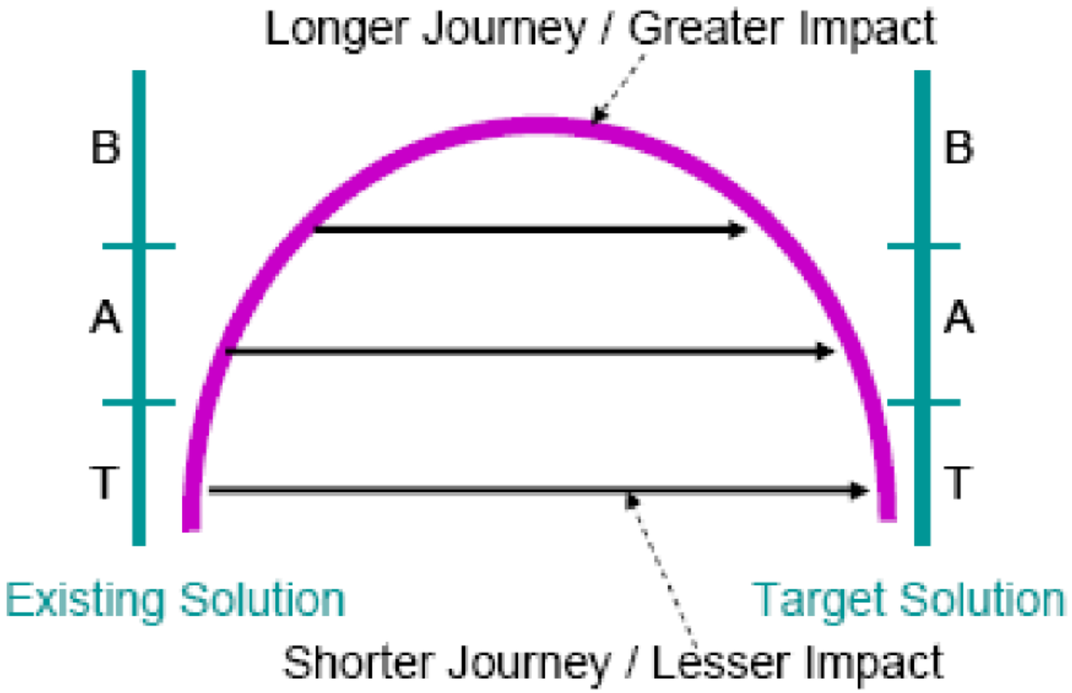
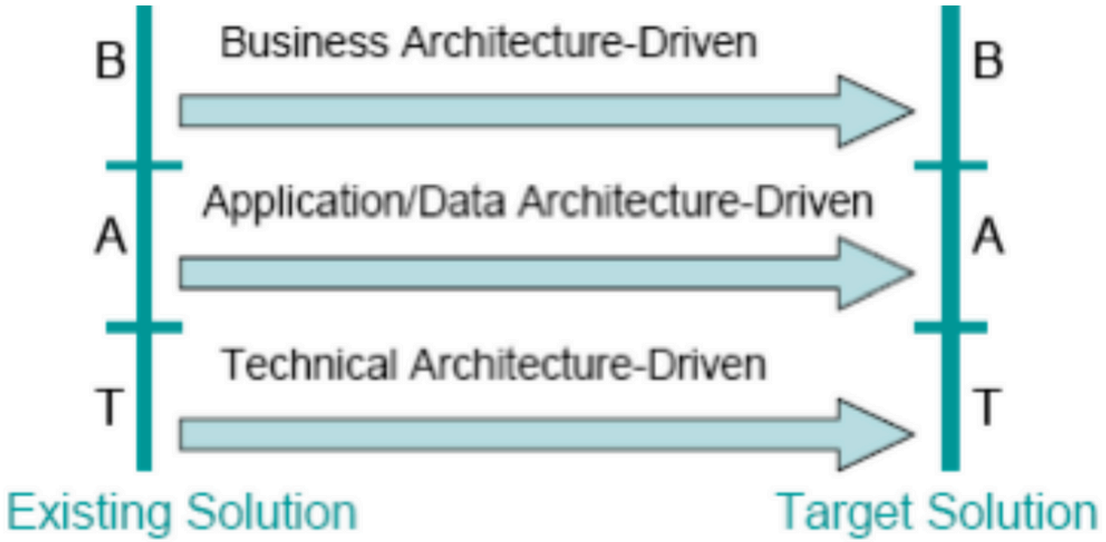
andrea.andrenucci@studenti.unisalento.it
andrea.elia3@studenti.unisalento.it

MIGRAZIONE DI SISTEMI LEGACY

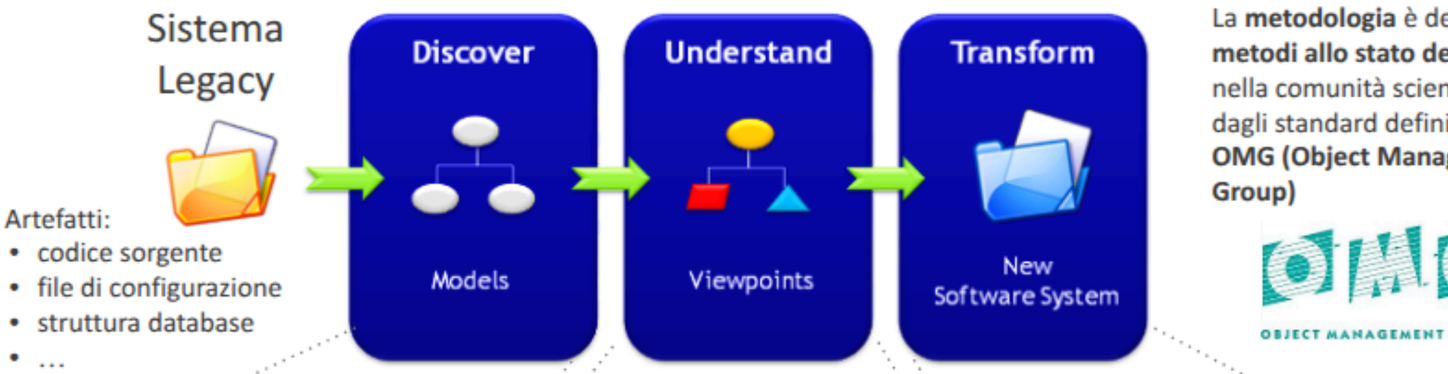
Stato dell’arte

LE TRAIETTORIE DI MODERNIZZAZIONE



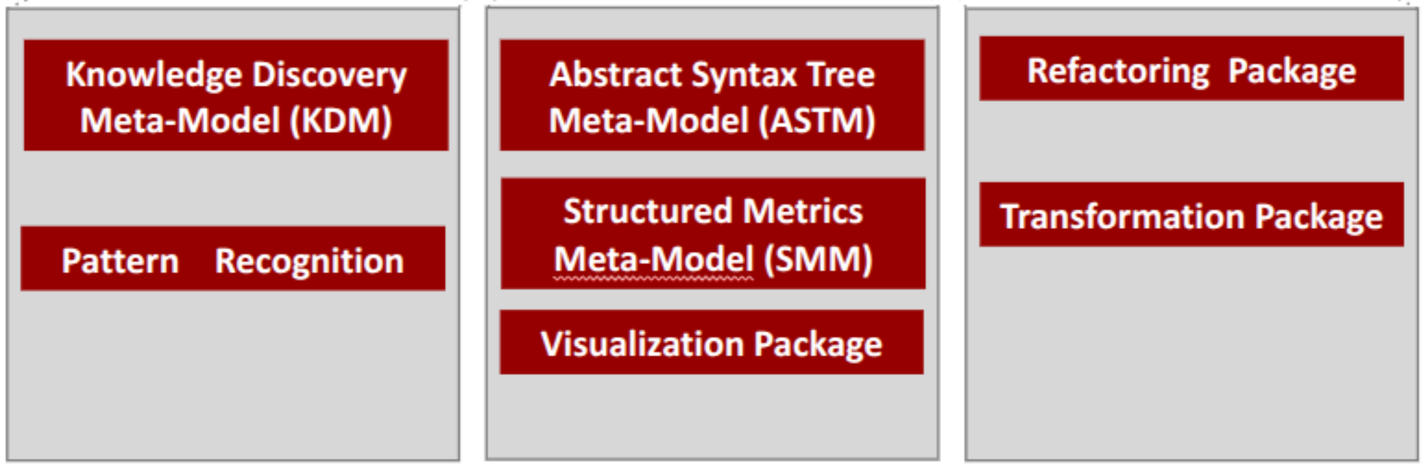


I PASSI DELLA MODERNIZZAZIONE MODEL-DRIVEN



La **metodologia** è derivata da **metodi allo stato dell'arte** nella comunità scientifica e dagli standard definiti dall' **OMG (Object Management Group)**

OMG
OBJECT MANAGEMENT GROUP



STATO DELL'ARTE

COME GIÀ NOTO, IL PROCESSO DI MIGRAZIONE DI UN SISTEMA LEGACY È UN TASK COMPLESSO. LA DIFFICOLTÀ DEL PROBLEMA RISIEDE NELLA NATURA STESSA DEL CODICE:

- MANCANZA DI DOCUMENTAZIONE
- POSSIBILE PRESENZA DI CODICE DI BASSA QUALITÀ
- POSSIBILI DISSERVIZI IN FASE DI MIGRAZIONE

IN QUESTO LAVORO PRELIMINARE, IL FOCUS È STATO QUELLO DI TROVARE SISTEMI (BASATI SU AI E NON) CHE POTESSERO AFFRONTARE QUESTE DIFFICOLTÀ

GLI ARTICOLI TROVATI SONO STATI DIVISI IN BASE AL PUNTO CRITICO CHE AFFRONTANO:

- Approcci architetturali
- Sistemi per la scomposizione in microservizi
- Tool di supporto alla migrazione
- Traduzione del codice via LLM
- Tool automatici basati su AI
- Testing automatico del codice

IN AGGIUNTA, SONO STATI RICERCATI DEI SISTEMI CHE AFFRONTANO IN MANIERA ALTERNATIVA UNA SERIE DI PROBLEMI SPECIFICI

- SISTEMI PER LA RAPPRESENTAZIONE INTERMEDIA
- BENCHMARK E VALIDAZIONE

APPROCCIO AGILE-ASSISTED

CHATTERJEE ET. AL. HANNO PRESENTATO UNA SUITE DI STRUMENTI BASATI SU INTEL PIN (INSTRUMENTATION DINAMICA) IN SUPPORTO AD UNA STRATEGIA AGILE PER LA MIGRAZIONE DI CODICE LEGACY

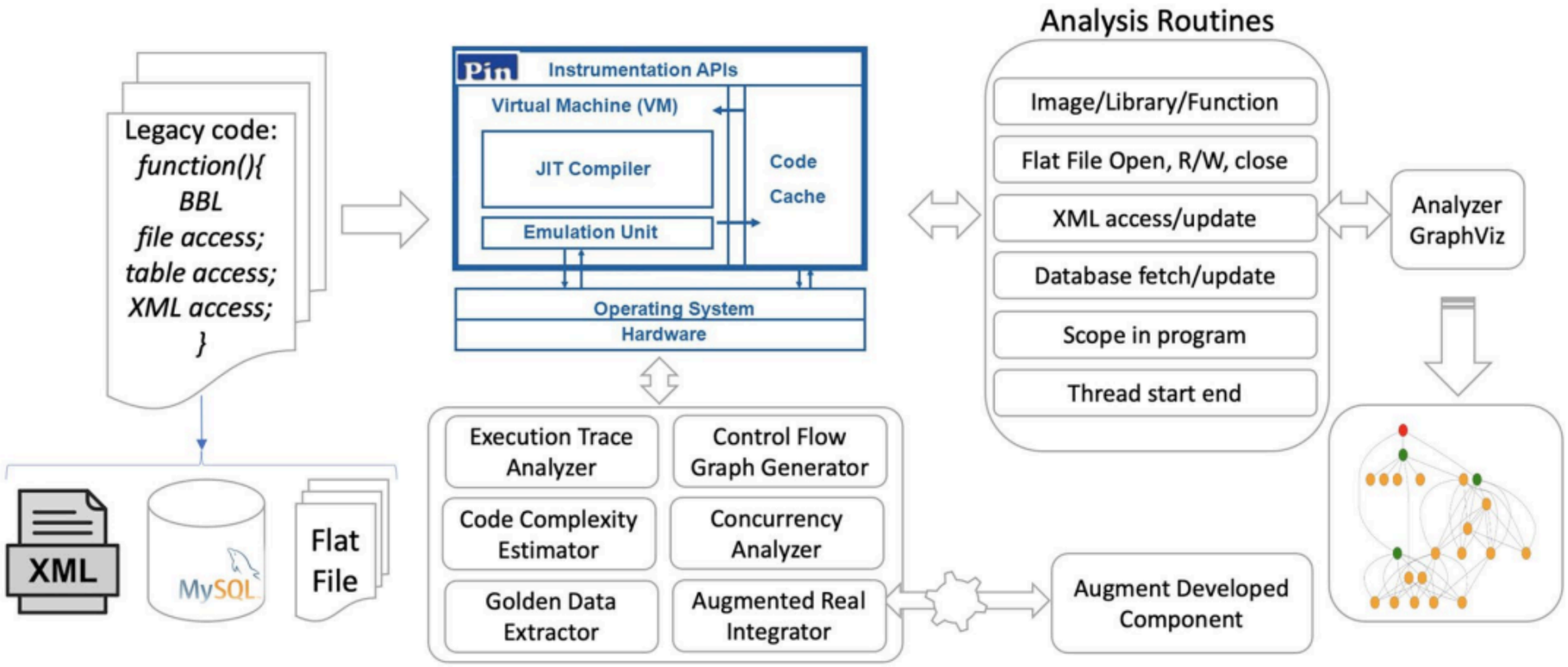
È STATO MISURATO UN'INCREMENTO DEL 30% ALLA SPRINT VELOCITY DOVUTO ALL'INTRODUZIONE DELLA SUITE

Strumento	Tipo di Instrumentation	Funzionalità principali	Output generato
Trace Analyzer	Function entry/exit, image-level instrumentation	- Traccia sequenze di chiamate di funzione- Raccoglie frequenze, percorsi, parametri- Identifica moduli attivi	File **.DOT** con grafo delle chiamate e frequenze
CFG Generator	Function entry/exit, basic blocks, I/O functions	- Genera Control Flow Graph- Rileva accessi a file, XML, DB- Ricostruisce dipendenze funzionali	File **.DOT** con grafo del flusso + IO access
Concurrency Analyzer	Thread-level instrumentation (pthread, Boost), synchronization events	- Traccia thread, lock, wait- Identifica deadlock/race- Costruisce modelli di concorrenza	**Thread model .DOT** (modello di concorrenza)
Code Complexity Estimator	Function entry/exit, basic blocks, instruction-level	- Calcolo Halstead metrics- Calcolo complessità ciclomatica- Misura LOC e complessità runtime	CSV / DOT con **metriche di complessità**
Golden Data Extractor	Function entry/exit, file/XML/DB read-write	- Estrae input-output reali (dati "golden")- Identifica funzioni pure/impure- Crea dataset per TDD	File **JSON** con test data e sequenze di transazione
Augmented Real System Integrator	Runtime image/function instrumentation (PIN RTN_Replace)	- Sostituzione dinamica di funzioni- Integrazione tra legacy e moduli moderni- Supporto cross-language (es. JNI)	**Esecuzione live** del modulo integrato (nessun file statico)

INOLTRE MEDIANTE LA SUITE SONO STATI ESTRATTI CON SUCCESSO, NEI VARI ESPERIMENTI ESEGUITI, I GOLDEN DATAS NECESSARI AI TEST

I golden data (o golden test data) sono insiemi di input-output reali estratti automaticamente dall'esecuzione del sistema legacy tramite instrumentation.

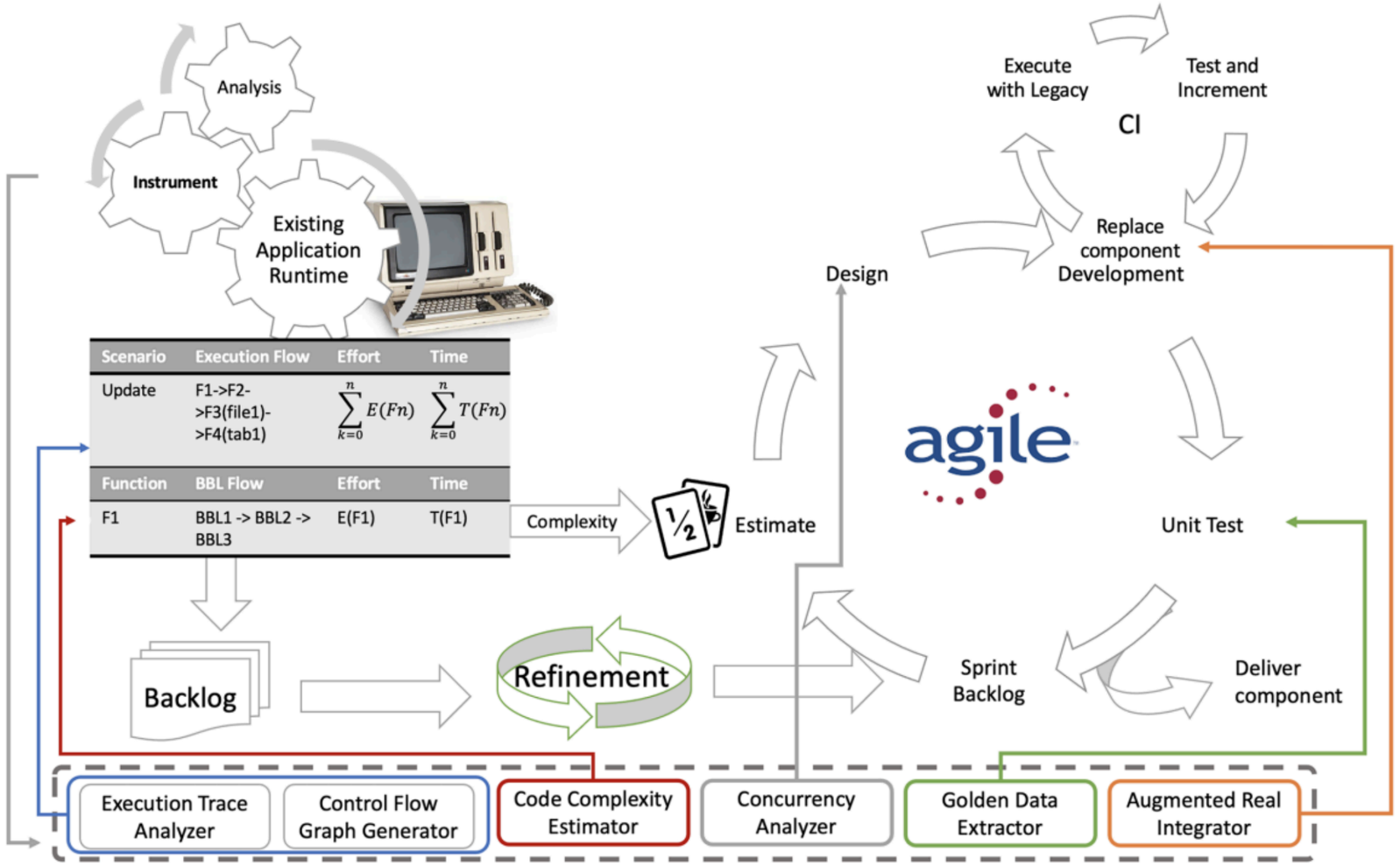
Servono come riferimento ("gold standard") per verificare che il nuovo sistema modernizzato riproduca esattamente il comportamento originale



L’APPROCCIO È STATO TESTATO SU
UNA VARIETÀ AMPIA DI LINGUAGGI:

- FORTRAN
- PASCAL
- COBOL
- C/C++

CON RISULTATI POSITIVI,
CHE CONFERMANO LA
RIUSCITA DELLO STUDIO



SCOMPOSIZIONE AUTOMATICA IN MICROSERVIZI

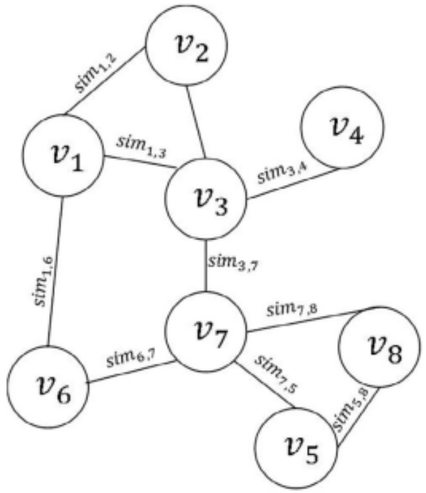
Lo screening ha identificato due modi, data un’architettura monolitica, di passare ad un’architettura a microservizi

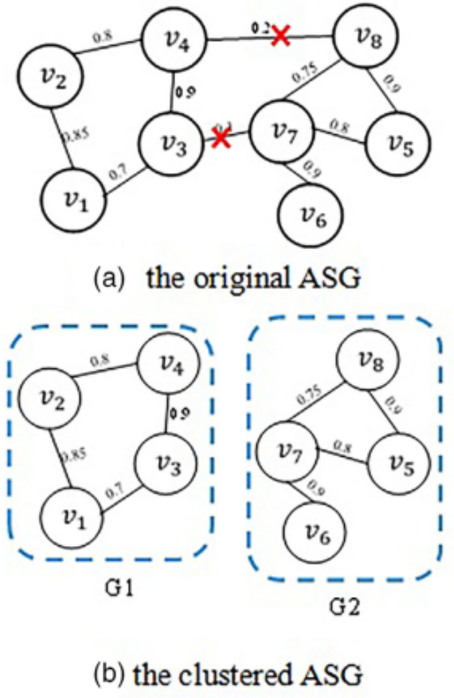
API similarity graph (Xiaoxiao Sun): modella il problema come una ricerca su grafo, usa per l’analisi unicamente gli standard OpenAPI e Swagger. Mediante un peso α si controlla quanto il sistema usa la similiarità di topic e dei messaggi di risposta del candidato.
Ottiene ottimi risultati (2 PMS, 11 GMS, 2 BMS)

Il sistema effettua:

- **Calcolo delle similiarità fra API**
- **Costruzione del grafo**
- **Spectral Clustering**

La scelta del numero di microservizi viene poi effettuata tramite K-means sullo spazio spettrale

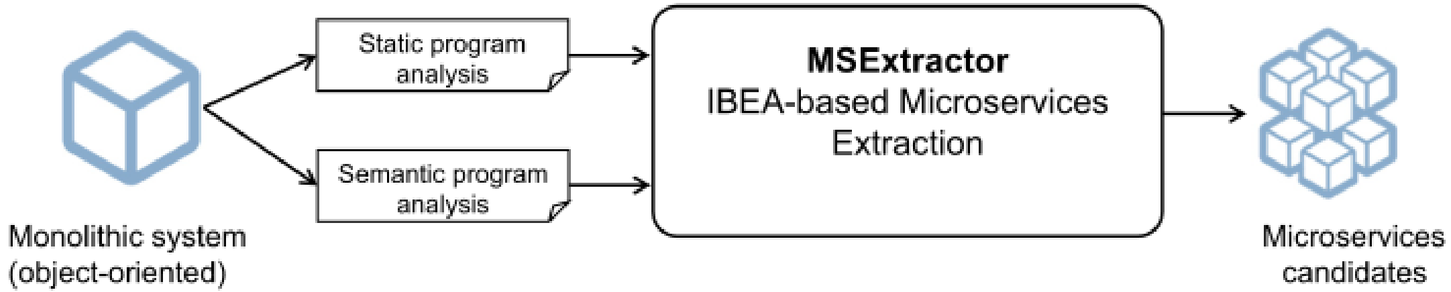




Evolutionary search (Khaled Sellami): ottimizza una funzione multi-obiettivo per identificare i microservizi.
Per quanto i risultati siano positivi (sempre sopra il 75%, e sul 90% in uno dei tre casi proposti), nell'articolo viene specificato che il piccolo numero di campioni presi a riferimento e il tipo di misure utilizzate potrebbero minare la validità del risultato.

L'obiettivo è identificare microservizi:

- coesi
- debolmente accoppiati
- con granularità adeguata



Basata su:

- chiamate di metodo tra classi
- direzione e frequenza delle chiamate

Via algoritmo IBEA, che minimizza un ipervolume costruito sugli obiettivi

TOOL DI SUPPORTO ALLA MIGRAZIONE

BASATI SU

-MODELLI MACHINE LEARNING

-MODELLI DEEP LEARNING

XMAINFRAME

PRIMO LLM BASATO SU DEEPSEEK-CODER SPECIALIZZATO NELLA COMPRENSIONE DEL SISTEMA LEGACY COBOL

Sviluppato da FPT Software AI Center

I GAP DEGLI LLM ESISTENTI

- TRAINING DATA INSUFFICIENTE
- ASSENZA DI BENCHMARK PER MAINFRAME SPECIFICI
- COMPRENSIONE OLTRE LA GENERAZIONE

LA COMPRENSIONE DEL CODICE RAPPRESENTA IL 40-60% DI EFFORT

ARCHITETTURA PER LA SPECIALIZZAZIONE DEL DOMINIO

DEEPSEEK-CODER

- CONTEXT WINDOW 16K TOKEN(ROPE)

MAINFRAMEBENCH

- DOMANDE RISPOSTA
- SPIEGAZIONE DEL CODICE COBOL
- DOMANDE A RISPOSTA MULTIPLA

TRAINING

- **CONTINUAL PRE-TRAINING** :COBOL SOURCE CODE (GITHUB),OFFICIAL DOCUMENTATION, GENERAL DATA PER RAGIONAMENTO E LINGUAGGIO

- **INSTRUCTION TUNING** SU INSTRUCT-DATASET

BENCHMARK

MULTIPLE CHOICE QUESTION TASK

Model	Accuracy (%)
GPT-4	73,90%
DeepSeek-Coder-Instruct 6.7B	47,49%
DeepSeek-Coder-Instruct 33B	53,29%
XMAiNframe-Instruct 7B	**68,57%**
XMAiNframe-Instruct 10.5B	**77,89%**

QUESTION ASNWERING TASK

Models	MAP	F1-Score	BERTScore	RougeL	Meteor	BLEU-4
GPT 4	0,12	0,19	0,88	0,18	0,34	5,71
DeepSeek-Coder-Instruct 6.7B	0,09	0,15	0,86	0,14	0,30	4,09
DeepSeek-Coder-Instruct 33B	0,09	0,15	0,86	0,15	0,31	4,41
XMAiNframe-Instruct 7B	**0,45**	**0,42**	**0,92**	**0,40**	**0,42**	**20,43**
XMAiNframe-Instruct 10.5B	**0,43**	**0,42**	**0,92**	**0,40**	**0,42**	**20,93**

BENCHMARK

COBOL CODE SUMMARIZATION

Models	BERTScore	RougeL	Meteor	BLEU-4
GPT 4	0,85	0,22	0,34	7,42
DeepSeek-Coder-Instruct 6.7B	0,85	0,22	0,32	7,72
DeepSeek-Coder-Instruct 33B	0,85	0,21	0,31	7,55
XMAiNframe-Instruct 7B	**0,89**	**0,41**	**0,56**	**22,23**
XMAiNframe-Instruct 10.5B	**0,96**	**0,74**	**0,74**	**62,58**

COBOL Code Function

```
TIME-RTN.
      *    ***<<   時間処理    >>***
           DISPLAY 'ENTERTIME:'.

           ACCEPT  WK-INPUT-IME.
           UNSTRING WK-INPUT-TIME DELIMITED BY ':'
           INTO WK-HH, WK-MM, WK-SS.
      *    ***<<   時間の編集    >>***
           IF  WK-HH < 10
               MOVE '0' TO WK-HH-FORMATTED
               MOVE WK-HH TO WK-HH-FORMATTED(2:2).
           ELSE
               MOVE WK-HH TO WK-HH-FORMATTED.
           END-IF.
           IF  WK-MM < 10
               MOVE '0' TO WK-MM-FORMATTED
               MOVE WK-MM TO WK-MM-FORMATTED(2:2).
           ELSE
               MOVE WK-MM TO WK-MM-FORMATTED.
           END-IF.
           IF  WK-SS < 10
               MOVE '0' TO WK-SS-FORMATTED
               MOVE WK-SS TO WK-SS-FORMATTED(2:2).
           ELSE
               MOVE WK-SS TO WK-SS-FORMATTED.
           END-IF.
           STRING WK-HH-FORMATTED, WK-MM-FORMATTED,
                     WK-SS-FORMATTED

           DELIMITED BY ':' INTO WK-TIME-FORMATTED.
TIME-RTN-X.
```

Summarization

The provided Cobol code is designed to accept a time input from the user, and then edit and format the input time. The code accepts the time input, unstrings it into hours, minutes, and seconds, and then adds leading zeros to each part if necessary. Finally, it concatenates the formatted time parts using a colon delimiter.

COME FUNZIONA?

INPUT :

- Codice cobol

OUTPUT:

- Spiegazione della complessita
- Spiegazione del codice Cobol
- Refactoring Suggestion

USE CASE:

- Comprensione del codice
- Valutazione della migrazione
- Generazione di documentazione

MODELLI DISPONIBILI

BASE-7B	INSTRUCT-7B	INSTRUCT-10.5B
Modello grezzo su cui fare fine-tuning • Si ha un dataset interno all'azienda • Necessita' di adattare il modello per una determinata azienda	**ChatBot** • Deployment immediato in IDE • Tool di supporto per sviluppatori • Generazione automatica di documentazione tecnica	**Versione di instruct-7b con piu layers** • Refactoring suggestion • Analisi critica pre-migrazione • Accuracy piu' elevata • Da usare nel caso di capacita computazione disponibile

MIGRATION EXP

LEARNING-TO-RANK:
GESTIONE DELLA PRIORITÀ IN FASE DI MIGRAZIONE

-Decision Support System

-Trainato su migrazioni reali

Mateus, Martinez,Kolski - 2023

PERCHE' E' IMPORTANTE L'ORDINE DEI FILE DA MIGRARE?

In letteratura si distinguono due tipologie di migrazione

- one-step migration (Big Bang)
- incremental migration (Chicken Little)

Nella Chicken Little emerge un problema: In che ordine migro i file?

MIGRATIONEXP NASCE PER RISOLVERE ESATTAMENTE QUESTO PROBLEMA: DATO UN PROGETTO DA MIGRARE, SUGGERISCE QUALI FILE MIGRARE PRIMA BASANDOSI SU PATTERN APPRESI DA MIGRAZIONI REALI.

1.TRAINING PHASE

Input: Progetti open-source già migrati da Java a Kotlin

Processo di training :

- Estrazione commit con migrazioni
- Feature extraction (56 metriche per file)
- Pairwise learning-to-rank
- LambdaMART : Impara relazione tra features e probabilità di migrazione

2.SERVING PHASE

Input: Progetto P da migrare (mix di file Java e Kotlin)

Processo di Production:

- Estrae features dai file Java non ancora migrati
- Crea vettori di features per ogni file candidato
- Query al modello: passa tutti i vettori al ranking model

Output: lista ranked di file con "predicted relevance score"

USE CASE: SIMPLE CALENDAR PRO

VERSIONE COMMIT 2D1C59:

- 38 FILE KOTLIN (GIÀ MIGRATI)
  - 6 FILE JAVA CANDIDATI

Candidate files	Predicted relevance	Relevant document	Predicted ranking
AboutActivity.java	0.96	Yes	1
MyWidgetProvider.java	0.58	–	2
WidgetConfigureActivity.java	0.42	–	3
Utils.java	0.32	–	4
LicenseActivity.java	0.27	–	5
Constants.java	−0.24	–	6

TRADUZIONE AUTOMATICA DEL CODICE CON LLM

INTERTRANS

UTILIZZO DI LINGUAGGI INTERMEDI PER LA TRADUZIONE AUTOMATICA DI CODICE

Conference paper Marcos Macedo et al.

2024

PERCHÉ É RILEVANTE

Studi recenti (Lost in Translation - Pan 2024) dimostrano che

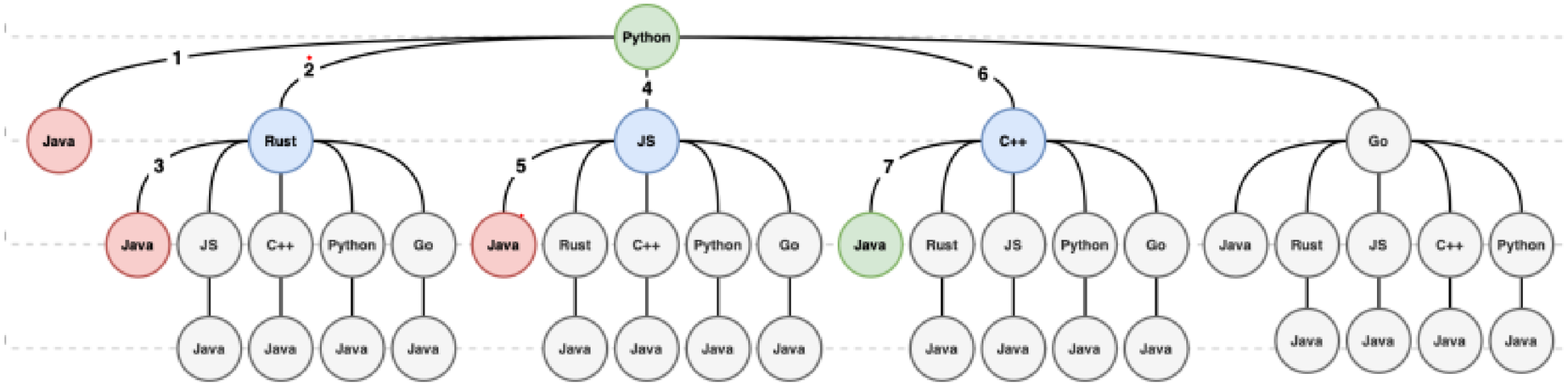
- l’80% degli errori di traduzione C++ → GO sono dovuti a differenze semantiche e sintattiche
- il 23.1% degli errori si verifica quando si traduce C++ → C

ALCUNI LINGUAGGI DI PROGRAMMAZIONE SONO PIU’ VICINI TRA LORO SUGGERENDO UNA RAPPRESENTAZIONE INTERMEDIA PER LA LORO TRADUZIONE

TECNICA PER MIGLIORARE GLI LLM GIA ESISTENTI

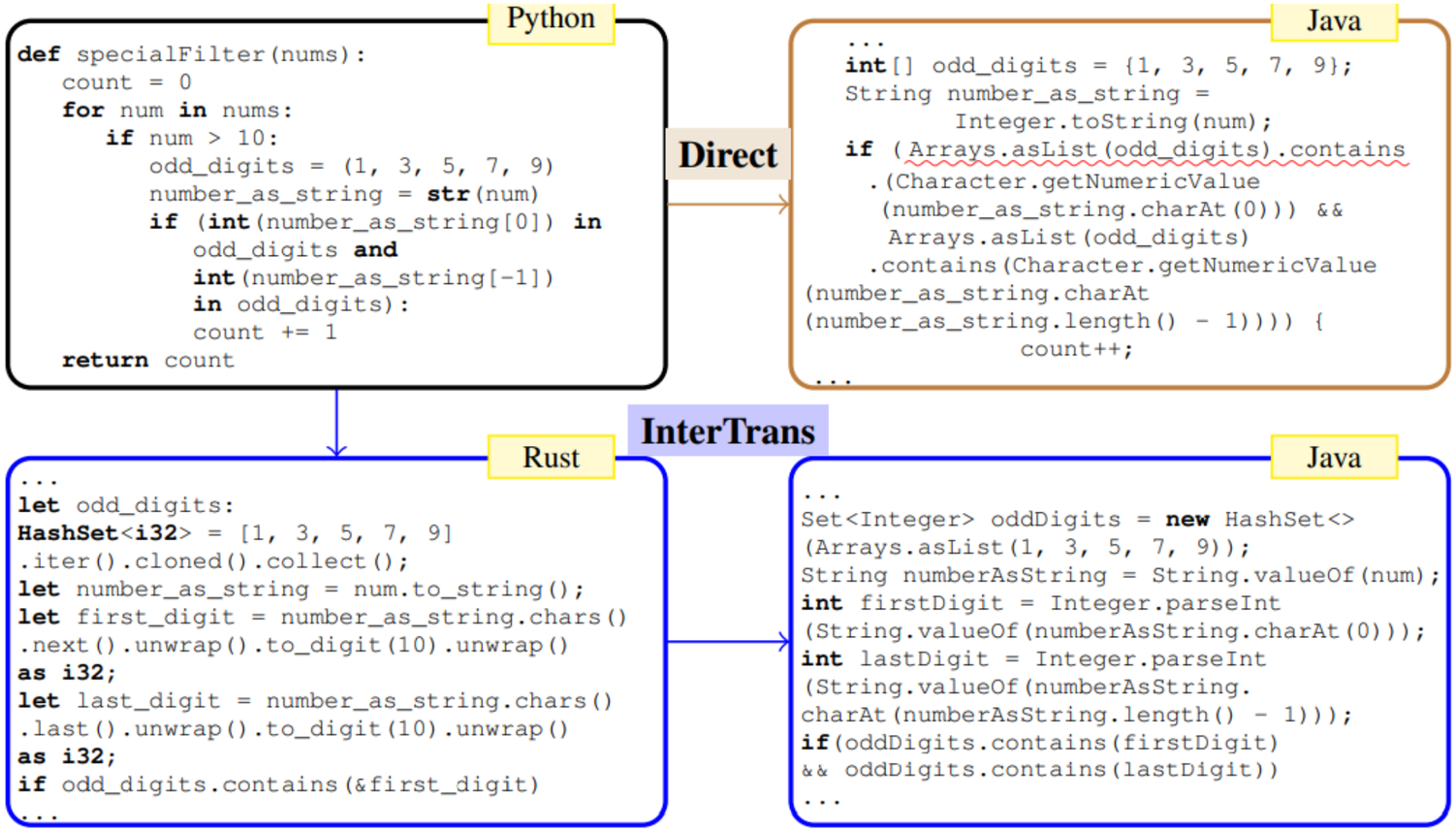
1. **Tree of Code of Translation**: costruzione di un albero di percorsi di traduzione.
2. Per ogni percorso un **LLM genera traduzioni** (fino alla maxDepth)
3. Se il risultato finale passa la **test suite**, ritorna la **traduzione migliore**

   (75% delle traduzioni riuscite richiede in media 3.9 tentativi)



USE CASE

- Python → C++ → Java
- C++ → Rust → Java
- COBOL → C → C++ → Java



1.if int(number_as_string[0]) in odd_digits

- Verifica se una cifra appartiene a una tupla di interi dispari

2.Java accetta solo reference types e non primitive types

- Il codice compila ma fallisce semanticamente

3.Utilizzando come path intermedio Rust il codice funziona correttamente

- Emerge una rappresentazione intermedia che si mappa meglio su Java

BENCHMARK(CODENET DATASET)

LLM BASE	DIRECT [CA*]	INTERTRANS [CA*]	DIFFERENZA
Code Llama	34.6%	60.8%	**26.2%**
Magicoder	49.0%	87.3%	**38.3%**
StarCoder2	41.0%	84.4%	**43.3%**

*Percentuale di traduzioni che producono output funzionalmente equivalenti al codice sorgente

DYNAMIC LANGUAGE PRODUCT LINES

CAZZOLA ET. AL. PROPONGONO UN APPROCCIO ALLA MODERNIZZAZIONE MODULARE DEL SOFTWARE CHE UTILIZZA DEI MICRO-LINGUAGGI PER LA CARATTERIZZAZIONE DELLE APPLICATION FEATURE

UN MICRO LINGUAGGIO È L'INSIEME DI COMPONENTI CHE CARATTERIZZANO UNA FUNZIONE O UNA FEATURE.

```python
def media(valori):
    if len(valori) == 0:
        return 0
    totale = 0
    for v in valori:
        totale += v
    return totale / len(valori)
```

Nell'articolo viene utilizzato Neverlang, un framework creato dagli stessi ricercatori: combina micro-linguaggi a runtime per supportare codice legacy e nuove estensioni simultaneamente.

```
MICRO_LANGUAGE μ_media:

  FEATURE: function_definition
    syntax: FUNCTION <name>(<parameters>) BLOCK
    semantics: creates a callable block of code

  FEATURE: conditional_statement
    syntax: IF <condition> THEN BLOCK
    semantics: executes BLOCK when condition is true

  FEATURE: return_statement
    syntax: RETURN <expression>
    semantics: exits the current function returning a value

  FEATURE: for_loop
    syntax: FOR <var> IN <collection> DO BLOCK
    semantics: iterates BLOCK once for each element

  FEATURE: assignment
    syntax: <var> = <expression>
    semantics: binds a value to a variable

  FEATURE: augmented_assignment
    syntax: <var> += <expression>
    semantics: increments a variable by a given value

  FEATURE: arithmetic_operations
    syntax: <expr1> + <expr2>
            <expr1> / <expr2>
    semantics: numeric operations

  FEATURE: builtin_len
    syntax: LEN(<collection>)
    semantics: returns size of the collection
```

TOOL AUTOMATICI BASATI SU AI

- APPROCCIO BASATO INTERAMENTE SU AGENTI AI
- PROGETTATI DA PROFESSIONISTI DEL SETTORE

MICROSOFT CAMF

FRAMEWORK DI MIGRAZIONE PER TRASFORMARE CODICE COBOL LEGACY IN APPLICAZIONI JAVA MODERNE

- 3 AGENTI AI
- PROMPT ENGEENIRING (NO TRAINING DATA)
- PROGETTO IN COLLABORAZIONE CON BANKDATA

Julia Kordick,Gustav Kaleta -2025

ARCHITETTURA ED ORCHESTRATOR

MICROSOFT SEMANTIC KERNEL →

- Framework per orchestrare agenti AI in pipeline
- Ogni agente riceve l'output dell'agente precedente

LLM SETTING →

- Azure OpenAi
- Temperature : 0.1
- TopP: 0.5
- Maxtokens: 32768

HYBRID DATABASE →

- SQLite : File Cobol,Analisi AI, Codice Java metada
- Neo4j: Analisi del grafo delle dipendenze

AGENT 1: COBOLAnalyzerAgent "Parsing Engine"

INPUT: Codice COBOL grezzo
OUTPUT: Structured analysis data

Estrae: Data Division, Procedure Division, variabili (PIC),paragraphs, embedded SQL/DB2, file access patterns

AGENT 2: DependencyMapperAgent "Cervello Architetturale"

INPUT: Analysis data da Agent 1
OUTPUT: Grafo dipendenze + Diagrammi Mermaid

Rileva 8 tipi: CALL | COPY | PERFORM | EXEC SQL | READ | WRITE | OPEN | CLOSE

AGENT 3: JavaConverterAgent
"Transformation Engine"

INPUT: Analysis + Dependency context
OUTPUT: Java Quarkus production-ready

Anti-pattern JOBOL: PERFORM → loops | GOTO → if-else/switch

PIPELINE

FASE 1: PREPARATION (Preparare codice COBOL per comprensione AI)

- **REVERSE ENGINEERING**
- **CODE CLEANING**
- **TRANSLATION**

FASE 2: ENRICHMENT (Arricchire codice con context semantico per AI)

- **ADD MEANINGFUL COMMENTS**
- **IDENTIFY DETERMINISTIC STRUCTURES**
- **DOCUMENT TEMPORARY RESULTS**

FASE 3: AUTOMATION AID (SUPPORTARE MIGRAZIONE CON ARTEFATTI E ANALISI)

- **FLOW ANALYSIS & VISUALIZATION**
- **DIAGRAMMI MERMAID/FLOWCHARTS**
- **TEST GENERATION**
- **ISOLATE UTILITY FUNCTIONS**

COBOL Migration Insights

Ask questions about your COBOL code migration

SQLite Neo4j

Migration Run: Run 9 (Current)

MCP Resources

View All Runs & Data Guide

Architecture Documentation

Example Queries

COBOL Analysis

"Describe the top 3 critical copybooks and what they do"

"Suggest a carveout strategy and where I should start"

"Give me an overview of what the COBOL code is responsible for"

"Which COBOL files have the highest impact?"

Migration Planning

"What's the recommended migration order for these programs?"

"Identify the main entry point

Ask a Question

Try asking:

Circular Dependencies | Critical Files | Impact Analysis | Copybook Usage | Dependency Summary | Main Programs

Prompt

e.g., What are the circular dependencies in this codebase?

Send

Database Query

Azure OpenAI

Building Response

Response

✅ Switched to Run 9

You can now query data from this migration run.

Note: If runs analyzed the same COBOL files, the dependency graph will be identical.

Dependency Graph | Run 9

Graph Controls

Query: Full Graph Layout: Force Directed

Refresh | Fit | Stabilize

Node Filters

Programs | Copybooks | Called Programs

Edge Filters

CALL | COPY | PERFORM | EXEC | READ | WRITE | OPEN | CLOSE

Legend

Program | Copybook | Called Prog

CALL | COPY | PERFORM

EXEC | READ | WRITE

OPEN | CLOSE

PUNTI DI FORZA

- Non e’ richiesto training
- Costi bassi (all’incirca 0.31$ dichiarati per 102 file)
- Testato su codice reale
- Architettura modulare: per l’analisi possiamo usare solo il DepencyMapperAgent

LIMITAZIONI

- Richiedere AzureOpenAi
- Call-chain depth
- Test suite esistente per validazione

REFORGE-AI

è un sistema agente AI (GPT-4)

- **GENERA DOCUMENTAZIONE** COMPLETA E PIANO DI TRASFORMAZIONE,
- ESEGUE AUTOMATICAMENTE IL **REFACTORING DEL CODICE** VERSO FRAMEWORK MODERNI UTILIZZANDO COME BLUEPRINT LA DOCUMENTAZIONE
- SPECIFICO SU **Java legacy → Spring Boot**

Gian Paolo Santopalo - 2025

ARCHITETTURA ED ORCHESTRATOR

CREWAI FRAMEWORK

- Framework per orchestrare agenti AI in pipeline
- Ogni agente riceve l'output dell'agente precedente

DUE CREW PRINCIPALI

- Documentation Crew
- Gen Code Crew

FEEDBACK LOOP UMANO

- Tra la fase 1 e la fase 2 un rewier umano valida il piano di migrazione

FASE 1 DOCUMENATION CREW

DOCUMENTATION CREW (FASE 1)
Agenti di analisi che scansionano il codebase

OUTPUT:
- Architecture Documentation (Mermaid diagrams)
- Module Analysis
- Dependency graphs
- plan.yaml (transformation plan)

- SCANSIONE AUTOMATICA DEL CODICE
- ANALISI ARCHITETTURALE
- GENERAZIONE DIAGRAMMI MERMAID
- FEEDBACK UMANO

FASE 2 GEN CODE CREW

CODE GENERATION AGENT

Migrazione usando la documentazione come prompt context

- EJB → Spring
- Java EE →Spring Boot idioms

DEPENDECY UPDATE AGENT

- Java 8 → Java 21
- Java EE patterns → Spring Boot 3
- Genera Unit Tests (JUnit 5)

COMPLIANCE CHECK AGENT

- Verifica Security Annotations
- Applica Style Guides
- OWASP best practices

PUNTI DI FORZA

- Non e’ richiesto training
- Human-in-the-loop validation
- Testato su codice reale

LIMITAZIONI

- Costi elevati (API GPT-4)
- Specializzato Java Legacy
- Build Agent Hallucinations
- Inconsistent Model Outputs

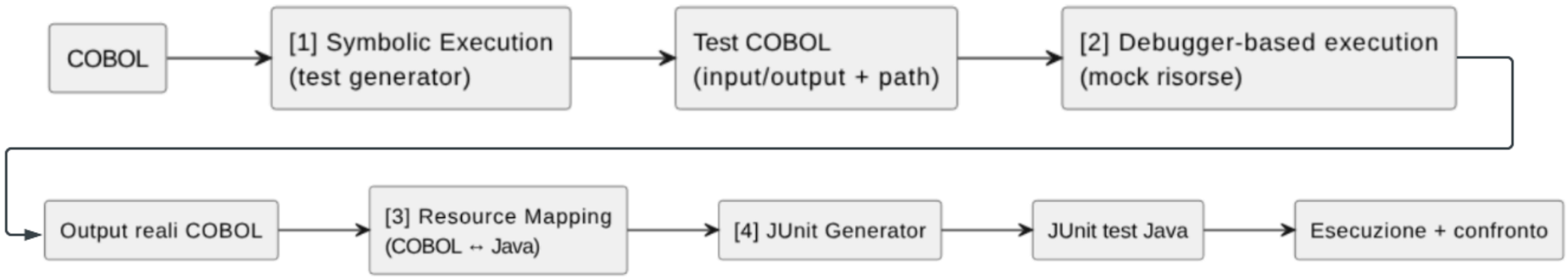
VALIDAZIONE E TESTING AUTOMATICO

SANDEEP ET. AL. PROPONGONO UN FRAMEWORK CON UN SISTEMA DI FEEDBACK LOOP AI-BASED PER LA MIGRAZIONE DEL CODICE COBOL IN JAVA.

LA MIGRAZIONE COBOL → JAVA È STATA ESEGUITA VIA WCA4Z (IBM WATSONX CODE ASSISTANT FOR Z)

Elemento	Finalità
COBOL Test Generation	Generare input di test per il codice COBOL originale
Intermediate Representation (IR)	Supportare l'analisi e la generazione di test
Mocking di risorse esterne	Consentire l'esecuzione dei test senza dipendenze esterne
JUnit Test Generator	Creare test Java equivalenti ai test COBOL
Equivalence Checking	Confrontare output COBOL vs Java per verificare correttezza
Feedback Loop	Identificare difetti nella trasformazione, dare correzioni
Coverage Criteria	Guidare la generazione di test per massimizzare la copertura dei percorsi

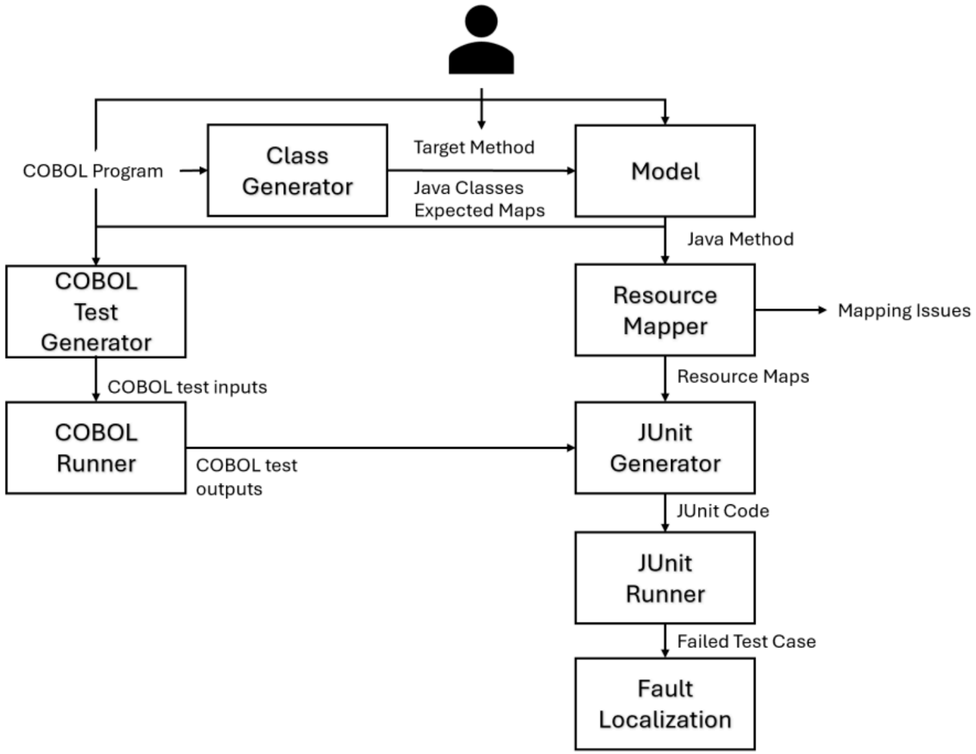
IL SISTEMA TESTA PRIMA IL CODICE COBOL, PER POI PASSARE AL TEST EQUIVALENTE DEL CODICE JAVA:



IN QUESTO MODO CI SI ASSICURA CHE IL CODICE JAVA EFFETTUI LE STESSE OPERAZIONI DEL CODICE IN COBOL

I TEST HANNO PRODOTTO RISULTATI POSITIVI:

- Alta copertura dei test: 84% dei paragrafi COBOL raggiunge full branch coverage e oltre il 92% su un dataset di 94.000 paragrafi
- Scalabilità elevata: generazione dei test <1 secondo
- Validazione accurata del Java generato: i test JUnit riproducono fedelmente il comportamento COBOL e individuano bug reali nella traduzione
- Riduzione del lavoro manuale: il tempo di validazione si riduce fino all'80% rispetto al processo tradizionale.
- Gestione delle risorse enterprise: il framework supporta SQL, CICS, IMS e file I/O



BENCHMARK E VALUTAZIONE

CODEXGLUE

BENCHMARK MULTI-TASK CON 10 TASK E 14 DATASET PER VALUTARE MODELLI SU CODE UNDERSTANDING E GENERATION, CON 3 BASELINE (CODEBERT, CODEGPT, ENCODER-DECODER)

CODE-CODE	TEXT-CODE	CODE-TEXT	TEXT-TEXT
CLONE DETECTION DEFECT DETECTION CLOZE TEST CODE COMPLETION CODE REPAIR CODE TRANSLATION	NATURAL LANGUAGE CODE SEARCH TEXT-TO-CODE GENERATION	CODE SUMMARIZATION	DOCUMENTATION TRANSLATION