les contrats doivent rendre cela explicite.

10. L'ontologie centrale

Les cinq objets architecturaux sont :

- Capacité = ce qui doit être faisable (primitive architecturale de l'article 1)
- Exécution = comment le calcul advient (modalité technique)

- Assertion = ce que le calcul produit (résultat épistémique)
- Décision = quelle action est choisie (basée sur assertion + politique + autorité)
- Effet = ce qui se passe dans le monde (conséquence opérationnelle)

Deux propriétés transverses :

- Epistemic_status = Comment nous savons (origine, méthode, temporalité)
- Authority = Ce que nous sommes autorisés à faire (droits d'action bornés)

Cette ontologie est stable à travers les trois articles.

11. Pré-positionnement de l'article 3 : de l'assertion à l'autorité

La progression de cette série est :

Article 1 (Ontologie) : L'agent n'est pas la capacité. Les agents sont des choix d'implémentation pour les capacités. La distinction entre primitive architecturale et modalité d'implémentation est critique.

Article 2 (Épistémologie) : L'assertion n'est pas le fait. Les assertions doivent porter le statut épistémique. Effondrer les assertions en faits sans justification épistémique est du blanchiment épistémique.

Article 3 (Gouvernance) : L'inférence n'est pas l'autorité. Produire une assertion ne confère pas le droit de produire un effet. La gouvernance doit opérer à quatre niveaux : Authority(Capability), Accountability(Decision), Observability(Execution), Liability(Effect).

Le motif plus profond à travers les trois articles : ne jamais laisser une propriété d'implémentation usurper une propriété architecturale.

Conclusion

Les systèmes distribués avec composants probabilistes font face à une exigence architecturale nouvelle : le typage épistémique — la discipline d'assurer que chaque assertion porte ses conditions de crédibilité et ses frontières d'autorité.

Ce n'est pas spécifique aux agents. Cela s'applique partout où le calcul probabiliste produit des valeurs pour la consommation d'autres services.

La série en trois articles propose une architecture cohérente :

Capacité (ce qui doit exister)

-> Droits de décision (qui décide)

-> Autorité (droits d'action bornés)

-> Exécution (réalisation technique)

-> Assertion (résultat épistémique)

-> Décision (sélection d'action)

-> Effet (changement dans le monde)

-> Gouvernance (gestion de conséquence)

Où les assertions ne s'effondrent jamais implicitement en faits, et l'autorité n'est jamais supposée à partir de la seule capacité technique.

C'est ce que la gouvernance des agents probabilistes exige.