Legacy Transformation Framework

Obiettivi e stato dei lavori

Data: 13/01/2026

Links group

AGENDA

1. GAIA O.R.5 – «Legacy Modernization in Banking»

2. Stato dell’arte di tecniche e metodologie per la Legacy Modernization (attività 5.2)

3. Definizione del framework per la Legacy Modernization (attività 5.3)

1.

GAIA – O.R. 5

Legacy Modernization in Banking

O.R. 5 - Legacy Modernization in Banking

- I sistemi informativi bancari tradizionali (legacy) presentano **limiti di efficienza, scalabilità e flessibilità** che ostacolano la piena attuazione della trasformazione digitale e la competitività
- La **modernizzazione dei sistemi legacy** è **una delle principali strategie** sempre più centrale nei piani di investimento degli Istituti bancari
- La scelta di intraprendere un percorso di trasformazione è guidata da una combinazione di fattori strutturali e strategici:
  - **crescente complessità del settore bancario,** in termini di prodotti e canali di vendita, richiede soluzioni IT flessibili e scalabili
  - affermazione delle **tecnologie emergenti**, come intelligenza artificiale e big data
  - **fattori regolamentari**, legati a requisiti di sicurezza e trasparenza sempre più stringenti
  - l'evoluzione verso un **mercato client-centric** orientato alla customer experience sollecita una revisione dei modelli di business, superando l'approccio tradizionalmente product-centric
- La modernizzazione dei sistemi legacy pone la **sfida** di **coniugare** l'adozione di **soluzioni software innovative**, coerenti con il paradigma del Banking as a Service, con il mantenimento dei **requisiti di robustezza** operativa e **conformità normativa** propri dei sistemi esistenti
- Le attività di ricerca industriale di GAIA si focalizzano sullo studio e sulla sperimentazione della legacy modernization e alla definizione di un **framework metodologico e tecnologico a supporto della trasformazione dei sistemi informativi bancari**

A 5.1	A 5.2	A 5.3	A 5.4
S.o.t.a. Framework mercato + Quadro Normativo	S.o.t.a. Metodologie Modernizzazione	Definizione del Framework	Sperimentazione del Framework

O.R. 5 - Legacy Modernization in Banking

A 5.1 – Analisi del contesto di mercato e quadro normativo in ambito bancario

- Analisi dei principali framework di mercato per la modernizzazione dei software legacy in ambito bancario
- Esame della normativa di riferimento per la compliance bancaria nel settore del credito

A 5.2 – Studio dello stato dell'arte di tecniche e metodologie per la modernizzazione del software legacy

- Analisi delle tecniche e delle metodologie per la re-ingegnerizzazione dei software legacy
- Esame delle principali tecniche e metodologie AI-based a supporto del software engineering

O.R. 5 - Legacy Modernization in Banking

A 5.3 – Definizione del framework per la Legacy Modernization

- Progettazione del framework metodologico per la legacy modernization:
  - implementazione dei principali processi e metodologie di modernizzazione del software
  - utilizzo di tecnologie di AI generativa a supporto dell'intero ciclo di vita applicativo
  - gestione della conformità come un layer distinto del sistema informativo
- Prototipo del framework per la legacy modernization

A 5.4 – Sperimentazione del framework per la legacy modernization in ambito bancario

- Pianificazione ed implementazione della sperimentazione in laboratorio del framework
- Raccolta ed analisi dei risultati sperimentali

2.

Stato dell’arte di tecniche e metodologie per la Legacy Modernization (*attività 5.2*)

Organizzazione dello Studio di stato dell'Arte

Tecniche e Metodologie per la modernizzazione dei software legacy

- T. & M. per il Reverse Engineering del Software Legacy
- T. & M. per il Reverse Engineering delle Basi Dati
- T. & M. per l'implementazione del codice trasformato
- T. & M. per il Testing
- Metriche per la misurazione del risultato della trasformazione

SW Reverse Engineering: principali approcci in letteratura

Approcci Rule-based / Sintattici

Reverse engineering di tipo principalmente **sintattico**: si analizza il codice tramite **Abstract Syntax Tree (AST) e regole esplicite deterministiche**, ottenendo una comprensione strutturale.

Punti di forza: approcci **deterministici e spiegabili**.

Limiti: **fortemente dipendenti** dal linguaggio e **poco efficaci** nel catturare semantica e architettura di alto livello.

Riferimenti: motori rule-based (es. ProLeap); confronto sistematico in Lano (2024).

Approcci Model-Driven Reverse Engineering (MRDE)

Focus sulla **semantica**: si applica astrazione da AST a modelli semantici **UML/OCL** che rappresentano esplicitamente **struttura e comportamento** del sistema, indipendentemente dal linguaggio, e sono utilizzabili per **comprensione, analisi e re-ingegnerizzazione**

Punti di forza: forte preservazione semantica, tracciabilità e supporto alla modernizzazione di sistemi legacy.

Limiti: accuratezza semantica dei tool di astrazione in modelli UML/OCL

Riferimenti: Lano et al. (2024–2025), AgileUML, CSTL/CGTL.

Approcci NLP

Si basano su modelli NLP con LLM (*Large Language Models*) che **apprendono** rappresentazioni latenti del codice (*embedding*) per **catturare pattern strutturali e semantici**

Punti di forza: supporto a **comprensione del codice**, **clustering** di moduli, estrazione di dipendenze e ***summarization*** architetturale.

Limiti: **risultati probabilistici e modelli impliciti**, difficili da ispezionare (spiegabilità) o usare per re-ingegnerizzazione formali; **allucinazioni**; **perdita di contesto** su codebase larghe.

Riferimenti: GraphCodeBERT, SPT-Code, CodeIE; SLR di Hou et al. (2024).

Approcci Ibridi: AI-aided MDRE

(Stato dell'arte) Integrazione: gli **LLM accelerano e ampliano** l'estrazione delle informazioni e dei pattern semantici - con prompt guidati e tecniche RAG; il **MDRE** fornisce **struttura, validazione e controllo semantico** a livello di architettura con in modelli espliciti UML/OCL.

Punti di forza: combinazione di **automazione e rigore ingegneristico**, riducendo allucinazioni, perdita di contesto e sforzo manuale.

Limiti: **Non-determinismo**, **allucinazioni**, **spiegabilità**, **scalabilità** a codebase larghe

Riferimenti: LLM4Models (Siala & Lano, 2024–2025), MDRE-LLM (Boronat, 2025).

Bibliografia *(estratto)*

Hou, X. et al. (2024). *Large Language Models for Software Engineering: A Systematic Literature Review*. DOI: 10.1145/3695988

Lano, K., Siala, H (2024). *Using model-driven engineering to automate software language translation*. DOI: 10.1007/s10515-024-00419-y

Siala, H. (2024). *Enhancing Model-Driven Reverse Engineering Using Machine Learning*. DOI: 10.1145/3639478.3639797

Siala, H., Lano, K. (2025). *Towards Using LLMs in the Reverse Engineering of Software Systems to Object Constraint Language*. DOI:10.1109/SANER64311.2025.00096

Lano, K. et al. (2025). *Comparing LLM-based and Model-driven program translation*. DOI: 10.1109/ICoSSE65712.2025.00021

Boronat, A., Mustafa, J. (2025). *MDRE-LLM: A Tool for Analyzing and Applying LLMs in Software Reverse Engineering*. DOI: 10.1109/SANER64311.2025.00090.

Yang, D. et al. (2025). *DocAgent: A Multi-Agent System for Automated Code Documentation Generation*. DOI: 10.18653/v1/2025.acl-demo.44

Lano, K. Et al. (2024). *A Concrete Syntax Transformation Approach for Software Language Processing*. DOI: 10.1007/s42979-024-02979-y

Khachouch, M.K., et al. (2023). *Architecture Driven Modernization: A Review on Reverse Engineering Techniques based on Models' Approach*. DOI: 10.37394/23209.2023.20.32

Lano, K. Et al. (2023). *Program Abstraction and Re-Engineering: An Agile MDE Approach*. DOI: 10.1109/MODELS-C59198.2023.00050

Lano, K. et al. (2024). *Agile model-driven re-engineering*. DOI: 10.1007/s11334-024-00568-z

Siala, H. & al. (2024). *Model-Driven Approaches for Reverse Engineering - A Systematic Literature Review*. DOI: 10.1109/ACCESS.2024.3394732.

Shubham, G. et al. (2024). *Translation of Low-Resource COBOL to Logically Correct and Readable Java leveraging High-Resource Java Refinement*. DOI: 10.1145/3643795.3648388

Zhao, W. X. Et al. (2023). *A Survey of Large Language Models*. DOI: 10.48550/arXiv.2303.18223

Roziere, B. et al. (2020). *Unsupervised translation of programming languages*. DOI: 10.48550/arXiv.2006.03511

Heitlager, I. et al. (2007). *A Practical Model for Measuring Maintainability*. DOI: 10.1109/QUATIC.2007.8.

Hemmat, A. et al. (2025) *Research directions for using LLM in software requirement engineering: a systematic review*. DOI: 10.3389/fcomp.2025.1519437

Nam, D. et al. (2024). *Using an LLM to Help With Code Understanding*. DOI: 10.1145/3597503.3639187

Megargel, A. et al. (2020). *Migrating from monoliths to cloudbased microservices: A banking industry example*. (2020). DOI: DOI:10.1007/978-3-030-33624-0_4

Khadka, R. et al. (2013). *Migrating a large scale legacy application to SOA: Challenges and lessons learned*. DOI: 10.1109/WCRE.2013.6671318.

Assunção, W. K. G. et al. (2025). *Contemporary Software Modernization: Strategies, Driving Forces, and Research Opportunities*. DOI: 10.1145/3708527

Fuhr, A. et al (2013). *Model-driven software migration into service-oriented architectures*. DOI: 10.1007/s00450-011-0183-z

Chen, L. et al. (2024). *A Survey on Evaluating Large Language Models in Code Generation Tasks*. DOI: 10.48550/arXiv.2408.16498.

Papineni, K. et al. (2002). *BLEU: a Method for Automatic Evaluation of Machine Translation*. DOI: 10.3115/1073083.1073135

Ren, S. et al. (2020). *CodeBLEU: a Method for Automatic Evaluation of Code Synthesis*. DOI: 10.48550/arXiv.2009.10297.

Haque, M. M. et al. (2022). *FixEval: Execution-based Evaluation of Program Fixes for Competitive Programming Problems*. DOI: 10.48550/arXiv.2206.07796.

Genero, M. et al. (2005). *A Survey of Metrics for UML Class Diagrams*.DOI: 10.5381/jot.2005.4.9.a1.

Liu, Y. et al. (2014). *Two-phase Automated Software Measure Approach - From Class Diagram Design to Object-Oriented Metrics*. DOI: 10.2174/1874110X01408010029

Gandhi, S. et al. (2024). *Translation of Low-Resource COBOL to Logically Correct and Readable Java leveraging High-Resource Java Refinement*. DOI: 10.1145/3643795.3648388

3.

Definizione del framework per la Legacy modernization

(attività 5.3)

Legacy Modernization WPs

Reverse Engineering JAVA

- **Approccio AI-aided MDRE**
- **Input:** Code Base (Analisi statica) e Tracing del runtime (Analisi dinamica)
- **Strumenti:** tradizionali e Sistemi agentici AI
- **Output:** modelli UML, requisiti funzionali

Reverse Engineering COBOL

- **Approccio MDRE ?**
- **Input:** Code Base (Analisi statica) e Tracing del runtime (Analisi dinamica)
- **Strumenti:** tradizional e Sistemi agentici
- **Output:** modelli, requisiti funzionali

Database Modernization

- **DB Structure Extraction**
- **DB Structure Improvement**
- **Data Migration**
- **Business Logic migration** (estrazone della knowledge dalla Stored Procedure)

Coding

- **Input:** Requisiti funzionali
- **Strumenti:** Sistemi Agentici
- **Output:** Applicativo nel codice e nell'architettura target
- **Plus**: Codice privo d'errori e di elevata qualità

Validation & Testing

- **Input**: Analisi Requisiti, Use Case Diagram
- **Strumenti**: Sistemi agentici
- **Output**: Test cases, test Execution

IDE

Legacy Modernization Platform

Reverse Engineering – dallo stato dell'arte alla definizione del framework

Principali evidenze dello stato dell'arte:

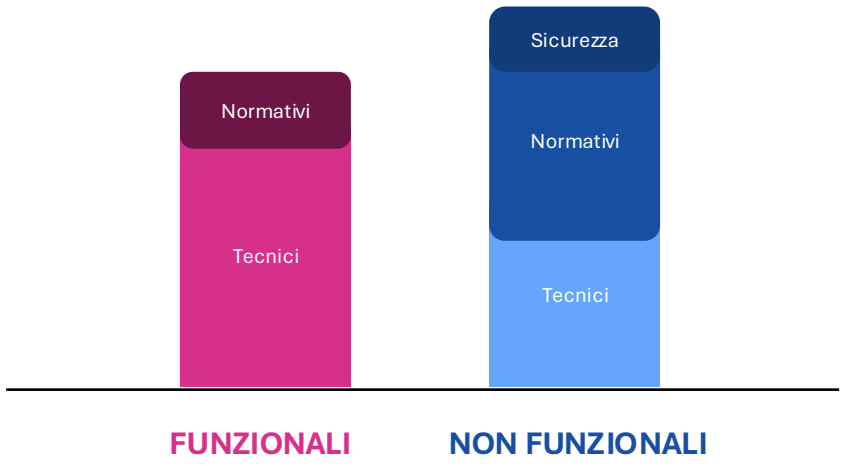
- approcci differenti e frammentati
- **MDRE** (Model Driving Reverse Engineering) sembra l'approccio più ampiamente adottato
- i risultati della fase di reverse engineering secondo il MDRE sono diagrammi UML, sia «structural» che «behavioural»
- molti autori hanno proposto metodologie e/o strumenti (es. FastUML [Fauzi], AgileUML [Lano] spesso non sufficientemente documentati e quasi sempre privi di riferimenti a repository
- i risultati basati sull'uso degli LLM sono ancora lontani dall'essere maturi, perché centrati sulla traduzione diretta o, in caso di approccio MDRE, verificati su snippet di codice

Con l'obiettivo di **estrarre i requisiti del software legacy**, le attività di studio e sperimentazione sono finalizzate a definire una metodologia che **adotta il principio MDRE** e si basa, secondo convenienza, su **strumenti ed approcci tradizionali** e su **tecniche assistite dall'AI**

Reverse Engineering

Reverse Engineering - Estrazione ed Elicitazione dei Requisiti, *obiettivi*

Condurre un'approfondita **retro-analisi** per l'estrazione dei **REQUISITI** funzionali e non funzionali dai sistemi legacy da modernizzare, attraverso: **analisi della eventuale documentazione e del codice sorgente, anche AI-Assisted**



REQUISITI TECNICI

- Analisi della documentazione AI-Assisted
- Analisi del codice sorgente (approccio ibrido)
- Approccio MDRE
- Tool tradizionali ed AI Agentic

REQUISITI NORMATIVI

- Analisi della normativa corrente con tool AI-Agentic
- Analisi del codice sorgente AI-Assisted

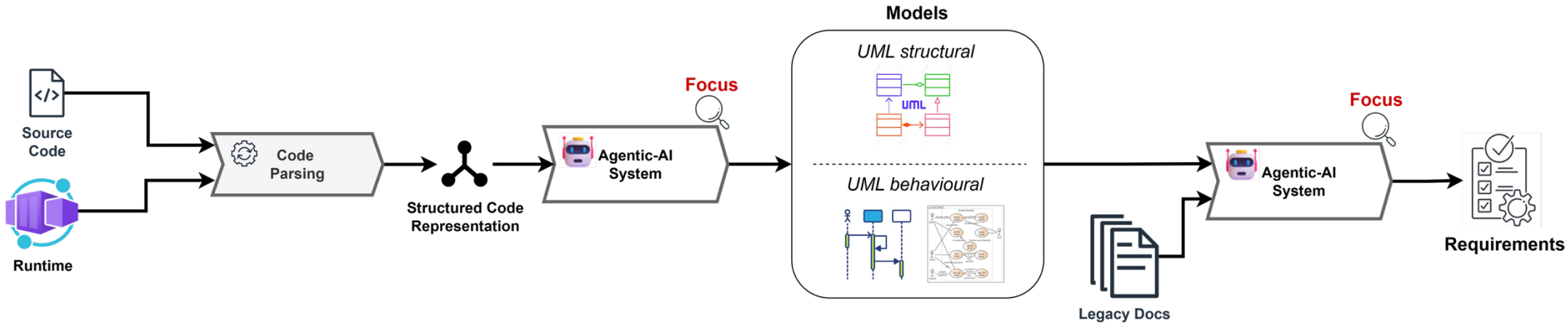
REQUISITI DI SICUREZZA

- Analisi dei sistemi e dei moduli utilizzati per i processi di autenticazione ed autorizzazione
- Analisi dello politiche per l'integrità dei dati
- Analisi delle strategie e delle infrastrutture per garantire la disponibilità dei dati

Rev. Eng. - Estrazione dei Requisiti TECNICI: Approccio AI-aided MDRE 1/2

L'obiettivo è applicare tecniche di **MDRE** tradizionali supportate da **strumenti AI** per estrarre automaticamente conoscenza strutturale, comportamentale e funzionale a partire da un repository software. Le principali intuizioni dietro a questo approccio:

- Dalla **codebase**:
  1. si estrae dal codice una rappresentazione strutturata e con informazioni topologiche attraverso strumenti tradizionali
  2. si estraggono i modelli strutturali (ad esempio, *class diagram*) che descrivono l'architettura e le dipendenze tra elementi
  3. si ricostruiscono **staticamente** i modelli comportamentali (ad esempio, *sequence diagram*, *use case diagram*) utilizzando chiamate, flussi di controllo e pattern ricorrenti nel codice
- Se è disponibile l'ambiente di **runtime**, dal tracing di esecuzione si integrano le interazioni estratte **dinamicamente** con i modelli comportamentali ottenuti staticamente per avere una vista più fedele dei flussi reali
- A partire dall'insieme di questi modelli e dall'eventuale documentazione esistente del sistema legacy, si procede infine a estrarre i **requisiti funzionali**, redatti secondo un template specifico definito all'interno del progetto, collegandoli in modo tracciabile a classi, scenari di esecuzione e casi d'uso identificati

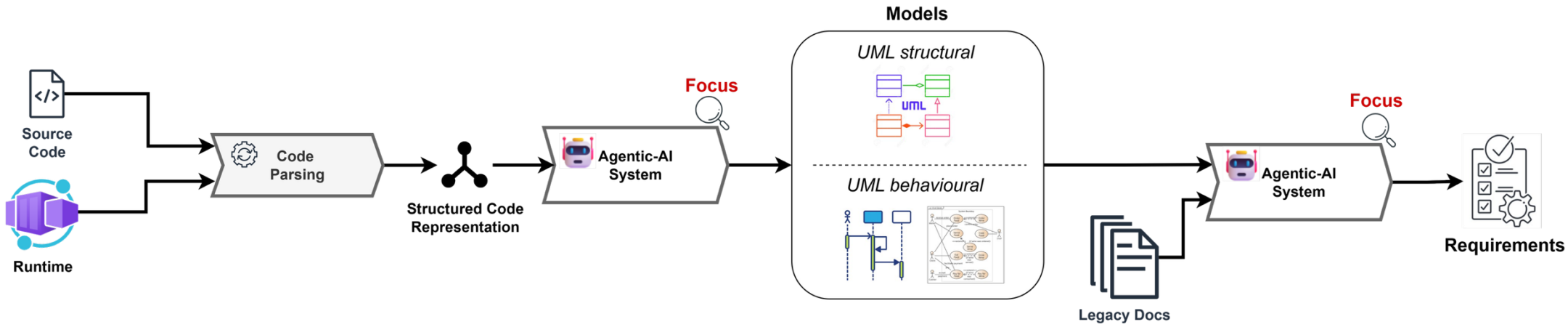


Rev. Eng. - Estrazione dei Requisiti TECNICI: Approccio AI-aided MDRE 2/2

Sistema agentico basato su AI come motore semantico per estrarre automaticamente i modelli dalla codebase e dai trace raccolti a runtime. Il sistema agentico riceve in input il codice oppure una sua meta-rappresentazione (ad esempio AST, grafi di chiamata, etc.) e genera in modo automatico *(i)* modelli strutturali e comportamentali e *(ii)* requisiti funzionali, dopo aver analizzato la documentazione legacy disponibile.

Alcune considerazioni sui sistemi agentici:

- L'impiego di un **singolo LLM** come soluzione end-to-end non è efficace, come mostrato dalla letteratura recente e dalle linee guida dei principali provider, perché tende a generare risultati poco controllabili, non riproducibili e difficili da validare.
- Si preferisce quindi un **approccio multi-agent**, in cui il problema complessivo viene scomposto in micro-task specializzati, ciascuno responsabile di produrre specifici output intermedi (ad esempio estrazione struttura, analisi comportamento, sintesi requisiti). Questo consente di *(i)* aumentare la trasparenza rispetto a soluzioni black-box; *(ii)* migliorare il controllo del flusso di esecuzione, definito a design-time tramite workflow espliciti; *(iii)* abilitare verifiche e validazioni intermedie tra i diversi micro-task, riducendo errori e allucinazioni lungo la catena di elaborazione.

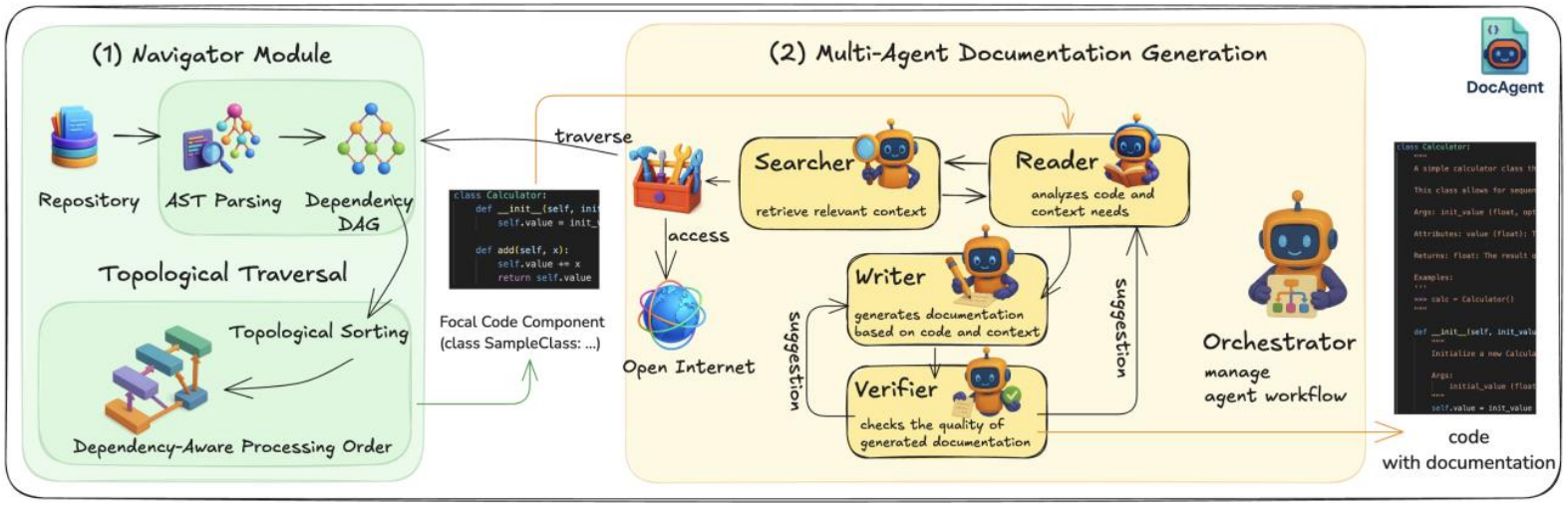


Inspired by the Agentic-AI State-of-the-Art Methods 1/2

DocAgent: A Multi-Agent System for Automated Code Documentation Generation

DocAgent usa una vista topologica del repository per **documentare** il codice in modo coerente e accurato, elaborando le entità nell'ordine dettato dalle loro dipendenze.

- Il sistema analizza l'intero repository, costruendo un grafo delle dipendenze tra funzioni, metodi, classi e moduli.
- Questo grafo viene reso aciclico (collassando eventuali cicli) e poi ordinato topologicamente, in modo che ogni componente venga documentato solo dopo che le sue dipendenze sono state già analizzate e descritte.
- Un insieme di agenti coordinati (lettore, ricercatore di contesto, scrittore, verificatore) passa nodo per nodo sul grafo, usando solo il contesto locale rilevante per generare docstring accurate e coerenti con il resto del sistema



Meta AI: https://arxiv.org/pdf/2504.08725

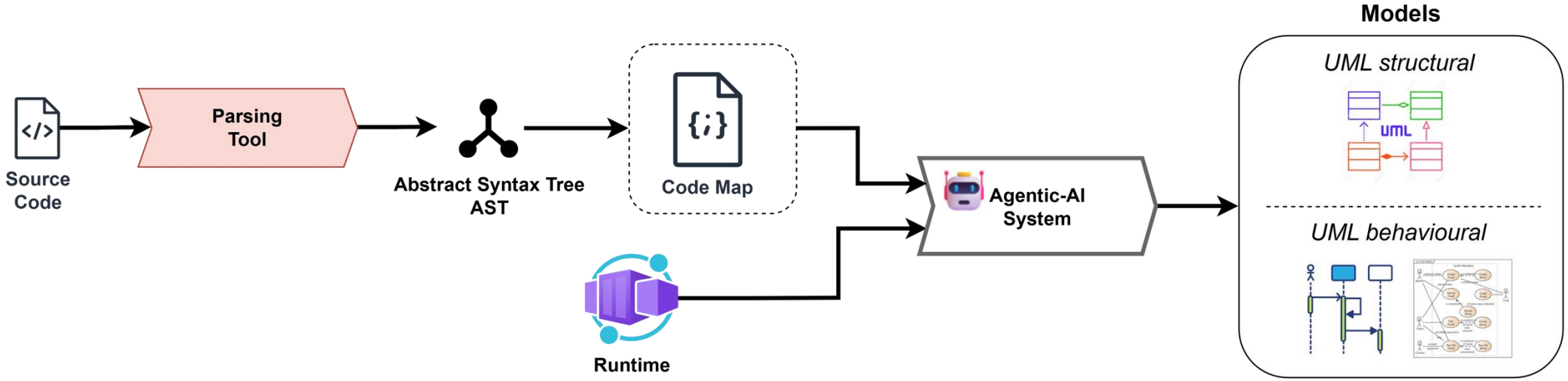
Inspired by the AI State-of-the-Art Methods 2/2

La metodologia *DocAgent* può essere applicata al Reverse Engineering?

Intuition: La documentazione può essere vista come un livello di astrazione del codice, esattamente come i diagrammi UML. Invece di passare l'intero repository agli agenti come testo grezzo, lo riconvertiamo in una rappresentazione strutturata, arricchita con informazioni topologiche, che permette ai modelli di linguaggio di cogliere in modo più efficace il contenuto informativo.

In scenari di **reverse engineering,** lo stesso grafo topologico può fungere da "meta-rappresentazione" stabile del sistema, su cui si possono costruire riassunti gerarchici, attori, casi d'uso e poi diagrammi, mantenendo un forte vincolo di aderenza al codice reale.

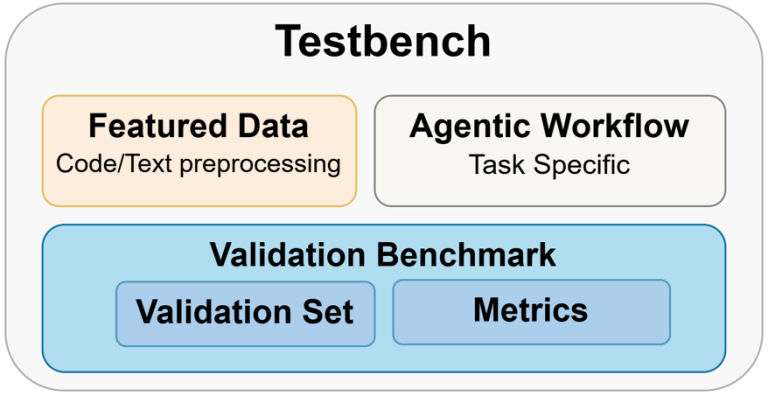
- La struttura topologica fornisce un "ordine naturale" di elaborazione segmentando i componenti software da fornire agli LLM, evitando di sovraccaricare il modello con l'intero repository a ogni passo.
- Questo riduce la lunghezza del contesto, migliora la coerenza tra le varie descrizioni e aumenta la probabilità che ogni diagramma generato sia ancorato a entità realmente presenti nel grafo del codice



AI-aided MDRE – Testbench platform

Piattaforma per supportare attività di esplorazione e sperimentazione. Principali componenti:

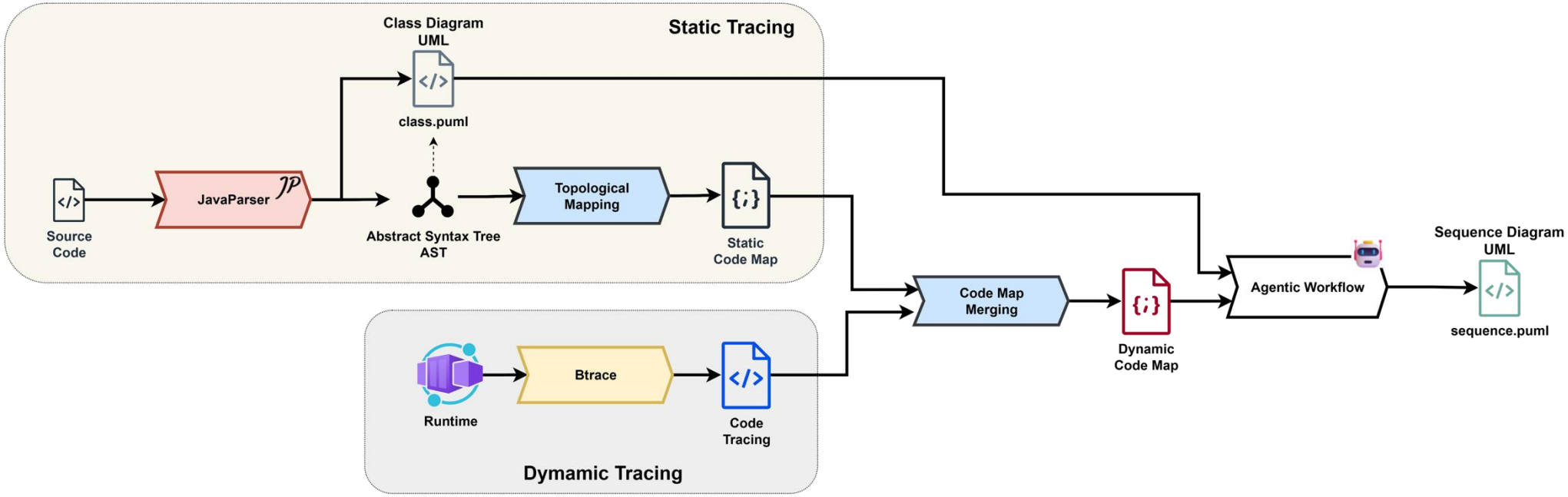
- **Featured Data**: suite di strumenti per la trasformazione dei dati. Include la conversione del codice in AST e in rappresentazioni grafiche (graph-based), oltre all'estrazione di contenuti testuali dalla documentazione (ad esempio PDF, file Word e altri formati), rendendoli disponibili come testo strutturato per l'analisi e la modellazione.
- **Agentic Workflow**: suite di tools per il design e orchestrazione di sistemi multi-agents, ognuno configurato per un task specifico.
  - Configurazione modulare via grafo: nodi rappresentano agenti o sotto-sistemi gerarchici annidati, per architetture scalabili e riusabili.
  - Principio "Agents as a Feature": incapsula il sistema come oggetto riutilizzabile, integrabile tramite file YAML con build automatico.
  - Basato su *LangGraph* (Python): *StateGraph* per stato centrale (transizioni, checkpoints), *runtime Pregel* per orchestrazione.



- **Validation benchmark**: sistema per la valutazione quantitativa dei risultati generati. Richiede un validation set con input e output attesi, oltre a metriche specifiche e pertinenti al task agentico, per misurare accuratezza, robustezza e qualità complessiva delle soluzioni prodotte.

AI-aided MDRE con Java – UML Sequence Diagrams 1/3

- **Static tracing**: a partire dalla codebase si costruiscono gli AST e il class diagram tramite JavaParser, insieme a un grafo del codice (code map statica) che rappresenta chiamate, dipendenze e relazioni tra le entità.
- **Dynamic tracing**: partendo dall'applicazione in esecuzione (runtime), un tracer registra il flusso reale delle chiamate e produce un tracciato dinamico del codice.
- La code map statica e il tracing dinamico vengono integrati per ottenere una code map dinamica, che combina struttura del sistema e comportamento osservato durante l'esecuzione.
- **Sequence Diagram Extraction Agentic Workflow**: a partire da code map dinamica e class diagram, un workflow multi-agente estrae il *sequence diagram*.
  - *Top-Down*: prima si genera il contesto macro (la sequenza principale ad alto livello), poi si arricchisce progressivamente con i dettagli dei singoli componenti interni.
  - *Bottom-Up*: si adotta un flusso tipo *MapReduce*; prima si generano le sequenze locali per ciascun elemento, poi queste vengono integrate gerarchicamente con i componenti padre fino a comporre l'intero sequence diagram.

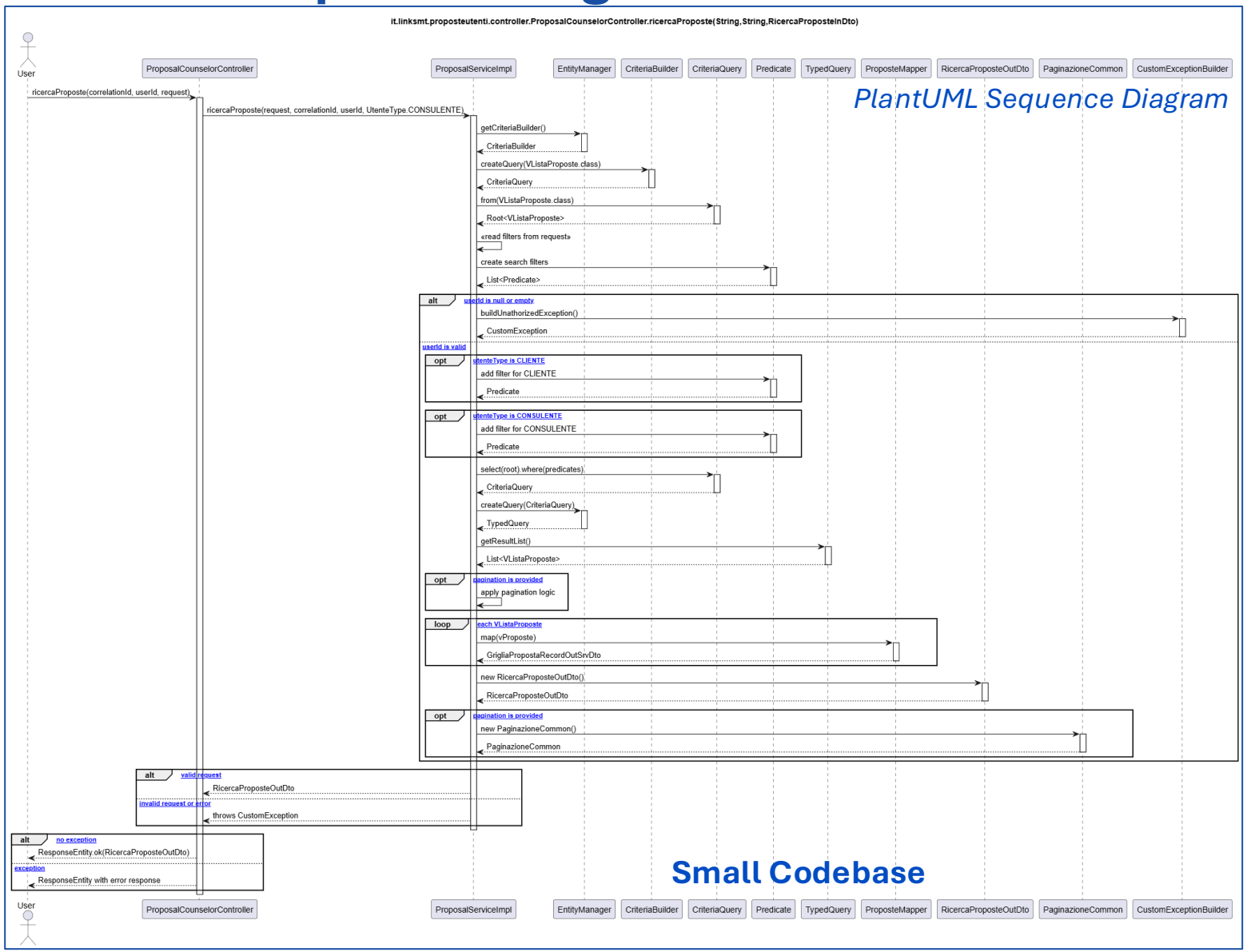


AI-aided MDRE con Java – UML Sequence Diagrams 2/3

- **Esempio**: **Content Search** application

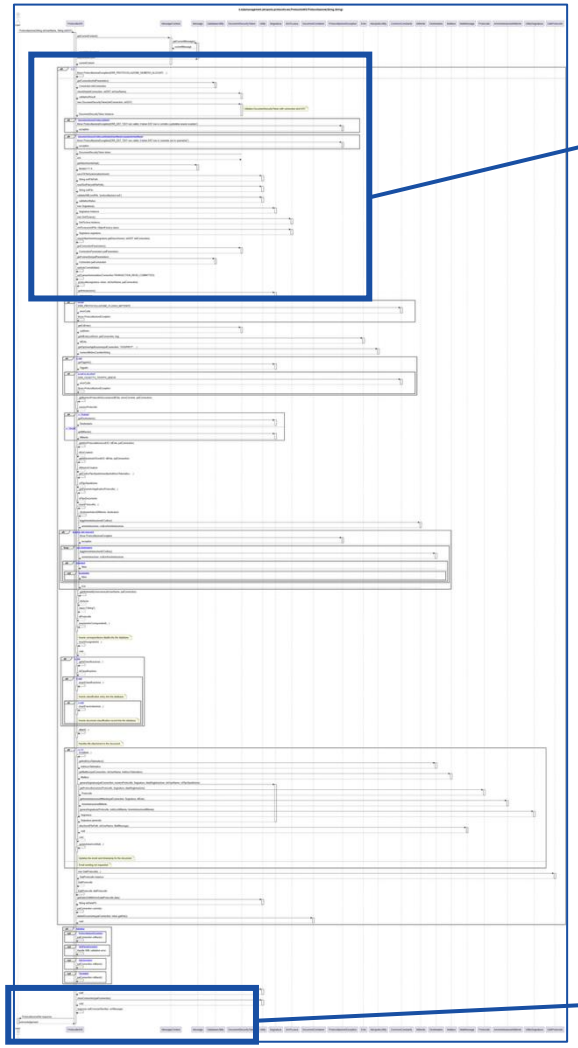

```
{
  "id": "it.linksmt.proposteutenti.controller.ProposalCounselorController.ricercaProposte(Strin
  "called_by": "USER",
  "description": "Handles the POST endpoint `/ricerca-proposte` for counselors. It receives fil
  "code": "\t@PostMapping(\"/ricerca-proposte\")\n\t@Operation(operationId = \"ricercaProposteC
  "calls": [
    {
      "id": "it.linksmt.proposteutenti.service.impl.ProposalServiceImpl.ricercaProposte(Ricerca
      "called_by": "it.linksmt.proposteutenti.service.ProposalService.ricercaProposte(RicercaPr
      "description": "Executes the proposal search logic. It validates the user, builds dynamic
      "code": "    @Override\n    public RicercaProposteOutDto ricercaProposte(RicercaProposteI
      "calls": [
        {
          "id": "it.linksmt.proposteutenti.exception.CustomExceptionBuilder.buildUnathorizedExc
          "called_by": "it.linksmt.proposteutenti.service.impl.ProposalServiceImpl.ricercaPropo
          "description": "Creates and returns a `CustomException` representing an HTTP 401 Unau
          "code": "\tpublic CustomException buildUnathorizedException() throws CustomException
          "calls": []
        },
        {
          "id": "it.linksmt.proposteutenti.mapper.ProposteMapper.map(String)",
          "called_by": "it.linksmt.proposteutenti.service.impl.ProposalServiceImpl.ricercaPropo
          "description": "Maps a string value representing a document type to an enum `Document
          "code": "\t@ValueMapping(target = \"PRODUCT\", source = \"PRODUCT\")\n\t@ValueMapping
          "calls": []
        },
        {
          "id": "it.linksmt.proposteutenti.dto.RicercaProposteOutDto.RicercaProposteOutDto()",
          "called_by": "it.linksmt.proposteutenti.service.impl.ProposalServiceImpl.ricercaPropo
          "description": "DTO constructor used to instantiate an output object containing the r
          "code": "@Builder\n@Data",
          "calls": []
        },
        {
          "id": "it.linksmt.proposteutenti.dto.PaginazioneCommon.PaginazioneCommon()",
          "called_by": "it.linksmt.proposteutenti.service.impl.ProposalServiceImpl.ricercaPropo
          "description": "DTO constructor for pagination metadata, storing total pages, total e
          "code": "@Data\n@Builder",
          "calls": []
        }
      ]
    }
  ]
}
```


Code Map (Static)



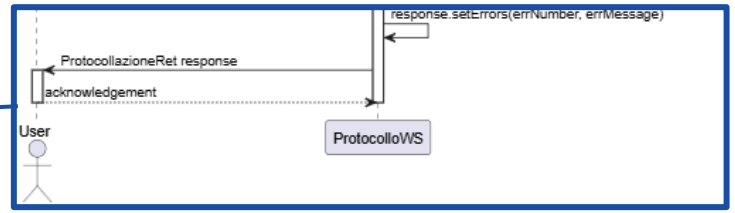
AI-aided MDRE con Java – UML Sequence Diagrams 2/2

- **Esempio**: **Protocol** application



Scalability issue in Large Codebase

Too detailed, almost code

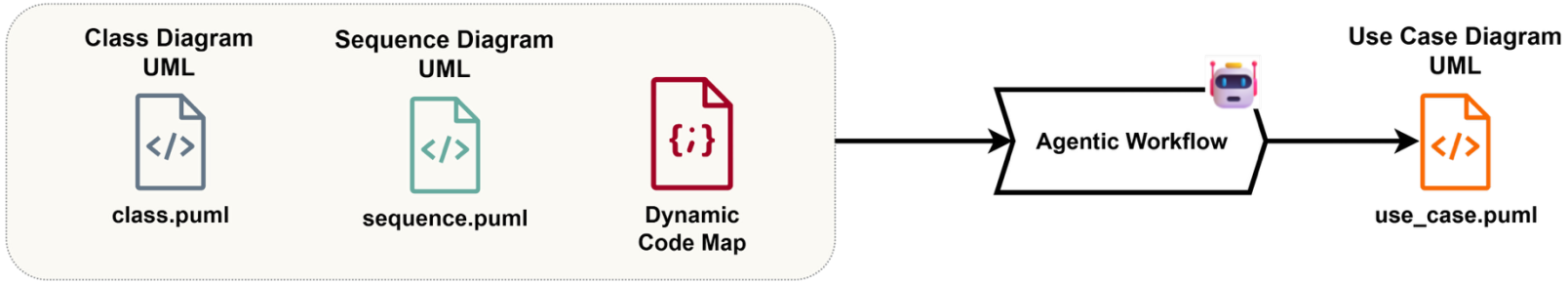


Syntax Error + Hallucination

AI-aided MDRE con Java – UML Use Case Diagrams 1/2

- **Use Case Diagram Extraction Agentic Workflow:** a partire dagli UML class e sequence diagrams e dalla code map dinamica, un workflow multi-agente ricava lo use case diagram, seguendo un ordine preciso di estrazione:

  - ***Attori***: attori di business escludendo componenti interni del sistema.
  - ***Obiettivi***: individuare i principali obiettivi di business associati al flusso considerato, che costituiranno la base per definire i singoli casi d'uso.
  - ***Use cases***:
    - creare casi d'uso solo per processi di business realmente distinti;
    - ignorare i passaggi puramente tecnici (accesso al database, mapping, DTO, logging, gestione di eccezioni tecniche);
    - utilizzare sempre terminologia di business e denominazioni verbo+sostantivo (es. "Gestire ordine");
    - le relazioni «include» sono riservate a sotto-processi di business riusabili, le «extend» a varianti di business rilevanti, mentre la generalizzazione tra use case è usata solo se il modello la richiede in modo esplicito.

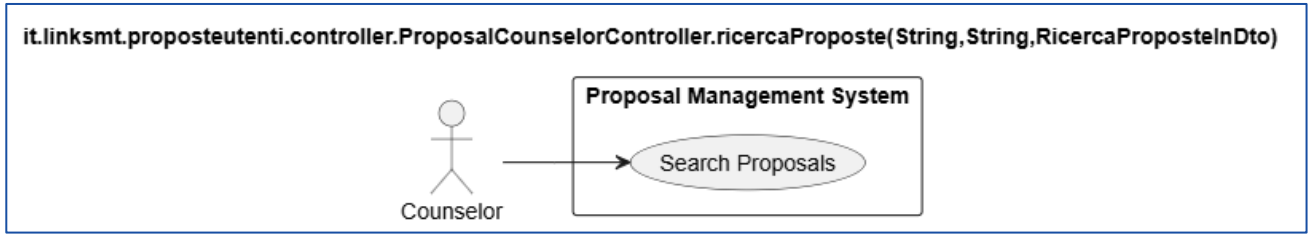


AI-aided MDRE con Java – UML Use Case Diagrams 2/2

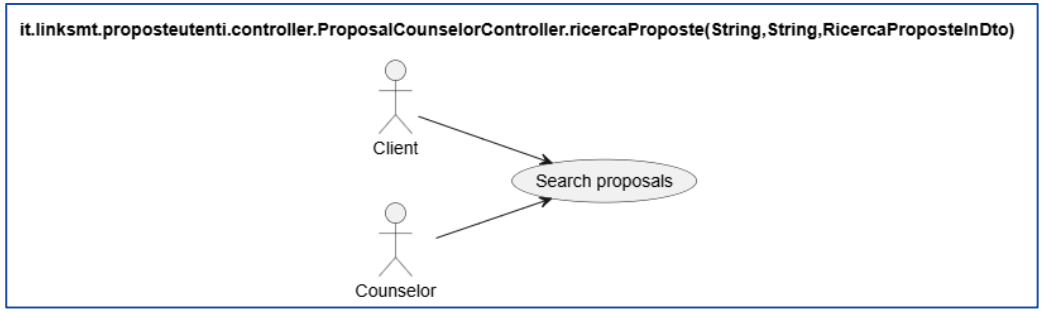
- **Esempio**: **Content Search** application

Issue: Non-determinism

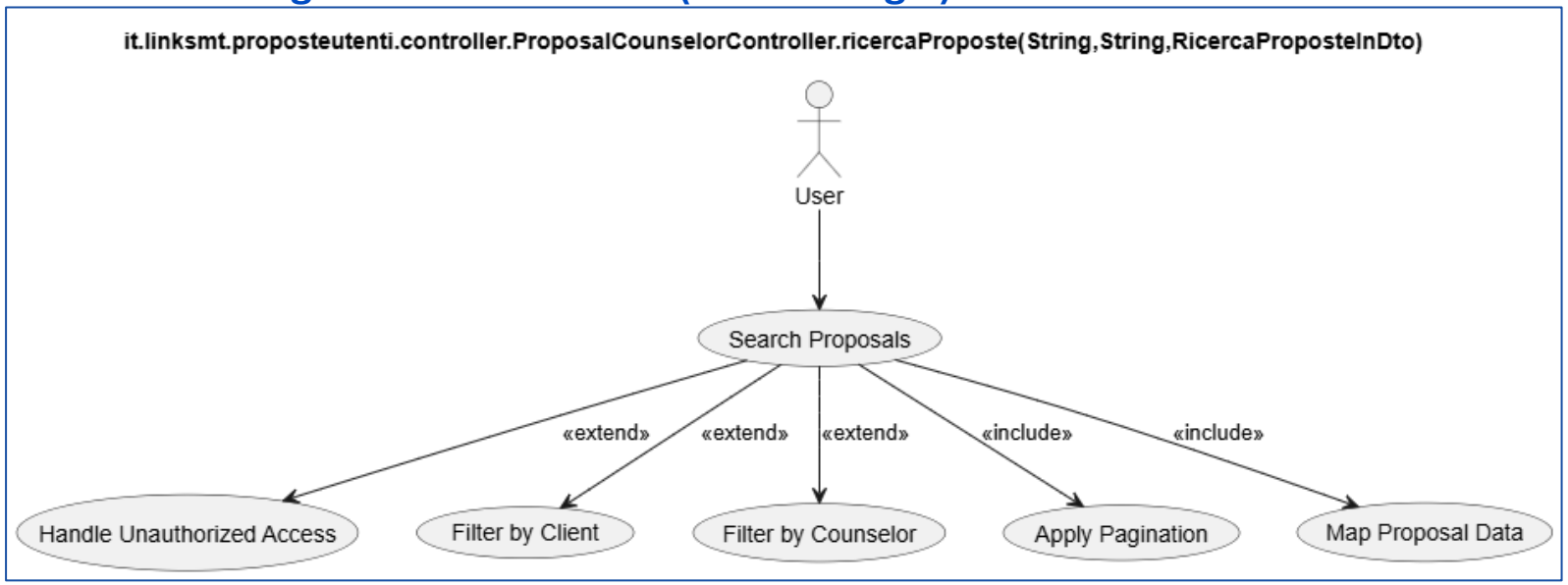
Use Cases Diagram 1: Only Context



Use Cases Diagram 2: Context + hallucinations



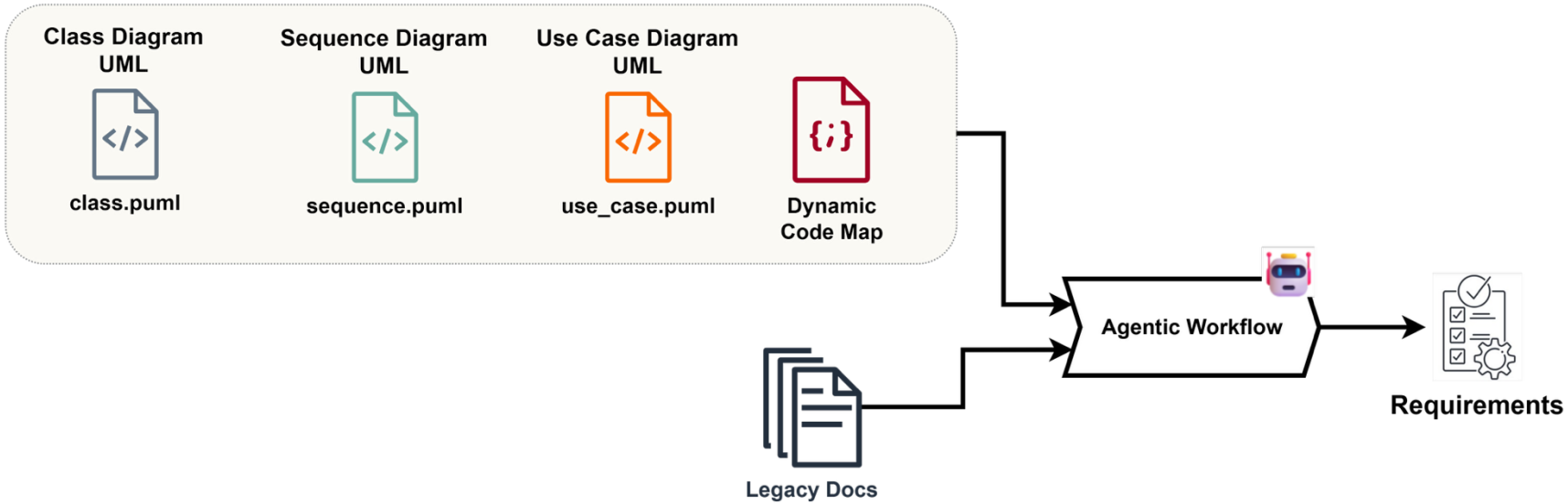
Use Cases Diagram 3: Too detailed (internal logic)



AI-aided MDRE con Java – Requirements 1/2

- **Requirements Extraction Agentic Workflow:** a partire dai diagrammi UML, dalla code map dinamica e dai documenti legacy di progetto, un workflow multi-agente ricava i requisiti funzionali del repository. I requisiti seguono un template standard che include:

  - *Descrizione del Servizio:* una pagina con scopo, attori, funzionalità principali, ciclo di vita delle entità e vincoli tecnici.
  - *Casi d'Uso*: elenco organizzato per funzionalità del servizio, con intestazione standardizzata.
  - *Dettaglio dei Casi d'Uso*: attori, pre/post-condizioni, scenario base numerato a step di 10, scenari alternativi e note tecniche.
  - *Dati Coinvolti*: filtri di ricerca, azioni disponibili, risultati/output, dati di input/configurazione con regole e vincoli.



AI-aided MDRE con Java – Requirements

- **Esempio**: **Content Search** application

Issue: Error propagation (es, wrong use cases → wrong requirements)

SERVIZIO: Ricerca Proposte 1/3

Descrizione del Servizio
Il servizio "Ricerca Proposte" consente ai consulenti di effettuare ricerche mirate sulle proposte disponibili nel sistema. Lo scopo principale è fornire un'interfaccia per filtrare e visualizzare le proposte in base a criteri specifici, garantendo un'esperienza utente ottimale e una gestione efficiente dei dati.

****Attori coinvolti:****
- ****Consulente:**** L'utente principale del sistema, responsabile della ricerca e gestione delle proposte.

****Funzionalità principali offerte:****
- Ricerca di proposte basata su criteri specifici (es. utente, tipo di proposta, ecc.).
- Applicazione di filtri personalizzati per restringere i risultati.
- Supporto alla paginazione per la gestione di grandi volumi di dati.
- Mappatura e trasformazione dei dati per la visualizzazione.

****Stati/ciclo di vita delle entità gestite:****
- ****Proposta:**** Può essere filtrata, visualizzata e restituita come risultato della ricerca.
- ****Filtri di ricerca:**** Configurati dinamicamente in base ai parametri forniti dall'utente.

****Note tecniche e vincoli importanti:****
- Il sistema utilizza un approccio basato su criteri per costruire query dinamiche.
- La gestione degli errori è centralizzata e prevede la generazione di eccezioni personalizzate in caso di input non valido o mancanza di autorizzazioni.
- La paginazione è opzionale e viene applicata solo se specificata nella richiesta.

2. CASI D'USO 2/3

1.1 Ricerca Proposte - Consulente

1.1.1 Attori, Pre-condizioni, Post-condizioni

****Attore:**** Consulente

****Descrizione:**** Il consulente effettua una ricerca di proposte utilizzando criteri specifici.

****Pre-condizioni:****
- L'utente deve essere autenticato e autorizzato come "Consulente".
- Devono essere forniti i parametri di ricerca (es. ID utente, tipo di utente, ecc.).
- Il sistema deve essere operativo e connesso al database.

****Post-condizioni:****
- Viene restituita una lista di proposte che soddisfano i criteri di ricerca.
- Se specificata, viene applicata la paginazione ai risultati.
- In caso di errore, viene restituita una risposta con un messaggio di errore appropriato.

1.1.2 Scenari

****Scenario Base****

Step	Descrizione
10	Il consulente invia una richiesta di ricerca con i parametri specificati.
20	Il sistema valida i parametri di input.
30	Il sistema costruisce i criteri di ricerca in base ai parametri forniti.
40	Il sistema esegue la query sul database per ottenere i risultati.
50	Il sistema applica la paginazione, se specificata.
60	Il sistema mappa i risultati in un formato leggibile per il consulente.
70	Il sistema restituisce i risultati al consulente.

****Scenari Alternativi**** 3/3

Step Base	Step Alternativo	Descrizione
20	A.20	Se i parametri di input non sono validi, il sistema genera un'eccezione.
40	A.40	Se non ci sono risultati, il sistema restituisce una lista vuota.
50	A.50	Se la paginazione non è specificata, il sistema restituisce tutti i dati.

1.1.3 Dati

****Filtri di Ricerca:****
- ****correlationId**** (Obbligatorio): Identificativo univoco della richiesta.
- ****userId**** (Obbligatorio): Identificativo dell'utente che effettua la ricerca.
- ****utenteType**** (Obbligatorio): Tipo di utente (es. CONSULENTE, CLIENTE).
- ****Criteri di ricerca**** (Facoltativi): Filtri specifici per restringere i risultati.

****Azioni Disponibili:****
- ****Ricerca Proposte:**** Esegue la ricerca in base ai criteri forniti.
- ****Paginazione:**** Applica la paginazione ai risultati.

****Risultati/Output:****
- ****Lista di Proposte:**** Contiene le proposte che soddisfano i criteri di ricerca.
  - ****Formato:**** Oggetto `GrigliaPropostaRecordOutSrvDto`.
  - ****Regole di ordinamento:**** Ordinamento predefinito o specificato nei criteri.

****Dati di Input/Configurazione:****
- ****correlationId**** (Obbligatorio): Stringa univoca per tracciare la richiesta.
- ****userId**** (Obbligatorio): Identificativo dell'utente.
- ****utenteType**** (Obbligatorio): Tipo di utente (es. CONSULENTE, CLIENTE).
- ****Filtri di ricerca**** (Facoltativi): Parametri per personalizzare la ricerca.
  - ****Vincoli:**** Devono essere validi e coerenti con il modello dati.

AI-aided MDRE con Java – Punti aperti

Tracing (Static & Dynamic)

- **Copertura incompleta del comportamento**: alcuni percorsi di esecuzione, condizioni limite o scenari alternativi possono non essere riflessi nei diagrammi, soprattutto se il tracing o la raccolta di contesto non è esaustiva.
- **Dipendenza forte dalla qualità delle meta-rappresentazioni:** AST, trace dinamici, grafi di chiamata e mappe del codice devono essere accurati e aggiornati; errori o omissioni in queste strutture si propagano direttamente nei diagrammi UML e nei requisiti

Agentic-AI

- **Non-determinismo dei risultati**: i diagrammi e il documento di requisiti possono cambiare sensibilmente tra una esecuzione e l'altra, anche a parità di input e configurazione del sistema agentico.
- **Debolezza nella causalità:** è difficile imporre nel prompting una logica rigorosa di causa-effetto tra chiamate di metodi, eventi, trigger e relazioni tra casi d'uso; il modello tende a sfruttare correlazioni semantiche anziché relazioni causali verificate.
- **Sensibilità al prompting**: per questa tipologia di task (UML) variazioni significative nei prompt, nella configurazione degli agenti, non impattano in modo significativo sulla struttura dei diagrammi e la formulazione dei requisiti.
- **Integrazione imperfetta tra diversi livelli di astrazione**: passare da dettagli di basso livello (sequenze di chiamate, metodi) a concetti di alto livello (use case, obiettivi di business, requisiti funzionali) è intrinsecamente complesso e soggetto a interpretazioni ambigue; difficoltà a descrivere nel prompt il livello giusto di dettaglio.
- **Scalabilità limitata su grandi codebase**: per sistemi di dimensioni reali, la quantità di contesto necessario (class diagram esteso, code map dinamica, log di esecuzione, documentazione) potrebbe superare la finestra di contesto effettivamente "attendibile" del modello. Anche quando la finestra massima è teoricamente sufficiente in termini di token, il modello tende a prestare maggiore attenzione alle parti iniziali e finali del contesto in input (*lost in the middle*), degradando la rilevanza delle porzioni centrali e causando perdita di dettagli importanti sui flussi di esecuzione e sulle relazioni tra componenti.
- **Allucinazioni strutturali e semantiche**: il sistema può introdurre attori, use case, messaggi o passi di sequenza che non esistono nel codice o nei log, ma che "sembrano plausibili" dal punto di vista linguistico.
- **Difficoltà di controllo fine sugli standard UML**: garantire che la sintassi e la semantica UML (per sequence e use case diagrams) siano rispettate in modo consistente richiede regole aggiuntive, validatori o post-processing non banali.

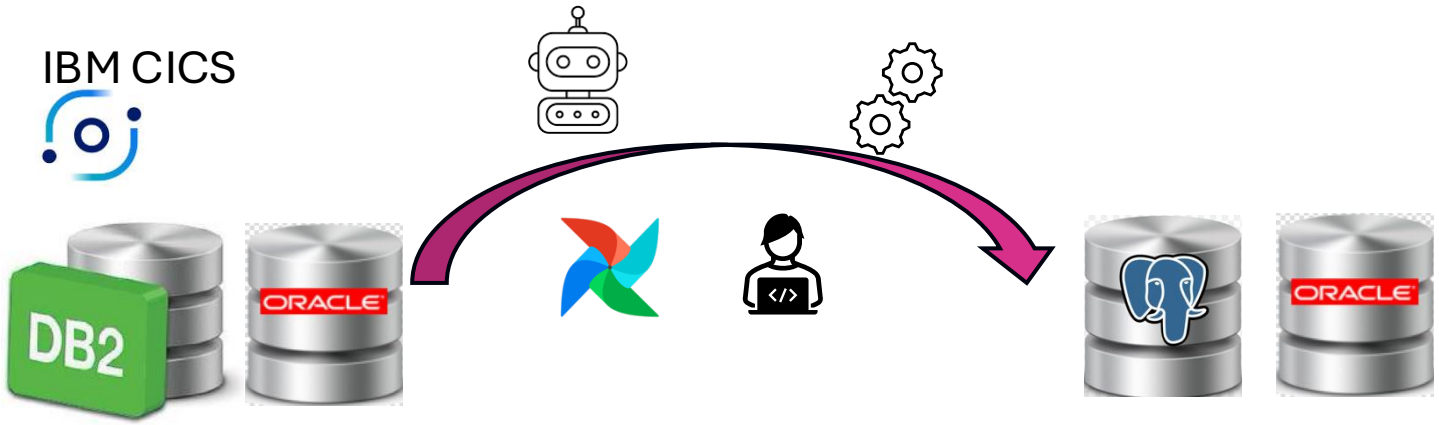
Validation

- **Validazione e benchmarking complessi:** definire metriche affidabili per valutare correttezza, completezza e utilità dei diagrammi e del documento di requisiti richiede dataset di riferimento, gold standard UML e criteri di giudizio spesso costosi da costruire.

Modernizzazione delle Basi Dati

Obiettivi

- **Reverse Engineering** degli schemi delle basi dati legacy
- **Raffinamento** degli schemi dei DB **assistita** da **sistemi agentici** che analizzano i diagrammi ER estratti, individuano potenziali inefficienze e propongono soluzioni, **validate** comunque da **esperti progettisti** di Database (human in the loop)
- **Migrazione** dei dati realizzata con **ETL** le cui routine di trasformazione sono realizzate con **sistemi agentici**

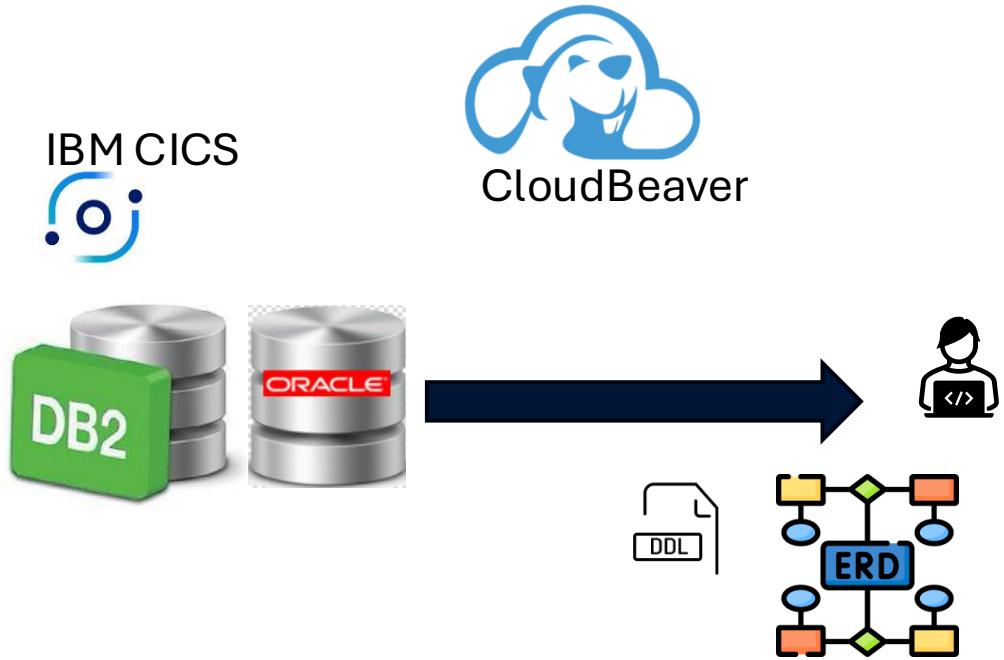


Modernizzazione delle Basi Dati

Attività in corso: *selezione di uno strumento open source*

Per il **Reverse Engineering** degli schemi delle basi dati si è scelto uno strumento «tradizionale», perché ritenuto più efficace di uno strumento agentico (causa: indeterminismo)

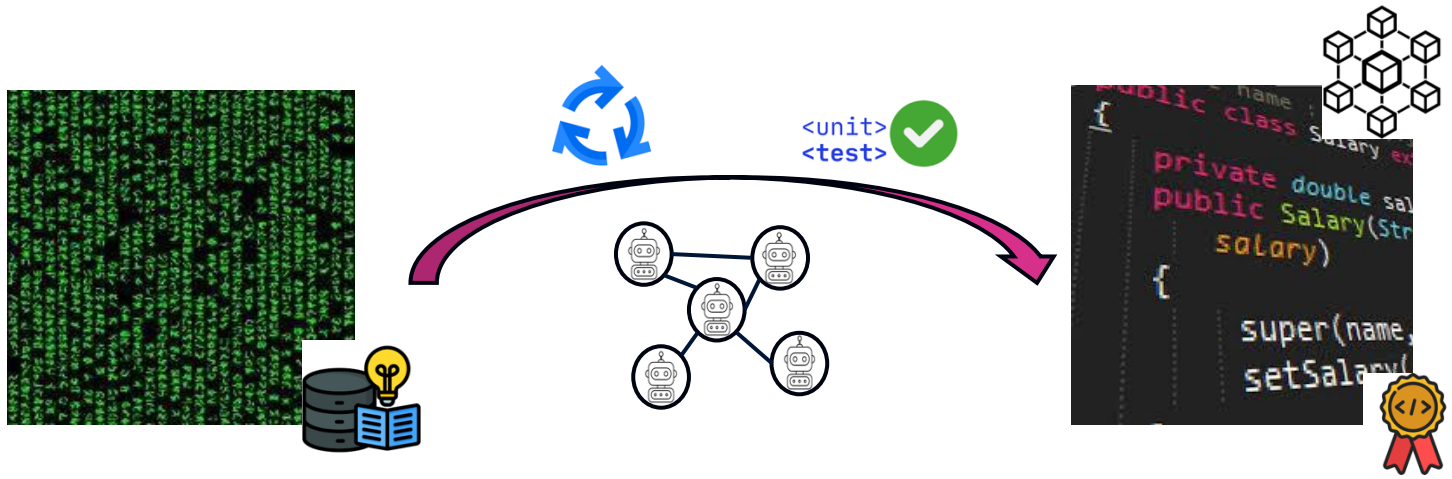
La scelta è ricaduta sulla soluzione open source **CloudBeaver** (versione cloud di **Dbeaver**) che supporta tutti i driver dei più popolari RDBMS, schema editor, strumenti di interrogazione dei DB (ecc.)



Implementazione del Software Trasformato

Obiettivi

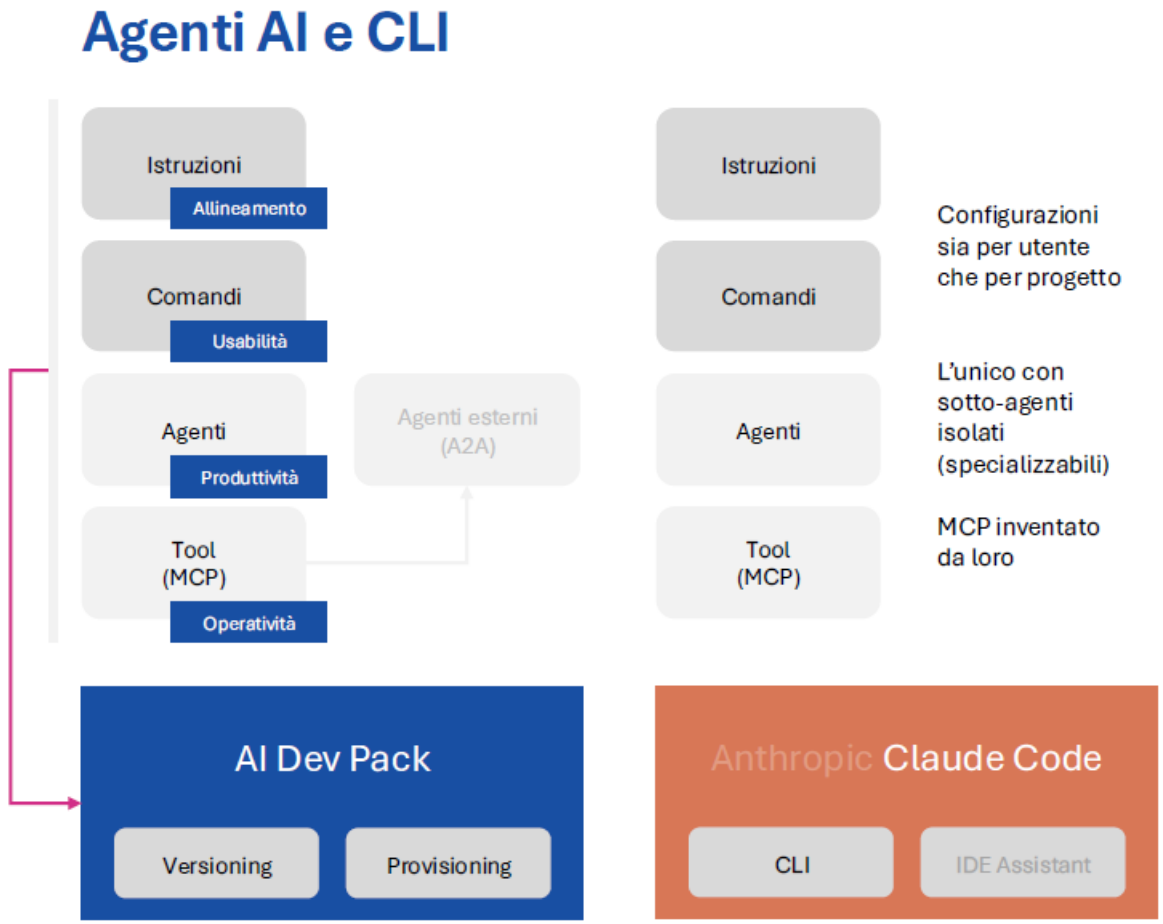
- La **Knowledge Base** dei requisiti **Funzionali** e **Tecnici** viene elaborata da un **sistema agentico** per produrre il codice modernizzato
- La trasformazione viene effettuata in maniera **incrementale** ed **iterativa**, effettuando di volta in volta la misurazione delle performance qualitative del codice rispetto a **benchmark**
- Il codice trasformato supera severi test di **Code Quality Assurance**, grazie all'utilizzo di sistemi allo stato dell'arte



Implementazione del Software Trasformato

Attività in corso: *sperimentazione strumenti commerciali*

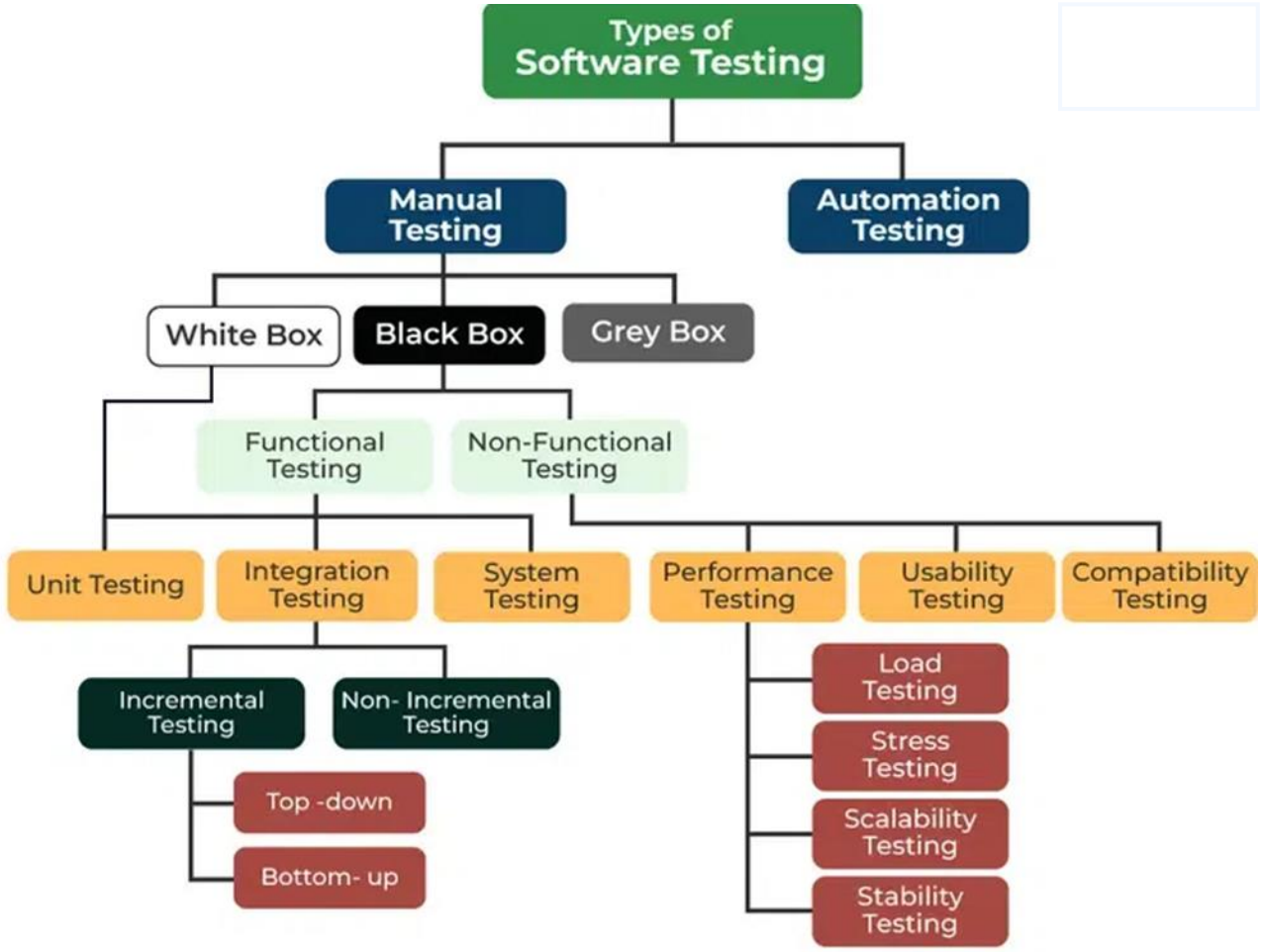
- Scouting degli strumenti agentici allo stato dell'arte (Codex, Copilot, Claude Code, ecc)
- Definizione di buone pratiche per l'uso di questi strumenti allo scopo di assistere gli sviluppatori
- Realizzazione di un DevPack come acceleratore per i team di sviluppo



Validation & Testing

Obiettivi

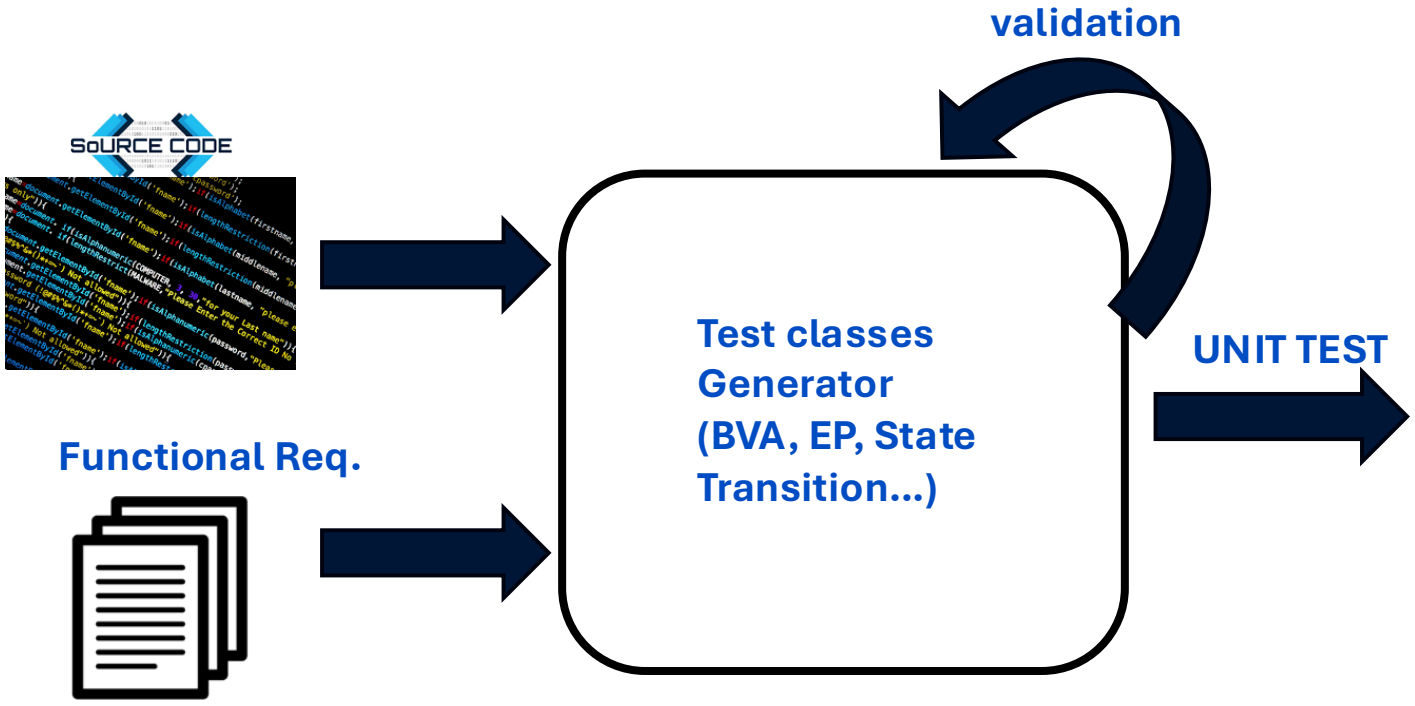
- **Test planning framework**:
  - implementazione dei piani di test a partire dai requisiti funzionali e non funzionali
  - piani relativi a tutte le tipologie di test allo stato dell'arte (Unit test, API Test, Integration Test, Component Test, System Test)
- **Test Automation**: esecuzione dei piani di test
- **Confronto con le prestazioni del legacy**



Validation & Testing

Attività in corso: *Unit Test*

- **Unit Test framework**: partendo dal codice e dai requisiti funzionali, per ciascun metodo vengono individuate le tipologie di test applicabili (BVA, EP, State Transition, etc). Per ciascuna di queste tipologie, vengono generati i valori da testare; a valle di una fase di validazione, si generano poi gli Unit Test
- **Sistema Agentico**, sia per l'analisi del codice che per l'analisi dei requisiti
- **Misurazione** della qualità dei test generati (con metriche classiche come precisione, recall, ecc.)



Integrazione con il motore di compliance normativa

Obiettivi

- **Step 1- Monitoraggio Normativo**: assistente digitale basato su AI generativa per monitorare il quadro normativo in continua evoluzione
- **Step 2 – Analisi di impatto normativo** – Analisi dell'impatto di una nuova norma sull'assetto regolamentare/procedurale interno
- **Step 3 – Realizzazione di un workflow agentico** per orchestrare i processi bancari secondo normativa.
- **Step 4 – Motore di compliance** del Software trasformato

Talent Drives Innovation

Grazie per l’attenzione

Links group | Links | Linfa | LOS | Nexus Advanced Technologies | idee75 | dli | codd&date