

ARCADIA et Capella comme mécanisme de contrainte pour des agents IA en ingénierie système

Synthèse exécutive

ARCADIA peut servir de socle crédible pour **contraindre** (au sens “encadrer, borner et vérifier”) des agents IA qui assistent l’ingénierie système, parce qu’elle impose une **décomposition par niveaux** (besoin → solution), une **séparation de points de vue**, et une **traçabilité “Need ↔ Solution”** au cœur du raisonnement d’architecture. La référence ARCADIA explicite précisément cette logique outillée : comprendre le besoin client, définir/partager l’architecture, valider tôt, maîtriser l’IVVQ, et gérer des contraintes (coût, performance, sûreté, sécurité, masse, etc.). ¹

Pour “contraindre” un agent IA, ARCADIA est particulièrement utile si l’agent agit **sur le modèle** (création/modification d’éléments, allocation fonctions↔composants, cohérence des échanges) ou **à partir du modèle** (questions/résumés, génération de livrables, analyse d’impacts), car elle fournit des invariants vérifiables : cohérence inter-couches, complétude de certaines allocations, règles de nommage/typing, et surtout traçabilité. En revanche, ARCADIA n’est pas, à elle seule, une sémantique formelle “exécutable” garantissant la correction d’une décision IA : une part des exigences reste en langage naturel et beaucoup de justifications restent argumentatives. Il faut donc compléter ARCADIA par des **garde-fous déterministes** (règles de validation, contrôles de couverture, contraintes de graphe, éventuellement formalisation partielle) et une **revue humaine** systématique, ce qui rejoint les messages industriels (“assistant, pas décideur”). ²

Côté recherche et preuves d’application, on observe déjà des travaux et démonstrateurs **au-delà du logiciel** (aérospatial, équipements industriels, PLM, sûreté) qui couplent explicitement des modèles MBSE (dont Capella) à des LLM : génération d’architectures et d’éléments de modèle depuis des exigences, vérification d’exigences par raisonnement sur une représentation graphe extraite du modèle, agents RAG “knowledge partner” adossés à un modèle Capella, et pilotes industriels intégrant GenAI pour accélérer la construction et l’exploitation de modèles. ³

Enfin, la faisabilité technique d’une “contrainte par modèle” dépend moins de la méthode seule que de la **mise en données** : extraction du métamodèle (EMF/Ecore), exports, APIs/scripting, transformation en graphe/knowledge fabric, et validation (ex. SHACL sur RDF). Des solutions existent déjà (scripting, headless Python, OSLC/édition web, connecteurs), mais l’industrialisation impose une discipline outillage (versioning, CI/CD, traçabilité outillée, gestion multi-utilisateur). ⁴

Hypothèse de cadrage : aucune contrainte organisationnelle, réglementaire ou de criticité particulière n’est imposée a priori (le rapport propose donc une approche “générique”, adaptable à du critique). ⁵

Contexte et rappel sur ARCADIA et Capella

ARCADIA est une méthode d’ingénierie basée modèle, orientée architecture, conçue pour couvrir l’ingénierie système (et, dans une certaine mesure, les architectures logiciel/matériel), avec un objectif

explicite de support à l'IVVQ et à la conciliation de contraintes fortes. Elle a été développée par Thales ⁶ (2005–2010, itératif, multi-domaines), et est référencée comme norme française XP Z67-140 chez AFNOR ⁷ . ⁸

La référence ARCADIA (publique) insiste sur le fait qu'il s'agit d'une méthode **outillée**, "devoted to systems & architecture engineering", et détaille le raisonnement attendu : comprendre le besoin, définir et partager l'architecture, valider précocement, et maîtriser l'IVVQ. ⁹ Elle structure l'ingénierie en grandes activités, notamment : *Customer Operational Need Analysis*, *System Need Analysis*, *Design Logical Architecture*, *Design Physical Architecture*, et des activités de "building strategy/contracts" et d'IVVQ. ⁹

Capella est l'atelier MBSE open-source qui implémente la méthode ARCADIA, hébergé dans l'écosystème Eclipse Foundation ¹⁰ via PolarSys ¹¹ . Le projet est explicitement décrit comme "industrial-grade" et positionné pour l'architecture de systèmes complexes, avec un écosystème d'extensions (requirements, collaboration, digital thread, etc.). ¹²

Deux éléments sont importants pour la question "contraindre des agents IA" :

Premièrement, ARCADIA revendique une approche **viewpoint-driven** (séparation et articulation des points de vue) et une logique de support aux processus d'ingénierie inspirée des processus du cycle de vie (référence ISO/IEC/IEEE 15288 citée dans la référence ARCADIA). ¹

Deuxièmement, la référence ARCADIA publie aussi une "annexe" décrivant une **définition plus formalisée** des concepts (métamodèle de langage), ce qui ouvre la voie à une exploitation machine (extraction, transformation, contraintes automatiques). ⁹

ARCADIA comme dispositif de contrainte pour des agents IA

Dans ce rapport, "contraindre un agent IA" recouvre trois mécanismes complémentaires :

Contraindre les **actions** : l'agent ne peut proposer/commettre que des transformations compatibles avec le processus (ex. ne pas "sauter" directement au physique si le logique n'existe pas), ni casser des invariants de modélisation (types, allocations, cardinalités, règles d'architecture). ⁹

Contraindre les **réponses** : l'agent, même conversationnel, doit s'appuyer sur le contenu du modèle (grounding) et restituer des résultats traçables (liens, justifications, couverture) plutôt que des réponses "plausibles". Les démonstrations récentes d'assistants sur modèles (ex. "knowledge partner" adossé à un modèle Capella) montrent justement un couplage RAG/embeddings/modèle pour borner l'analyse. ¹³

Contraindre la **qualité** : l'agent est évalué et corrigé par des boucles déterministes (règles), et par une revue humaine, car la méthode elle-même vise la rigueur mais ne remplace pas la validation d'ingénierie. Ce point est explicitement assumé par des acteurs outillage ("powerful assistant, not a decision-maker") et par la littérature sur génération d'architectures (l'humain reste central pour une sortie de qualité). ¹⁴

Pourquoi ARCADIA est structurellement favorable

ARCADIA fournit une **grammaire d'architecture** qui est immédiatement exploitable comme "espace de contraintes" pour un agent :

- Les niveaux (besoin → solution) sont explicites et la référence ARCADIA documente les activités et données produites/consommées, y compris une vue "Need/Solution traceability" et des actifs d'IVVQ/justification. ⁹
- La méthode est conçue pour gérer des systèmes sous contraintes fortes (coût, performance, sûreté, sécurité, masse, etc.) et pour faciliter l'IVVQ, ce qui correspond précisément aux zones où l'on veut empêcher un agent de "prendre des libertés". ¹
- La présence d'un métamodèle (même "semi-formalisé") signifie qu'on peut générer des règles de cohérence et des extracteurs, puis vérifier automatiquement les états du modèle avant/après actions de l'agent. ¹⁵

En pratique, cela permet de transformer ARCADIA en "contrat" d'action pour un agent, par exemple : *l'agent peut proposer des modifications dans une couche donnée, mais doit prouver qu'il maintient la cohérence des allocations, des échanges et des liens de traçabilité, et fournir un diff/revue.*

Ce qu'ARCADIA ne garantit pas, et pourquoi c'est critique pour l'IA

ARCADIA est une méthode d'architecture et de raisonnement outillé ; elle n'est pas une preuve formelle de correction d'un design, ni un langage d'exigences complet. La référence indique d'ailleurs que les documents requis par des standards peuvent être produits depuis les modèles, ce qui confirme que le modèle sert de "source de vérité", mais la qualité dépend du contenu et des validations aval. ⁹

Deux limitations récurrentes apparaissent quand on veut "enfermer" une IA :

- **Ambiguïté des exigences** : beaucoup d'exigences restent textuelles, et même si ARCADIA organise la traçabilité, l'interprétation d'une exigence n'est pas automatiquement déterministe (d'où l'intérêt de boucles d'inférence/validation contrôlées et de vérification humaine). ¹⁶
- **Sémantique partiellement implicite** : les diagrammes et relations portent beaucoup de sens "méthodologique". Pour contraindre un agent, il faut expliciter ce sens en règles (contraintes de graphe, cardinalités, patterns autorisés, etc.) et en choix d'autorisations (RBAC, droit à écrire sur certaines zones du modèle). Les projets de vérification d'exigences sur graphe extrait d'un modèle montrent justement que l'on passe par une représentation structurée et des étapes de filtrage/ranking pour rester dans le budget de contexte des LLM. ¹⁷

Conclusion analytique : ARCADIA est un bon langage de **contrainte de processus et de structure**, mais doit être complété par un **niveau de contrainte computationnel** (validation/règles) si l'on veut un agent IA opérant de façon fiable.

État actuel de la recherche

Couplage LLM ↔ modèles MBSE (incluant Capella)

Un signal fort (au-delà des effets d'annonce) est la publication d'expérimentations où un LLM est **couplé à un environnement MBSE** pour générer des éléments d'architecture, avec une évaluation qualitativement orientée traçabilité/exigences.

L'article de 2025 (Université de University of Bristol ¹⁸) sur la génération d'architectures de systèmes spatiaux décrit un outillage Python couplant un LLM à Capella pour générer architecture, fonctions, modes et composants à partir d'un jeu d'exigences; les auteurs rapportent une qualité correcte pour certains artefacts (modes/composants) et soulignent que l'assistant IA reste dépendant d'une validation et d'un pilotage humain. ¹⁹

Un autre axe de recherche vise non pas la génération, mais la **vérification d'exigences** par raisonnement contrôlé sur une représentation graphe du modèle. Le préprint 2025 "Inference-Time Intervention..." (Université de University of Strathclyde ²⁰) utilise deux modèles Capella (missions spatiales), transforme le modèle en graphe textuel (triples), filtre les entités pertinentes par similarité sémantique et applique une technique de "steering" (interventions à l'inférence) pour contrôler précision/rappel dans une tâche critique de vérification d'exigences. ¹⁷

Ces deux travaux sont directement pertinents pour ARCADIA comme "contrainte": ils montrent que (1) la valeur vient du **grounding sur un modèle structuré**, (2) la fiabilité passe par une **pipeline outillée** (extraction, sélection, formatage, contrôle), et (3) l'humain reste dans la boucle. ²¹

Projets, feuilles de route et cadrages "MBSE Co-Pilot / AI Assistant"

Même si certaines publications centrales peuvent être derrière paywall, le concept de "MBSE Co-Pilot" est explicitement formulé comme un **programme de recherche** visant une machine IA pour augmenter les activités MBSE (génération, analyse, exploitation), ce qui converge avec votre cas d'usage "agent contraint par la méthode et le modèle". ²²

Du côté communautés professionnelles, IfSE ²³ (UK) publie une description détaillée d'un papier "MBSE AI Assistant / Co-Pilot" introduisant un cadre de capacités et une échelle de niveaux d'autonomie (mappée sur des catégorisations analogues à la robotique/AV), en insistant sur le fait que l'assistant IA doit collaborer avec l'ingénieur et standardiser/automatiser des tâches MBSE sans le remplacer. ²⁴

Enfin, des cadres plus larges "MBSE 2.0" mettent l'IA comme composant transversal d'intégration du cycle de vie (modélisation, simulation, optimisation), mais cela reste une vision macro: pour votre problématique, l'important est d'identifier où une méthode d'architecture (ARCADIA) fournit un "squelette" exploitable comme garde-fou d'agent. ²⁵

GenSE et industrialisation de l'IA en ingénierie système (au-delà du logiciel)

Au-delà de Capella, des travaux comme GenSE (papier 2025, France, secteur nucléaire) décrivent une intégration de GenAI dans des processus d'ingénierie système orientés exigences: extraction, reformulation, allocation PBS, contrôle qualité (guidelines), conformité aux standards, et maintien du "human-in-the-loop". Les gains reportés en temps (ordre de grandeur) sont très importants, mais l'article souligne explicitement les risques de résultats probabilistes et la nécessité de post-traitements et de validation humaine pour des domaines critiques. ²⁶

Pour ARCADIA, l'enseignement est double: (1) la "contrainte" doit être pensée comme un **système de contrôle** (règles + HITL), pas comme une simple méthode, (2) la valeur la plus industrialisable à court terme est souvent du côté **exigences/traçabilité/industrialisation documentaire**, avant même la génération d'architectures complètes. ²⁷

Applications industrielles et retours d'expérience au-delà du logiciel

Adoption industrielle de Capella (domaines variés)

Le site de Capella indique une utilisation sur des projets concrets dans plusieurs secteurs (aéronautique, énergie, transport, etc.) et maintient des pages “adopters” et “case studies” illustrant des déploiements réels. ²⁸

À titre d'exemples de secteurs (au-delà du logiciel), la page “case studies” met en avant Thermo Fisher Scientific ²⁹ (healthcare), Naval Group ³⁰ (naval & défense) et Rolls-Royce ³¹ (aviation). ³²

Le site “webinars” mentionne aussi une expérimentation où CNES ³³ aurait utilisé des modèles Capella (avec extensions) pour démontrer l'amélioration d'un processus document-centrique sur un cas spatial (SVOM), ce qui confirme des usages “systèmes” bien au-delà du logiciel purement applicatif. ³⁴

Pilotes et démonstrateurs explicitement orientés IA + MBSE

Un cas particulièrement instructif (car explicitement “production pilot”) est Applied Materials ³⁵ : leurs slides (Capella Days 2024) indiquent une évaluation commencée en 2023, un “Production Pilot” en 2024, et une roadmap “What's next” qui inclut l'usage de GenAI pour créer automatiquement des éléments MBSE (RAG stack + LLM + pycapellambse), ainsi que la création programmatique de diagrammes d'architecture (LLM + Python + APIs). ³⁶

Maturité (au vu des éléments publics) : pilote en production / industrialisation progressive dans un contexte PLM/digital thread, encore avec POCs planifiés. ³⁷

Autre signal fort : un support “knowledge partner” basé sur Capella présenté dans des slides (Capella Days 2025) côté Siemens Digital Industries Software ³⁸. On y voit un notebook “Retrieval Augmented Generation... based on Capella” et une liste d'opérations outillées (création d'artefacts, requêtes sur un modèle Capella via embeddings, génération d'une “knowledge fabric” YAML, notebook de replay). Cela correspond exactement à une architecture d'agent “contraint par la matière modèle”, parce que l'agent consomme/produit des artefacts traçables et rejouables. ¹³

Maturité : démonstrateur outillé (notebook/agent), orienté exploitation et capitalisation, avec une trajectoire annoncée vers mises à jour modèle/exigences/tests. ³⁹

Enfin, du côté éditeurs de l'écosystème, Obeo ⁴⁰ indique avoir prototypé la faisabilité d'intégrations IA-MBSE (assistant de modélisation et interface NL d'accès au modèle) avec des outils incluant Capella, et travailler à une “productisation”, tout en rappelant explicitement que l'IA ne remplace pas la validation humaine. ⁴¹

Approches techniques pour rendre ARCADIA exploitable par des agents

Typologie de stratégies “machine-readable”

Pour qu'un agent IA soit effectivement contraint par ARCADIA, il faut transformer “la méthode + le modèle” en **interfaces, formats et règles**. Les stratégies se classent en quatre familles (souvent combinées) :

Accès direct au modèle via métamodèle et APIs : Capella est basé sur Eclipse Modeling Framework ⁴² (EMF), ce qui implique un métamodèle Ecore et des modèles sérialisés en ressources EMF, interrogeables via APIs Java et langages de requête. ⁴³

Conséquence : un agent peut appliquer une politique “read-mostly” (analyse) ou “read/write” (mutation contrôlée) via des APIs, mais on doit instrumenter validation/diff.

Scripting côté outil (dans le workbench) : Python4Capella est un add-on de scripting, mais des échanges du forum indiquent qu’il n’est pas conçu pour tourner “hors environnement” (la couche Python délègue vers l’API Java) et que pour serveur/CI on s’oriente plutôt vers des usages en mode commande. ⁴⁴

Accès headless (hors Capella) : la librairie open-source py-capellambse (capellambse) revendique explicitement lecture/écriture de modèles Capella depuis Python sans Java ni l’outil, ce qui est particulièrement utile pour un pipeline agentique en CI/CD ou microservice. ⁴⁵

Publication et interop (web / ALM / PLM) : l’écosystème inclut des solutions de publication web et de liaison à d’autres artefacts (exigences, change, tests) via OSLC (ex. “Publication for Capella”, et intégrations mentionnées avec Jira/Confluence). ⁴⁶

C’est un levier majeur pour “contraindre l’agent” : l’agent n’agit plus seulement sur le modèle, mais dans une **chaîne de traçabilité** multi-outils.

Conversion vers graphe / knowledge graph et contraintes

Le passage “modèle → graphe” est récurrent dans les travaux de recherche et démos, car il sert à deux choses : (1) fournir une représentation compacte adaptée au contexte d’un LLM, (2) offrir un support à des règles déterministes.

Le préprint de vérification d’exigences sur modèles Capella décrit explicitement une extraction en représentation greffée sur le modèle (sélection via embeddings, BFS, triples “Entity-Relation-Entity”), ce qui est une transposition directe vers un graphe de connaissances. ¹⁷

Sur le plan des standards, si vous choisissez RDF/Linked Data, RDF formalise le graphe en triplets. ⁴⁷ En JSON-LD, on peut sérialiser ces graphes de manière compatible avec des environnements web et APIs JSON. ⁴⁸ Pour exprimer des *règles de validation* sur ces graphes, SHACL est une recommandation W3C dédiée à la validation de graphes RDF par des “shapes” (contraintes). ⁴⁹

Dans un contexte digital thread, OSLC Core 3.0 (OASIS) est pertinent car il définit un cadre d’interopérabilité REST/Linked Data, historiquement très utilisé pour relier exigences-changement-tests-architecture. ⁵⁰

Un point concret côté écosystème Capella : un projet open-source “Capella OSLC Adapter” propose d’exposer des données EMF extraites de Capella via un serveur embarqué et un serveur OSLC, avec une mention explicite que le mapping est expérimental et read-only, ce qui illustre à la fois la faisabilité et la difficulté de l’alignement sémantique. ⁵¹

Collaboration, versioning et CI/CD

Pour industrialiser un agent IA contraint par modèle, la gestion multi-utilisateur et l’intégration outillage sont déterminantes.

Les déploiements multi-utilisateurs reposent souvent sur des serveurs de modèles (ex. Team for Capella), dont la documentation d'installation décrit une architecture serveur/administration/clients.

⁵² La même documentation mentionne des pratiques de versioning (ex. modèle exporté + SCM) et l'écosystème discute de mécanismes de sécurisation (ex. SSL côté serveur CDO). ⁵³

Pour CI/CD et génération automatique de livrables, des usages en ligne de commande existent : par exemple, le forum documente un lancement de génération M2Doc via l'application command-line Capella (arguments -application / -appid / -data, etc.). ⁵⁴

Pour transformations/validations automatisées, des outils de l'écosystème Eclipse comme Epsilon documentent explicitement des cas de requête/validation/transformation sur des modèles Capella. ⁵⁵

Architecture d'agents IA contraints par MBSE et recommandations de prototypage

Architecture de référence agent ↔ modèle

Le schéma ci-dessous est une proposition pragmatique d'architecture "agent contraint", compatible avec ARCADIA : l'agent ne parle pas seulement "langage naturel", il manipule des **artefacts structurés** (sélections de modèle, patches, rapports) et passe par des **vérifications déterministes** avant toute écriture persistée.

flowchart TD

U[Ingénieur système] -->|Intentions / questions (NL)| A[Agent IA]

A -->|Récupère contexte| RAG[RAG sur modèle + docs]

RAG --> A

A -->|Propose plan + patches| P[Planificateur d'actions]

P -->|Lecture/écriture via API| M[(Référentiel de modèles)]

M -->|Extraction / snapshot| G[Représentation graphe / "fabric"]

G --> V[Validations déterministes]

V -->|Contraintes de structure
+ complétude traceability| V

V -->|OK| M

V -->|KO + diagnostic| A

M -->|Diff + rapport de traçabilité| U

U -->|Approbation / corrections| M

M --> S[Analyses/simulations/IVVQ outillées]

S -->|Résultats + écarts| V

Points de conception associés à ARCADIA :

- L'agent est **contraint par le niveau de travail** (OA/SA/LA/PA) et par les artefacts attendus (besoin, solution, traceability, IVVQ), ce qui est cohérent avec la référence ARCADIA. ⁹
- Le "garde-fou" dur est la brique de validation : c'est là qu'on encode vos règles ARCADIA (ou votre profil interne) et qu'on refuse une mutation invalide.

- Les démos “knowledge partner” et les pilotes industriels montrent la valeur d’une gestion d’artefacts rejouables (notebook de replay, artefacts de sélection, etc.), ce qui est exactement une stratégie de réduction du risque agentique. ⁵⁶

Exemple de mapping modèle → knowledge graph

Ci-dessous un ER minimal (à adapter) qui couvre le cœur “contraignable” ARCADIA: composants, fonctions, échanges, exigences, liens de traçabilité.

```
erDiagram
    COMPONENT ||--o{ COMPONENT : "decompose"
    FUNCTION ||--o{ FUNCTION : "decompose"

    COMPONENT ||--o{ ALLOCATION : "allocates"
    FUNCTION ||--o{ ALLOCATION : "is_allocated_to"

    COMPONENT ||--o{ EXCHANGE : "source"
    COMPONENT ||--o{ EXCHANGE : "target"
    EXCHANGE }o--|| FUNCTION : "realized_by(optional)"

    REQUIREMENT ||--o{ TRACE_LINK : "traced_to"
    FUNCTION ||--o{ TRACE_LINK : "satisfies"
    COMPONENT ||--o{ TRACE_LINK : "implements"

    COMPONENT ||--o{ PROPERTY : "has"
    FUNCTION ||--o{ PROPERTY : "has"
```

Si vous encodez ce graphe en RDF, SHACL peut exprimer des contraintes comme “toute Fonction de niveau X doit être allouée à au moins un Component”, ou “toute Requirement de type sécurité doit tracer vers une Fonction et une stratégie de V&V”. RDF/JSON-LD/SHACL sont précisément conçus pour représenter et valider des graphes structurés. ⁵⁷

Principaux défis d’intégration (à anticiper)

Gestion de la confidentialité et des droits d’écriture: l’industrialisation passe par des politiques d’accès (ex. serveurs de modèles, chiffrement transport, restriction des zones modifiables, logs). Les discussions autour de serveurs multi-utilisateurs mentionnent des mécanismes comme SSL côté serveur. ⁵⁸

Gestion des grands modèles (taille, performance, diffs): les modèles EMF volumineux et la granularité des représentations rendent la comparaison/merge difficile ; une stratégie “snapshot + patch” et une extraction “subset” (pour rester dans les context windows) est cohérente avec les pipelines décrits dans la littérature (sélection top-k, re-ranking, BFS). ¹⁷

Rigueur de traçabilité et prévention de la “fausse assurance”: les travaux sur assistants rappellent que le gain de productivité ne doit pas masquer le besoin de validation et la possibilité d’erreurs/hallucinations, surtout dans des domaines critiques. ⁵⁹

Interop digital thread: si votre agent doit agir “au-delà du modèle” (exigences dans un RM, change dans un ALM, tests), l’alignement OSLC/Linked Data est une option structurante. ⁶⁰

Recommandations concrètes de prototypage

Recommandation structurante: démarrer par un prototype où ARCADIA contraint **le périmètre d'action** et **la vérification**, plutôt que d'ambitionner une génération autonome d'architecture complète dès la première itération. Les pilotes industriels mettent souvent "GenAI pour accélérer la construction/exploitation" comme un axe progressif, pas un remplacement. ⁶¹

Sous-ensemble ARCADIA minimal (MVP) à viser (orienté "contrainte") :

- Un niveau d'architecture cible (souvent SA+LA, ou LA seule au début).
 - Les objets essentiels : Component / Function / Exchange / Requirement / TraceLink / Property.
 - Un petit nombre de règles "dures" (10-30) : typage, allocations minimales, cohérence d'échanges, complétude de traçabilité sur un set d'exigences priorisées, règles de nommage.
- Justification : c'est le noyau commun aux approches "graph extraction + validation" et c'est aligné avec les données ARCADIA "Need/Solution traceability". ⁶²

Stack outillage (suggestion) :

- Extraction/édition : py-capellambse (headless) pour pipelines, et/ou scripting in-tool si vous voulez du "live authoring". ⁶³
- Publication/interop : OSLC (si digital thread multi-outils) ou un export static + API interne. ⁶⁰
- Graphe+contraintes : RDF/JSON-LD + SHACL pour valider. ⁶⁴
- CI/CD : génération de rapports/livrables et validations en ligne de commande (ex. génération documentaire, scripts). ⁵⁴

Métriques d'évaluation (à instrumenter dès le départ) :

- Taux de violations de règles (avant/après agent) et temps moyen de résolution.
- Couverture de traçabilité (exigences prioritaires → fonctions → composants).
- "Human acceptance rate" des patchs proposés (proportion acceptée sans retouche, acceptée avec retouche, rejet).
- Temps d'exécution bout-en-bout (prompt→patch validé→rapport) et coût d'exploitation (tokens + compute).
- Robustesse (tests de non-régression sur jeux de modèles) et rejouabilité (artefacts + logs).

Comparaison de trois options de prototype

Option	Objectif principal	Périmètre ARCADIA	Stack typique	Délai indicatif	Ressources indicatives	Bénéfices attendus	Risques majeurs
Lightweight	Q/R sur modèle + extraction de vues + rapports	Lecture seule, 1 couche (ex. LA)	Export subset + RAG/ embeddings + rapports	4-6 semaines	1-2 dev + 1 SE référent	Valeur rapide (accès au modèle, capitalisation), faible risque d'intégrité	Risques "hallucinations" si gros incohérences valeurs si pas d'écriture

Option	Objectif principal	Périmètre ARCADIA	Stack typique	Délai indicatif	Ressources indicatives	Bénéfices attendus	Risques majeurs
Hybrid	Propositions de patches + validations déterministes + HITL	Lecture/écriture contrôlée, 1-2 couches, règles "dures"	py-capellambse + graphe + SHACL + pipeline CI	8-12 semaines	2-4 dev + 1-2 SE + 0,5 devops	Cœur "agent contraint" : gains sur tâches répétitives, cohérence mesurable, traceability renforcée	Coût d'ingénierie des règles de contraintes, gestion des merges, la gouvernance
Full-scale	Agent multi-outils (RM/ALM/PLM) + digital thread + génération + V&V avancée	Plusieurs couches + assets IVVQ + interop complète	OSLC + publication web + référentiel modèles + validations + workflows	4-9 mois	4-8 dev + SE/IVVQ + devops + sécurité	Transformation de processus (industrialisation), bénéfice transversal, scalabilité organisationnelle	Complexité d'intégration, sécurité, confiance, dette d'exploration, nécessité d'une gouvernance forte

Bibliographie priorisée (utile pour cadrer rapidement un POC): la référence ARCADIA (structure, données, métamodèle) ⁹ ; le papier 2025 sur génération d'architectures MBSE+LLM et traçabilité ¹⁹ ; le préprint sur vérification d'exigences par graphe extrait de modèle Capella et contrôle au moment de l'inférence ¹⁷ ; les preuves industrielles CapellaDays (Applied Materials, Siemens) ⁶⁵ ; et les standards RDF/JSON-LD/SHACL/OSLC pour bâtir des contraintes vérifiables et une interop digital thread ⁶⁶

¹ ⁵ ⁶ ⁹ ¹⁵ ⁴² ⁶² <https://mbse-capella.org/resources/arcadia-reference/html/output/ARCADIA/1a79525b3b624a7094acd0b580842ce6.html>

<https://mbse-capella.org/resources/arcadia-reference/html/output/ARCADIA/1a79525b3b624a7094acd0b580842ce6.html>

² ¹¹ ¹⁴ ¹⁶ ³⁵ ⁴¹ ⁵⁹ <https://blog.obeiosoft.com/ai-integrated-as-an-mbse-modeling-assistant-in-syson>

<https://blog.obeiosoft.com/ai-integrated-as-an-mbse-modeling-assistant-in-syson>

³ ¹⁹ ²¹ ³⁰ https://www.researchgate.net/publication/388476324_Assessment_of_large_language_models_for_use_in_generative_design_of_model_based_spacecraft_system_architectures

⁴ ³³ ⁴³ <https://docs.couchbase.com/home/contribute/generate-rest-api.html>

⁷ ¹⁷ <https://arxiv.org/html/2503.14130v1>

⁸ <https://mbse-capella.org/arcadia.html>

¹⁰ ⁴⁹ <https://www.w3.org/TR/shacl/>

- 12 <https://projects.eclipse.org/projects/polarsys.capella/releases/6.1.0/review>
<https://projects.eclipse.org/projects/polarsys.capella/releases/6.1.0/review>
- 13 20 23 39 56 https://mbse-capella.org/resources/pdf/capelladays2025/CapellaDays2025_Slides_SIEMENS.pdf
https://mbse-capella.org/resources/pdf/capelladays2025/CapellaDays2025_Slides_SIEMENS.pdf
- 18 28 <https://mbse-capella.org/>
<https://mbse-capella.org/>
- 22 <https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.70011>
<https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.70011>
- 24 29 https://ifse.org.uk/Program_Files/Calendar/Calendar?CatID=Events%3FCatID%3DEvents&NoOfRows=25
https://ifse.org.uk/Program_Files/Calendar/Calendar?CatID=Events%3FCatID%3DEvents&NoOfRows=25
- 25 40 <https://www.mdpi.com/2079-8954/13/7/584>
<https://www.mdpi.com/2079-8954/13/7/584>
- 26 27 <https://www.scitepress.org/Papers/2025/134434/134434.pdf>
<https://www.scitepress.org/Papers/2025/134434/134434.pdf>
- 31 47 57 64 66 <https://www.w3.org/TR/rdf11-concepts/>
<https://www.w3.org/TR/rdf11-concepts/>
- 32 https://mbse-capella.org/case_studies.html
https://mbse-capella.org/case_studies.html
- 34 <https://mbse-capella.org/webinars.html>
<https://mbse-capella.org/webinars.html>
- 36 37 61 65 https://mbse-capella.org/resources/pdf/capelladays2024/CapellaDays2024_Slides_Applied-Materials.pdf
https://mbse-capella.org/resources/pdf/capelladays2024/CapellaDays2024_Slides_Applied-Materials.pdf
- 38 52 53 https://www.obeosoft.com/download/release/team-for-capella/6.1/6.1.0/smw/TeamForCapella_InstallationGuide_6.1.0_SMW_2312.pdf
https://www.obeosoft.com/download/release/team-for-capella/6.1/6.1.0/smw/TeamForCapella_InstallationGuide_6.1.0_SMW_2312.pdf
- 44 <https://forum.mbse-capella.org/t/to-run-python4capella-api-outside-of-capella-smw-environment/7362>
<https://forum.mbse-capella.org/t/to-run-python4capella-api-outside-of-capella-smw-environment/7362>
- 45 63 <https://github.com/DSD-DBS/py-capellambse>
<https://github.com/DSD-DBS/py-capellambse>
- 46 <https://www.obeosoft.com/en/products/publication-for-capella>
<https://www.obeosoft.com/en/products/publication-for-capella>
- 48 <https://www.w3.org/TR/json-ld11/>
<https://www.w3.org/TR/json-ld11/>
- 50 60 <https://www.oasis-open.org/standard/oslc-core-version-3-0/>
<https://www.oasis-open.org/standard/oslc-core-version-3-0/>
- 51 <https://github.com/vojtechspevak/CapellaOslcAdapter>
<https://github.com/vojtechspevak/CapellaOslcAdapter>

54 <https://forum.mbse-capella.org/t/m2doc-command-line-interface/4430>

<https://forum.mbse-capella.org/t/m2doc-command-line-interface/4430>

55 <https://eclipse.dev/epsilon/doc/articles/>

<https://eclipse.dev/epsilon/doc/articles/>

58 <https://forum.mbse-capella.org/t/team-and-sensitive-data/4827>

<https://forum.mbse-capella.org/t/team-and-sensitive-data/4827>