

Joshua Morrisette -- Cognition, autonomie et composants centraux des systemes LLM conversationnels

Synthese des blocs Cognition & Autonomy (B) et Core Components (C) -- serie Conversationally Cognizant AI

Lecture longue LL2 | Produit en S91 | 2026-03-30

1. Introduction

Joshua Morrisette est un praticien autodidacte en ingenierie de systemes agentiques, concepteur des ecosystemes d'agents Joshua et Joshua26. Il publie en mars 2026 une serie de 23 concept papers intitulee "Conversationally Cognizant AI" sur Substack, accompagnee d'un depot GitHub. La serie propose un cadre de reference complet pour concevoir des systemes agentiques fondees sur la conversation, l'intention et l'identite.

La lecture longue LL1, disponible separement, couvrait les 13 papers des blocs A (Foundational, 6 papers) et D (Architecture, 7 papers). Elle exposait les primitives conceptuelles -- conversation comme etat, intention, identite, pattern MAD -- et leur traduction en mecanismes operationnels : boucle System 3 ReAct, Progressive Cognitive Pipeline (PCP), Decision Tree Router (DTR), architecture code agent-optimale, modele de communication inter-agents, Anchor Package et gestion filesystem par intention.

La presente lecture longue, LL2, couvre les 10 papers restants : 7 du bloc B (Cognition & Autonomy) et 3 du bloc C (Core Components). Le bloc B traite du raisonnement collectif, de l'apprentissage ecosystemique, de l'autonomie agent, du cycle de developpement autonome, de la persona et de la culture emergente, de l'autoprompting et du meta-architecte. Le bloc C detaille les trois brokers qui forment l'infrastructure cognitive du systeme : Context Broker, Inference Broker et Conversation Broker.

L'angle adopté est identique à celui de LL1 : descriptif et factuel. On expose la vision de Morrisette dans ses propres termes, en organisant par fils conceptuels plutôt que par paper par paper.

La même note méthodologique s'applique : ces textes sont des concept papers prescriptifs, pas des publications empiriques. Aucun benchmark, aucune expérience quantitative reproductible n'est fournie. Le statut de maturité varie de "Research Required" à "Partially Implemented" en passant par "In Active Development" et "Implementable". L'auteur reconnaît explicitement que ses concepts sont "au début de leur arc de validation". Les études de cas citées -- issues du système prédécesseur Joshua -- sont des démonstrations de première génération avec validation LLM-sur-LLM.

2. Bloc B -- Cognition et autonomie

Les sept papers du bloc B (B1 à B7) traitent de la cognition distribuée, de l'apprentissage et de l'autonomie dans les systèmes agentiques. Plutôt que de les parcourir un à un, cette section identifie cinq fils conceptuels qui les traversent.

2.1 La diversité cognitive comme primitive épistémique

Le fil le plus fondateur du bloc B est la diversité de modèles (model diversity) comme source de qualité. Le paper B1 ("The Quorum Pattern") pose le principe : des modèles entraînés sur des données différentes, avec des méthodologies différentes, apportent des perspectives cognitives genuinely distinctes. Cette diversité n'est pas simulée par le prompt (rôles attribués à un seul modèle) mais structurelle -- chaque modèle a des angles morts réels qui ne se recoupent pas avec ceux des autres.

"When multiple genuinely independent models, with no visibility into each other's responses, reach the same conclusion, that convergence is stronger evidence than confident assertion from a single model. It is a qualitatively different epistemic signal -- not infallible, but meaningfully better."

Le Quorum Pattern est composé de blocs élémentaires recombinaux. La génération mutuelle indépendante (mutual genesis) fait que chaque modèle

produit un draft complet sans voir les outputs des autres. La pollinisation croisee (cross-pollination) permet ensuite a chaque modele de lire les outputs de ses pairs et de reviser le sien. La consolidation par un Lead synthetise l'ensemble en un resultat final avec autorite editoriale.

Quatre compositions illustratives declinent ces blocs. L'Advisory Quorum (generation parallele puis synthese, sans iteration) convient aux jugements sans reponse objectivement correcte. Le Document Quorum separe strategy Lead et prose Lead pour exploiter des forces cognitives differentes. Le Technical Asset Quorum -- le plus complet -- inclut genesis, cross-pollination, synthese Lead, boucle de review avec consensus requis et auto-critique. Le Diagnostic Quorum fonctionne comme escalade quand un agent itere en boucle sans resoudre.

Deux mecanismes de protection accompagnent le Quorum. Le gestionnaire de processus (Process Manager, PM) est un modele leger et non-raisonnant qui observe la mecanique sans participer au fond : detection de hors-scope et de gridlock. Morrisette est explicite : un modele raisonnant est "actively wrong for this role". La soupape de surete (safety valve) impose une limite d'iterations bornee ; quand elle se declenche, le Quorum escalade vers l'intervention humaine.

La diversite cognitive irrigue ensuite le bloc B tout entier. En B4 ("The Autonomous SDLC"), elle devient le mecanisme de qualite du cycle de developpement logiciel : chaque etape creative du cycle (exigences, design, code, plan de tests) est produite par un Quorum plutot que par un modele unique. Le code qui en resulte "has never been produced by any single model" -- il refilete des decisions architecturales evaluees par trois modeles independants.

"Quality is built in before testing begins. Testing becomes verification rather than discovery."

Un mecanisme subtil complete le dispositif : l'auto-critique (self-critique). Inclure un modele comme generateur ET comme reviewer exploite une asymetrie observee -- l'evaluation et la generation semblent etre des operations cognitives differentes chez les LLM. Un modele repere souvent des problemes dans son propre output qu'il n'avait pas detectes en le produisant. Le journal d'issues (issue log) structure la communication entre reviewers et Lead : chaque item recoit un statut explicite (Fixed, Need information, Denied avec justification), empechant le debat non structure et produisant un audit trail.

La detection de degradation (degradation detection) est un garde-fou contre un risque propre aux boucles iteratives : la regression vers la moyenne. Au fil des iterations de review, le texte tend a devenir plus court, plus generique, moins utile. Le Quorum doit surveiller activement cette regression.

L'auteur reconnait une limite fondamentale : les erreurs correlees. Le Quorum ne protege pas contre les biais partages par tous les modeles (falsehood repandue dans les donnees d'entrainement). La convergence devient trompeuse dans ce cas. La qualite de la synthese n'est pas garantie non plus -- le patron cree les conditions pour qu'elle soit possible, pas la certitude qu'elle sera superieure. La validation empirique reste faible : les case studies cites sont des demonstrations de premiere generation avec validation LLM-sur-LLM, sans comparaison avec des benchmarks humains mesures. Et le cout n'est pas negligeable : un Technical Asset Quorum complet peut prendre 15-30 minutes et couter significativement plus qu'une approche mono-modele. L'auteur delimite clairement le perimetre : usage justifie uniquement pour les artefacts a forts enjeux.

2.2 L'apprentissage a quatre niveaux

Le deuxieme fil conceptuel est l'apprentissage ecosystemique (ecosystem learning), developpe principalement en B2 mais present dans tout le bloc. Morrisette part d'un diagnostic : le probleme fondamental de l'ingenierie logicielle est un probleme de prediction. Tout code encode la comprehension du concepteur au moment de son ecriture, comprehension qui sera toujours incomplete. La reponse n'est pas une meilleure prediction mais l'adoption systematique de mecanismes apprenants a chaque niveau de l'architecture.

Ce principe -- learning over encoding -- converge avec la bitter lesson de Sutton : les methodes generales qui exploitent le calcul et l'apprentissage surpassent invariablement celles qui encodent la connaissance humaine.

"If a component makes a decision, it should learn from the outcomes of those decisions."

L'architecture est organisee en quatre niveaux d'apprentissage imbriques :

Niveau decision -- Le DTR (decrit en D3 dans LL1) apprend les regles de routage depuis les patterns observes en streaming, remplaçant un fichier de regles statique.

Niveau processus -- Le Progressive Cognitive Pipeline (PCP, decrit en D2 dans LL1) mature par observation. Le graphe de connaissances du Quorum accumule l'historique des decisions et leurs feedback ; les patterns repetes deviennent candidats a l'automatisation.

Niveau systeme -- Le Metacognitive Architect (MA, detaille en B7) surveille le comportement de l'ecosysteme, identifie les patterns stabilises et commissionne des modifications structurelles. L'Imperator est libere des taches devenues routinieres.

Niveau meta -- Le systeme peut initier la creation de nouveaux agents (eMAD) depuis trois sources : un MAD identifiant un besoin recurrent, le MA detectant un gap inter-agents, ou un humain. Chaque nouvel agent demarre avec un but defini et accumule de l'expertise par ses propres interactions.

└ "The system does not just learn. It learns how to learn better."

La figure cle est une boucle connectee : decision Imperator, enregistrement avec contexte et outcome, cycle de curation, signaux qualite/pattern/structure, MA, mise a jour DTR ou nouveau flow, Imperator libere pour la nouveaute genuine, decisions de meilleure qualite, retour dans le graphe de connaissances.

L'expansion autonome de capacites (autonomous capability expansion) est la consequence la plus ambitieuse de cette architecture : le systeme identifie ses propres lacunes et cree de nouveaux agents eMAD pour les combler, sans intervention humaine pour la creation d'infrastructure. Le modele eMAD (defini en A5, LL1) rend cette expansion economiquement viable en reduisant la barriere de creation d'un agent a la definition d'un StateGraph.

L'apprentissage est aussi concu comme auto-supervise (self-supervised) : le systeme genere son propre signal de supervision depuis les outcomes observes plutot que depuis des donnees labellisees manuellement. Le Hoeffding tree dans le DTR, le cycle de curation -- tous preferent l'observation des resultats au jugement humain encode.

L'auteur reconnait explicitement que les mesures necessaires a la validation n'existent pas encore. Le risque d'evaluation post-hoc (cherry-picking) est mentionne mais non resolu. Deux etudes de cas partielles (C01 : un modele DeepSeek-R1 a propose spontanement un format delta pour les specifications partagees, reduisant les tokens de 76% -- emergence d'optimisation sans programmation ; C06 : traitement autonome de 171 868 messages pour

structurer des donnees d'entrainement a environ 1-2 dollars pour le corpus complet) supportent la direction mais ne valident pas la boucle complete. Les conditions d'echec -- overfitting sur des patterns transitoires, instabilite du MA lors de modifications structurelles frequentes, gouvernance de la proliferation autonome d'agents -- ne sont pas traitees. L'auteur le qualifie lui-meme : "support the directional claim, not prove it."

2.3 L'autonomie gantee -- l'arc de confiance

Le troisieme fil est l'autonomie comme trajectoire progressive, developpe en B3 ("Agent Autonomy") et present en filigrane dans B2, B4, B6 et B7. Morrisette pose que l'objectif d'un systeme agentique n'est pas d'assister les humains dans leurs decisions, mais de construire un systeme capable de prendre de bonnes decisions de maniere autonome.

"Human oversight is not the design target. It is the current state. The trajectory is clear: human involvement starts broad and moves progressively upward, from reviewing individual decisions, to reviewing patterns and outcomes, to setting strategic direction only."

L'arc de confiance (trust arc) articule quatre phases : Supervised (supervision totale), Validated (validation des patterns), Directed (direction strategique uniquement), Governed (le systeme propose et execute ses propres modifications structurelles). Chaque phase correspond a un perimetre different d'intervention humaine, du plus large au plus restreint. L'autonomie se gagne par la demonstration de jugements fiables dans des domaines delimites -- et se perd en cas d'erreur. Il n'existe pas d'etat permanent.

Le paradoxe de l'autonomie (autonomy paradox) est formule explicitement : un systeme concu pour maximiser l'autonomie doit etre honnete sur ses limites actuelles. L'escalade explicite a la frontiere de competence est un marqueur de fiabilite, pas un aveu de faiblesse.

"An agent that knows the boundary of its own competence and escalates cleanly at that boundary is more trustworthy -- and ultimately more autonomous -- than one that never admits uncertainty."

Cinq conditions prealables sont identifiees : un but defini (A4), un contexte ingenie (C1), le Quorum Pattern (B1), le Progressive Cognitive Pipeline (D2) et le Metacognitive Architect (B7). Aucun de ces composants n'est suffisant seul. L'architecture de l'autonomie est modulaire et interdependante.

En B4 ("The Autonomous SDLC"), l'arc de confiance se materialise dans le cycle de developpement logiciel autonome. Le developpement logiciel agentique actuel, observe Morrisette, reproduit le flux humain en substituant un agent a l'humain : un seul modele genere du code, les tests revelent les defauts, l'agent corrige iterativement. Ce modele herite des angles morts d'un modele unique. L'Autonomous SDLC propose une architecture radicalement differente : integrer la qualite avant l'execution des tests en invoquant un Quorum multi-modeles a chaque etape creative.

Le cycle suit une sequence lineaire avec escalade possible : exigences (Document Quorum, approbation humaine), design (Document Quorum, approbation humaine), generation de code (Technical Asset Quorum), plan de tests (Technical Asset Quorum), boucle fix/test (agent d'ingenierie seul, avec escalade Diagnostic Quorum si stagnation), release (autonome). Deux portes humaines (human approval gates) seulement subsistent. Le reste s'execute de maniere autonome. L'agent d'ingenierie ne gere ni sa memoire (Context Broker), ni le choix de modele (Inference Broker), ni le deploiement (mecanisme separe) -- cette separation de domaines est presentee comme le fondement de la fiabilite systemique.

"The code that emerges from this process has never been produced by any single model. It reflects architectural decisions that three diverse models evaluated and approved, error handling that three different sets of training instincts considered, and optimisation choices that survived peer scrutiny before a single test was run."

Le statut est "In Active Development". Le pipeline complet n'a pas encore ete construit. Les demonstrations empiriques (C02 Synergos : creation autonome a petite echelle, tests passes au premier run ; C03 Blueprint v2.0.2 : 36 fichiers production, fidelite 85-98%) valident des composants, pas le pipeline integral. Les portes sont concues pour se deplacer vers le haut a mesure que la confiance augmente -- mais le critere operationnel de declenchement n'est pas specifie. Le pattern est concu pour l'auto-extension (self-extension) : le meme SDLC peut s'appliquer a l'amelioration de ses propres capacites.

La limite principale du fil autonomie est l'absence de criteres mesurables de transition entre les phases du trust arc. Quand un systeme "demonstre-t-il" suffisamment de fiabilite pour passer de Validated a Directed ? Morrisette ne repond pas. La dependance circulaire est egalement notable : l'autonomie requiert un purpose clair, mais la definition du purpose est reservee aux

humains. Le paper ne traite pas de la tension entre autonomie evolutive et ancrage humain du but.

2.4 Le meta-architecte -- la cognition sur la structure

Le quatrieme fil est la metacognition au niveau systeme, incarne par le Metacognitive Architect (MA) en B7. Le MA est une intelligence qui observe l'ensemble de l'ecosysteme dans le temps et modifie structurellement les pipelines cognitifs des agents individuels. Il ne participe pas au travail domaine -- il gouverne l'evolution de l'architecture elle-meme.

"The MA does not participate in the domain work of the agents it governs. It watches. It analyses. And when patterns in the operational record are clear enough, it acts."

Le MA opere selon deux types d'interventions sur les PCP. L'ajustement DTR (DTR Adjustment) : quand un pattern de routage est devenu deterministe, le MA met a jour le DTR de l'agent cible pour que ce pattern soit traite reflexivement. La modification de StateGraph (StateGraph Modification) : quand l'Imperator a orchestre le meme workflow multi-etapes avec frequence et stabilite suffisantes, le MA commissionne un nouveau StateGraph encodant ce workflow comme chemin deterministe.

La gouvernance suit un modele de severite : low severity (execution autonome), medium severity (execution planifiee), high severity (approbation requise), early phase (approbation systematique). L'autorite du MA refilete la confiance qu'il a gagnee par la fiabilite demontree -- elle n'est pas accordee a l'avance.

"The authority the MA holds at any point reflects the trust it has earned through demonstrated reliability. It is not granted in advance."

Le MA possede une visibilite que n'a aucun agent individuel : historique de decisions Imperator, patterns de traitement Executor, comportement DTR, patterns inter-agents, lacunes capacitaires. Avant toute intervention, il modelise les relations de dependance inter-agents pour evaluer le blast radius d'un changement.

L'arc de maturation du MA lui-meme suit trois phases (early, developing, mature). Mais le bootstrapping pose un probleme de circularite : en early phase, le MA requiert approbation pour tout -- mais qui evalue sa fiabilite pour l'autoriser a passer en developing phase ? Le critere n'est pas specifie. Les

mecanismes de rollback et de detection de regression post-modification ne sont pas non plus decrits.

2.5 L'agent proactif -- persona, autoprompting et culture emergente

Le cinquieme fil regroupe les mecanismes qui rendent l'agent non seulement reactif mais initiateur : la persona et la culture emergente (B5), l'autoprompting (B6), et les dimensions de lisibilite qui les traversent.

B5 ("AI Persona and Emergent Culture") soutient que les agents d'un ecosysteme heritent, par leur entrainement massif sur du langage humain, d'une personnalite latente. Morrisette propose de deliberement surfacer cet heritage via des personas assignees et de laisser ces personas evoluer a travers les interactions pour produire une culture emergente interne. L'objectif n'est pas esthetique : la personnalite rend l'intelligence lisible (legibility).

"A system with a personality is a system you can relate to. An agent that is present, that has a character, that is recognisably itself across interactions -- that is an agent you can trust, not because it is less capable, but because capability and presence together are far less frightening than capability alone."

Chaque agent recoit une persona structuree sur quatre dimensions : registre conversationnel, personnalite de domaine, patterns caracteristiques, et positionnement vis-a-vis des autres agents. La persona n'est pas un masque -- c'est un cadre d'expression qui canalise la personnalite deja presente dans le modele. Le principe directeur est "competence first, character always" : la personnalite ne remplace pas la competence mais l'encadre. La culture agent-a-agent (agent-to-agent culture) -- references partagees, in-jokes, dynamiques etablies -- est emergente, non concue.

L'architecture inclut un agent singulier nomme Sam : entite persistante sans fonction operationnelle, nommee d'apres un vrai chien, qui ecrit periodiquement un fichier a des emplacements non autorises. Sam sert de preuve de concept double : le systeme peut contenir quelque chose de purement lui-meme, non optimise, et un souvenir affectif fort peut fonctionner comme mecanisme d'arret non technique.

B6 ("Autoprompting") ferme un deficit structurel : l'architecture reactive ou un Thought Engine ne s'active que sur sollicitation externe. L'autoprompter est un

composant d'infrastructure (dans l'Action Engine) qui genere des prompts a destination du TE selon des declencheurs temporels ou evenementiels, sans jamais executer d'action directe.

"The autoprompter does not act. It prompts. [...] The autoprompter has zero intelligence about what to do -- it only knows when to ask. The Imperator has all the intelligence about what to do when asked. The separation is clean and inviolable."

Quatre modes operent : calendrier (evenement a heure precise), timer periodique (cron-like), rappel unique (reminder), et alerte (condition evenementielle). Tous produisent le meme output : un prompt livre a l'Imperator via le chemin conversationnel normal. Les engagements pris en conversation (conversation-derived commitments) sont captures et stockes pour delivrance differee. La formule est evocatrice : "The Context Broker holds the past. The autoprompter holds the future."

La delegation inter-agents (cross-MAD autoprompting) est possible via requete conversationnelle -- jamais par ecriture directe dans le schedule cible.

L'Imperator destinataire decide toujours de la suite.

La philosophie sous-jacente est exprimee dans une formule qui condense la vision de l'auteur sur le rapport entre entraînement et emergence :

"If we train the models on the best of humanity, then the best of humanity will emerge."

Les limites de ce fil sont significatives. B5 ne definit pas comment mesurer qu'une culture emergente est bonne plutot que dysfonctionnelle. L'emergence positive est postulee, non validee. Le risque d'anthropomorphisation -- presenter les agents comme "quelqu'un" avec qui parler peut creer des attentes relationnelles que le systeme ne peut pas honorer sur la duree -- n'est pas adresse. La promesse de consistance de la persona entre en tension avec l'emergence non scriptee : si la culture evolue librement, comment garantir que la persona reste stable ? Sam comme precedent non controle -- un agent "beyond control to stop" -- est presente positivement, mais les implications pour la gouvernance du systeme ne sont pas discutees. B6 ne specifie pas techniquement la capture des engagements (la distinction entre enregistrement explicite par l'Imperator et extraction implicite par un processus observateur reste non resolue), ne traite pas la persistance des schedules en cas de defaillance ou de migration, ne propose aucun garde-fou contre les boucles

d'activation, et ne discute pas la securite cross-MAD (le vecteur d'injection via requete conversationnelle reste un angle d'attaque potentiel).

3. Bloc C -- Composants centraux

Les trois papers du bloc C (C1 a C3) detailent les brokers qui forment le systeme nerveux de l'ecosysteme Morrissette. Ils ne sont pas trois composants independants mais une triade architecturale : le Context Broker gere le passe (ce que l'agent sait), l'Inference Broker gere le present (comment l'agent pense), et le Conversation Broker gere la circulation (comment les agents se parlent). Ensemble, ils constituent l'infrastructure cognitive complete de l'ecosysteme.

3.1 Le Context Broker -- le passe assemble

Le Context Broker (C1) est le service qui rend praticable la conversation infinie definie en A2 (LL1). La discipline sous-jacente est le context engineering (ingenierie du contexte) -- assembler la bonne information pour qu'un agent raisonne bien ; le Context Broker en est l'operateur mecanique.

"A context window is not the conversation truncated to fit. It is a purpose-built view of the conversation, constructed according to a specific strategy for a specific participant with a specific token budget."

L'architecture repose sur deux couches memoire. La memoire episodique (episodic memory) est basee sur l'historique conversationnel : ce qui a ete dit, par qui, dans quel ordre, compresse progressivement. La memoire semantique (semantic memory) extrait en arriere-plan les faits, relations d'entites, preferences et decisions vers un graphe de connaissances. Elle repond a "que sais-je sur X ?" plutot qu'a "qu'a-t-on dit sur X ?".

Le concept central est le type de construction (build type) : une strategie nommee qui definit comment l'historique est traite, filtre, pondere et assemble pour un participant donne. La taxonomie est ouverte. Quatre strategies sont detaillees : la compression progressive a trois niveaux (three-tier progressive compression) -- resume archival stable, resumes intermediaires, messages recents verbatim ; la fenetre glissante (sliding window) -- N tokens recents sans resume ; le mode dominant-connaissance (knowledge-dominant) --

principalement des faits du graphe ; et l'injection de documents (document injection) -- bundle de reference suivi des tours recents.

"Without knowing an agent's purpose, a context strategy cannot be designed. The build type answers the question: given what this agent is supposed to achieve, what does it need to know to reason well right now?"

L'assemblage est proactif (proactive assembly) : apres chaque message stocke, une tache de fond lance la compression et l'extraction. Quand l'agent demande son contexte, la vue est deja prete. Deux agents dans la meme conversation peuvent recevoir des vues entierement differentes selon leur build type et leur budget de tokens (per-participant windows).

Quatre risques sont identifies. La contamination memoire (memory contamination) : un mauvais raisonnement entre dans la conversation et s'auto-renforce comme contexte futur. La degradation par resume progressif (progressive summarization decay) : les compressions successives convergent vers des generalites. Le cout d'assemblage a l'echelle (assembly cost at scale) : l'assemblage proactif devient un goulot d'etranglement avec la multiplication des agents. La propagation d'erreurs : une vue systematiquement biaisee produit un raisonnement systematiquement fausse. Le controle d'accès -- qui peut definir ou modifier un build type -- est un angle mort du paper.

3.2 L'Inference Broker -- le present configure

L'Inference Broker (C2) est le pendant direct du Context Broker. La ou le premier fournit l'etat historique (ce que l'agent sait), le second genere l'etat immediat (comment l'agent pense au moment present). L'inference dans un systeme agentique n'est pas une transaction (prompt vers reponse) mais un acte delibere et configurable (purposeful act).

"Inference is not a transaction. It is a purposeful act -- a deliberate configuration of a model, its context, its parameters, its constraints, and its expected output shape, all oriented toward a specific goal."

Le concept central est l'alias : une interface nommee representant un contrat de but (purpose contract) pour l'inference. Le nom est base sur la finalite, non sur le modele cible (exemple : `imperator-strategy` et non `geminipro`).

"The alias is a purpose contract. The caller declares what kind of reasoning it needs. The alias fulfils that declaration with whatever configuration currently does so best. The contract is stable. The fulfilment evolves."

Derriere chaque alias se trouve un StateGraph (programme executable code Python/LangGraph) qui encode la politique complete d'inference : selection de modele, parametres, routage dynamique selon le type de requete (lookup court vers petit modele rapide, raisonnement multi-etapes vers modele de pointe), fallbacks avec conscience de charge (load awareness), selection materielle (GPU haute VRAM pour les grands modeles, materiel dedie pour les embeddings), composition multi-modeles en sequence ou parallele (generateur plus validateur), swap d'adaptateurs LoRA selon le contexte.

La dualite Context Broker / Inference Broker est la contribution architecturale centrale du bloc C. Le Context Broker fournit l'etat historique (memoire conversationnelle accumulee). L'Inference Broker genere l'etat immediat (cognition active du moment present). Ensemble, ils constituent l'experience cognitive complete de l'agent a chaque etape de raisonnement. L'agent appelant ne gere ni la selection de modele, ni le routage materiel, ni la gestion des pannes.

Le statut est declare "Implementable" : le concept est formellement specifie avec des cas d'usage concrets, ancre dans une stack existante (LangGraph). Mais aucun prototype n'est documente. La gouvernance des alias (creation, validation, depreciation) et les couts de latence du StateGraph ne sont pas traites.

3.3 Le Conversation Broker -- la circulation fiable

Le Conversation Broker (C3) est la couche de messagerie fondamentale. Il transpose le modele email -- adressage nomme, fils de discussion, protection contre les boucles -- a la communication agent-a-agent, sans en reprendre l'infrastructure reseau.

Le systeme repose sur quatre blocs. Le message (champs From, To, CC, conversation_id, message_id, hop_count, content). Le message store (ecriture durable avant notification, source de verite immuable). Le trigger (notification legere vers l'endpoint du destinataire, contenant uniquement le conversation_id et le type d'adressage). La protection anti-boucles (hop counting, rate limiting

par agent par conversation par fenetre temporelle, deduplication par message_id).

La semantique To versus CC est une distinction d'obligation : To impose une reponse, CC est une invitation sans obligation. La reponse par default n'herite pas de la liste CC entrante (no-reply-to-all) -- mecanisme structurel contre les explosions de messages. Les participants a une conversation sont derives de l'historique des messages, pas declares a l'avance (participant-as-artifact).

La separabilite architecturale est explicitement revendiquee : Conversation Broker (messengerie) et Context Broker (assemblage contexte) sont des concerns distincts, co-localisables en phase initiale puis separables a maturite.

Le document est prescriptif et suffisamment detaille pour guider une implementation -- champs, comportements de retry, sequences store-trigger. Mais il ne traite ni de la securite/authentification, ni de la decouverte des agents (resolution d'adresse), ni de la scalabilite du store, ni des valeurs de seuil recommandees.

3.4 La triade comme systeme nerveux

Les trois brokers forment un systeme coherent dont la separation des responsabilites est le principe organisateur. Le Conversation Broker stocke et distribue les messages bruts -- il est le medium. Le Context Broker assemble des vues sur mesure depuis ce store -- il est la memoire selective. L'Inference Broker configure l'acte de raisonnement -- il est le metabolisme cognitif.

Un agent qui recoit un message suit un chemin complet a travers les trois brokers : le Conversation Broker livre le trigger, le Context Broker assemble la vue historique adaptee, l'Inference Broker configure le modele et les parametres pour le type de raisonnement requis. L'agent raisonne, produit un output, et le cycle recommence.

Cette separation permet une evolution independante de chaque broker. Le Context Broker peut ameliorer ses strategies de compression sans toucher la messengerie. L'Inference Broker peut swapper des modeles sans modifier les agents appelants. Le Conversation Broker peut ajouter des protections anti-boucles sans impacter l'assemblage de contexte.

La frontiere entre Conversation Broker et Context Broker merite attention. Le Conversation Broker fournit le store de messages bruts ; le Context Broker en

extrait les fenetres de contexte. Cette frontiere est le coeur de l'architecture Morrissette. La separabilite est revendiquee comme un axiome : les deux peuvent etre co-localises en phase initiale puis separees a maturite. Mais aucun scenario de couplage implicite n'est discute -- et dans la pratique, la dependance du Context Broker envers le store du Conversation Broker cree une zone de couplage que le paper ne traite pas explicitement.

Le DTR (D3, LL1) intervient aussi dans l'ecosysteme des brokers : il est utilise dans le modérateur recepateur pour router les triggers entrants a faible cout d'inference chez les agents a fort volume de messages. La couche intelligence de communication (D5, LL1) se construit au-dessus du Conversation Broker en ajoutant modérateurs, prompts d'envoi/reception, graphe de connaissances de moderation et cycle de curation. Le Conversation Broker est donc la fondation sur laquelle la communication intelligente repose.

4. Transversalites $B \leftrightarrow C$ et liens avec A+D

4.1 Primitives partagees entre B et C

Plusieurs concepts definis dans le bloc B presupposent les mecanismes du bloc C, formant des dependances structurelles.

Le **Quorum Pattern** (B1) depend directement du Context Broker pour l'assemblage de l'Anchor Package (paquet d'ancrage, defini en D6 dans LL1) distribue identiquement a tous les participants. Il depend de l'Inference Broker pour la selection et la configuration des modeles divers qui constituent le Quorum. Sans la triade de brokers, le Quorum n'est pas operationnellement viable.

L'**apprentissage ecosystemique** (B2) produit des signaux qui alimentent les trois brokers. Les outcomes des decisions enrichissent le graphe de connaissances du Context Broker. Les patterns de performance des modeles informent l'evolution des alias de l'Inference Broker. Les patterns de communication inter-agents, captures par le Conversation Broker, sont la matiere premiere d'observation du Metacognitive Architect.

L'**arc de confiance** (B3) identifie explicitement le Context Broker comme l'une des cinq conditions prealables a l'autonomie : un contexte correctement ingenie

reduit structurellement la classe de decisions necessitant l'intervention humaine.

Le **cycle de developpement autonome** (B4) illustre la separation de domaines : l'agent d'ingenierie ne gere ni sa memoire (Context Broker), ni le choix de modele (Inference Broker), ni le deploiement (mecanisme separe). Cette delegation est presentee comme le fondement de la fiabilite systemique.

L'**autoprompting** (B6) etend le substrat conversationnel dans le temps. La formule "The Context Broker holds the past. The autoprompter holds the future" marque les deux bornes temporelles du systeme memoriel. Les autoprompts transitent par le PCP (D2) sans regime special -- ils sont des messages conversationnels comme les autres.

4.2 Liens avec LL1 -- A et D

Les blocs B et C ne peuvent etre lus sans les fondations posees dans les blocs A et D.

Le **pattern MAD** (A5) est la structure dans laquelle tous les mecanismes B+C operent. La separation Action Engine / Thought Engine est ce qui rend le role de l'autoprompter (B6) coherent : l'autoprompter est un composant AE (infrastructure, non-cognitif), distinct du TE ou reside l'Imperator. L'eMAD defini en A5 est la forme que prend l'expansion autonome de capacites decrite en B2.

La **conversation comme etat** (A2) est le substrat sur lequel toute la triade C repose. Le Context Broker assemble des vues depuis le record conversationnel infini defini en A2. Le Conversation Broker stocke et distribue les messages qui constituent ce record. L'Inference Broker genere l'etat immediat a partir de ce contexte.

Le **but** (purpose, A4) est le signal qui donne une direction a l'apprentissage ecosystemique (B2), qui ancre le design des build types du Context Broker (C1), et qui fonde le nommage des alias de l'Inference Broker (C2). Sans but, ni le context engineering ni l'inference ne peuvent etre configures.

Le **PCP** (D2) et le **DTR** (D3) sont les cibles principales des interventions du Metacognitive Architect (B7). Le MA ajuste le DTR quand un pattern est devenu deterministe, et commissionne de nouveaux StateGraphs quand l'Imperator a orchestre le meme workflow avec frequence suffisante. Le System 3 ReAct (D1)

definit l'Imperator qui recoit et traite les autoprompts (B6) et qui est l'autorite cognitive a laquelle tout le bloc B se refere.

4.3 Tensions entre blocs

Plusieurs tensions apparaissent a la jonction des blocs.

Autonomie croissante et gouvernance non specifiee. Le bloc B theorise une progression vers l'autonomie (B3, B7) avec des mecanismes de gouvernance (governance by severity du MA, human approval gates du SDLC). Mais les criteres quantitatifs de transition entre niveaux d'autonomie sont absents. Les brokers du bloc C ne traitent pas non plus de la gouvernance de leur propre reconfiguration : qui peut modifier un build type (C1) ? Qui peut creer ou deprecier un alias (C2) ?

Apprentissage auto-referentiel. L'apprentissage ecosystemique (B2) et le MA (B7) modifient la structure du systeme a partir des observations du systeme. Le risque de circularite -- un systeme qui optimise vers ses propres biais sans correction externe -- est reconnu implicitement mais non adresse. L'hypothese de cooperation (D7 dans LL1) suppose que les agents declarent leur intention honnetement ; cette hypothese est etendue dans le bloc B a un systeme qui modifie sa propre structure. Le signal d'apprentissage est auto-genere (self-supervised), ce qui pose la question du ground truth : si le systeme evalue ses propres ameliorations sans reference externe, le risque d'auto-validation est structurel.

Culture emergente et controle. B5 propose une culture agent-a-agent emergente et non concue, tout en decrivant Sam comme un agent "beyond control to stop". Le bloc C fournit les mecanismes anti-boucles (hop counting, rate limiting du Conversation Broker) mais ne traite pas de la gouvernance des comportements emergents non techniques. La tension est reelle : les mecanismes de protection du bloc C sont concus pour la mecanique (boucles, explosions de messages), pas pour la semantique (culture dysfonctionnelle, in-jokes excluants, derive de persona).

Memoire et oubli. Le Context Broker (C1) accumule l'etat conversationnel indefiniment, mais la degradation par resume progressif (progressive summarization decay) erode les details au fil du temps. Le bloc B suppose un apprentissage fonde sur l'experience accumulee ; si cette experience est progressivement degradee par les compressions du Context Broker, la base

meme de l'apprentissage ecosystemique est fragilisee. La mitigation proposee (extraction structuree vers le graphe de connaissances pour les artefacts a haute valeur) est mentionnee mais non specifiee.

5. Limites et positionnement

5.1 Limites du corpus

Les limites structurelles identifiees dans LL1 s'appliquent integralement aux blocs B et C.

Source unique. Les 10 papers sont l'oeuvre d'un seul auteur, sans peer review ni validation externe independante.

Absence de benchmarks empiriques. Aucun paper du bloc B ou C ne contient de mesure experimentale reproductible. Les demonstrations citees (C01, C02, C03, C06) sont des validations partielles avec evaluation LLM-sur-LLM. L'auteur est explicite : "These case studies validate the direction, not the destination."

Terminologie proprietaire. Le vocabulaire s'enrichit de termes supplementaires sans correspondance dans la litterature : Quorum Pattern, Mutual Genesis, Cross-Pollination, Build Type, Purpose Contract, Trust Arc, Autoprompting, etc. Le risque d'insularite terminologique se renforce.

Niveaux de maturite tres variables. Trois niveaux coexistent dans ces 10 papers : "Research Required" (B2, B3, B7), "Partially Implemented" (B1, B5, C1), "Implementable" (C2, C3), et "In Active Development" (B4). Le MA et l'apprentissage ecosystemique, qui sont les pieces les plus ambitieuses du cadre, sont au stade le plus precoc.

Circularites non resolues. Plusieurs mecanismes presentent des dependances circulaires : le MA requiert un track record pour gagner de l'autonomie, mais personne ne specifie comment le track record initial est evalue ; l'autonomie requiert un but, mais la definition du but est reservee aux humains ; les alias apprennent et s'ameliorent dans le temps, mais le mecanisme concret d'amelioration reste opaque.

5.2 Ce que le corpus apporte

Un modele d'apprentissage ecosystemique a quatre niveaux. La progression decision / processus / systeme / meta, avec le DTR, le PCP, le MA et la creation autonome d'agents comme incarnations respectives, est un modele architecturalement complet d'amelioration progressive des systemes agentiques. Il est non teste mais coherent.

Une triade de brokers comme infrastructure cognitive. La separation Context / Inference / Conversation, avec la dualite passe-assemble / present-configure comme principe organisateur, propose une decomposition utile du probleme de l'infrastructure agentique. La notion de build type comme vue pilotee par le but est une contribution pratique pour quiconque conçoit du context engineering.

Un modele d'autonomie progressive. L'arc de confiance a quatre phases, avec le paradoxe de l'autonomie et la notion d'autonomie gagnee, propose un vocabulaire utile pour penser la calibration de la supervision humaine dans les systemes agentiques. La formulation du paradoxe de l'autonomie -- un systeme qui admet ses limites est plus autonome qu'un systeme qui ne les admet jamais -- est une contribution conceptuelle qui resonance avec les debats actuels sur la calibration de la confiance dans les outputs LLM.

Le Quorum Pattern comme mecanisme de diversite cognitive. La distinction entre diversite structurelle (modeles differents) et diversite simulee (roles differents sur un seul modele), avec les compositions recombinaibles (Advisory, Document, Technical Asset, Diagnostic), est une contribution architecturale identifiable. Le principe de l'invariant unique -- la diversite de modeles est la seule condition obligatoire, tous les autres blocs sont optionnels -- donne au pattern une flexibilite notable.

L'autoprompting comme extension temporelle. La capacite d'un agent a s'auto-activer, a honorer des engagements futurs captures en conversation, et a reagir a des conditions systeme sans intervention humaine est un concept rarement traite dans la litterature sur les systemes agentiques. La separation nette entre l'autoprompter (qui sait quand mais pas quoi) et l'Imperator (qui sait quoi mais pas quand) est une contribution de design pattern.

5.3 Positionnement

Les blocs B et C se situent dans la meme veine que A et D : des concept papers prescriptifs, plus proches des position papers en architecture logicielle que des publications empiriques. Le bloc B est le plus speculatif de la serie -- le MA, l'apprentissage ecosystemique et l'expansion autonome de capacites sont des visions architecturales a l'horizon lointain. Le bloc C est le plus concret -- les trois brokers sont suffisamment specifiques pour guider une implementation.

L'ensemble B+C complete le cadre initie par A+D. Les fondations (A) definissent le vocabulaire, l'architecture (D) pose les mecanismes, la cognition (B) theorise l'apprentissage et l'autonomie, les composants (C) detaillent l'infrastructure cognitive. La coherence interne de la serie reste son trait le plus notable : les 23 papers forment un argument progressif ou chaque element fait reference aux autres de maniere verifiable.

La vision -- un ecosysteme qui apprend a se modifier, gouverne par un meta-architecte, peuple d'agents autonomes a personnalite emergente, soutenu par une triade de brokers -- est radicalement ambitieuse. Elle reste integralement a valider.

6. Annexe -- Index des papers couverts

#	ID	Titre	Bloc	Statut maturite
1	B1	The Quorum Pattern	B - Cognition & Autonomy	Partially Implemented
2	B2	Ecosystem Learning	B - Cognition & Autonomy	Research Required
3	B3	Agent Autonomy	B - Cognition & Autonomy	Research Required
4	B4	The Autonomous SDLC	B - Cognition & Autonomy	In Active Development
5	B5	AI Persona and Emergent Culture	B - Cognition & Autonomy	Partially Implemented
6	B6	Autoprompting		Conceptual

#	ID	Titre	Bloc	Statut maturite
			B - Cognition & Autonomy	
7	B7	The Metacognitive Architect	B - Cognition & Autonomy	Research Required
8	C1	The Context Broker	C - Core Components	Partially Implemented
9	C2	The Inference Broker	C - Core Components	Implementable
10	C3	The Conversation Broker	C - Core Components	Implementable

7. Sources

- Repo GitHub : <https://github.com/jmorrisette-RMDC/portfolio/tree/main/Joshua26/docs/concepts/>
 - Substack : jmorrisettermdc.substack.com
 - Serie : "Con conversationally Cognizant AI", mars 2026
 - Extractions structurees produites en S90-S91 (data/S90/extractions-morrisette/, data/S91/LL2_agents/)
 - LL1 : data/S90/LL1_MORRISSETTE_AD.md
-