Ciência de Dados e I.A.
Escola de Matemática Aplicada
Fundação Getúlio Vargas

Engenharia de Requisitos

TCC

Reverse Engineering of Source Code to Sequence Diagram Using Abstract Syntax Tree

Aluno: Isabela Yabe
Orientador: Rafael de Pinho André
Escola de Matemática Aplicada, FGV/EMAp
Rio de Janeiro - RJ.

Rio de Janeiro, 2025

1 Revisão literária

Artigo revisado Sabir *et al.* (2019):

A revisão tem o objetivo de compreender o estado da arte das abordagens de engenharia reversa que partem de código-fonte e produzem artefatos de alto nível, como diagramas UML. Para garantir uma análise sistemática e comparável entre diferentes propostas, foram definidas perguntas de pesquisa (*Research Questions — RQs*) que orientam a coleta e síntese dos dados extraídos dos estudos selecionados.

- **RQ1.** Em quais linguagens e domínios as abordagens que partem de código-fonte foram aplicadas?
- **RQ2.** Quais modelos/artefatos de alto nível são gerados?
- **RQ3.** Qual aspecto é privilegiado (estático, dinâmico, híbrido) e com qual objetivo (compreensão, redocumentação, migração, qualidade)?
- **RQ4.** Quais técnicas e transformações viabilizam a passagem do código para o modelo de alto nível?
- **RQ5.** Quais ferramentas/frameworks são utilizados?
- **RQ6.** Como as abordagens são validadas e com que qualidade prática?

2 RQ1. Em quais linguagens e domínios as abordagens que partem de código-fonte foram aplicadas?

O domínio abordado é Java, em sistemas legados orientados a objetos. A proposta extrai e transforma código-fonte Java em modelos estruturais e comportamentais UML.

3 RQ2. Quais modelos/artefatos de alto nível são gerados?

São gerados os modelos de alto nível *Class Diagram* e *Activity Diagram*, entregues em um pacote UML. Também são produzidos artefatos de suporte, como o *Intermediate Model* (IM), em formato XML, que viabiliza as transformações T2M/M2M. Cada *Function Behavior* atua como ponte para a construção das atividades correspondentes a cada operação.

4 RQ3. Qual aspecto é privilegiado (estático, dinâmico, híbrido) e com qual objetivo (compreensão, redocumentação, migração, qualidade)?

O aspecto privilegiado é o estático, o pipeline realiza parsing do código, gera a AST e a transforma em modelo intermediário, culminando em representações UML. Trata-se de um processo *Text-to-Model* (T2M) voltado à compreensão e redocumentação do sistema.

5 RQ4. Quais técnicas e transformações viabilizam a passagem do código para o modelo de alto nível?

A passagem do código para modelo UML é viabilizada por um pipeline T2M/M2M em duas fases, (i) descoberta do modelo intermediário (IMD), (ii) gerador do modelo UML.

Na primeira fase o código é parseado para um modelo de fácil compreensão (JavaParser) e transformado numa AST com apenas informações relevantes, então informações de estrutura e comportamento são extraídos da AST gerando o modelo intermediário (IM) em XML (UML2-EMF).

Na segunda fase são aplicadas regras de mapeamento do IM para artefatos UML, tanto mapeamento estrutural quando comportamental.

6 RQ5. Quais ferramentas/frameworks são utilizados?

A implementação do *Src2MoF* apoia-se em três pilares tecnológicos: (i) **Eclipse** e a plataforma **UML2/EMF** para implementar o gerador de modelos e as transformações *Text-to-Model*/*Model-to-Model*; (ii) o **JavaParser**, integrado ao *Intermediate Model Discoverer* (IMD), para analisar o código Java e construir uma AST refinada, a partir da qual é derivado um *Intermediate Model* textual em XML; e (iii) o ecossistema **UML2** para materializar os modelos UML de alto nível (diagramas de classes e de atividades) a partir do IM.

Especificamente, o *model generator* é "carried out using Eclipse tool" e suas regras de transformação são "written and implemented in UML2 platform", acionando dois *templates* (*Class Diagram Creator* e *Activity Diagram Creator*). Já o IMD "uses an open source tool 'Java Parser'" para produzir a AST e, a partir dela, extrair estrutura e comportamento para um IM em XML. Para a comparação manual na avaliação, os autores mencionam Papyrus, StarUML, Rational Rose e o editor UML do Eclipse; tais ferramentas não integram o *pipeline* automático do *Src2MoF*.

7 RQ6. Como as abordagens são validadas e com que qualidade prática?

A validação do *Src2MoF* fundamenta-se na comparação entre modelos gerados automaticamente e modelos elaborados manualmente por especialistas, configurando uma evidência empírica e qualitativa, sem o uso de métricas quantitativas de acurácia ou desempenho. Nessa configuração, engenheiros de software constroem diagramas de classes e de atividades com ferramentas de modelagem (Eclipse, Papyrus, StarUML e Rational Rose). Em seguida, o mesmo código-fonte é processado pelo *Src2MoF*, e os resultados são confrontados para avaliar a correspondência estrutural e comportamental entre as abordagens.

O protocolo experimental foi aplicado a cinco estudos de caso, sendo dois detalhados no artigo: *Automated Teller Machine (ATM)* e *Amadeus Hospitality*. Os resultados indicam que os diagramas de classes gerados refletem atributos, operações e relacionamentos observáveis no código, sugerindo boa fidelidade estrutural. Do ponto de vista comportamental, o framework deriva uma `Activity` para cada método, convertendo o corpo da operação em `OpaqueAction` e utilizando elementos como `CallOperationAction`, `CreateObjectAction`, nós iniciais e finais, condicionais e fluxos de controle e de objeto. Essa sistematização demonstra que estrutura e comportamento são extraídos de forma coordenada a partir do código-fonte.

Entretanto, a qualidade prática permanece no domínio da conformidade percebida: não há métricas objetivas de precisão, *recall* ou tempo de execução, tampouco análise de concordância entre avaliadores. Além disso, a implementação divulgada restringe-se à linguagem Java, foca o diagrama *Activity* como artefato comportamental e não cobre certos construtos, como nós *merge* e *fork*. Essa ausência de medidas quantitativas limita a reprodutibilidade e a generalização dos resultados.

Autores / Referência	**Linguagem / Domínio**	**Modelo Gerado**	**Aspecto**	**Técnica / Transformação**	**Ferramenta / Framework**	**Validação / Estudo de Caso**
Sabir *et al.* (2019)	Java (sistemas legados orientados a objetos)	UML *Class Diagram* + *Activity Diagram* (em pacote UML)	Estático; objetivo: compreensão/redocumentação	T2M/M2M em duas fases: Parser→AST→IM(XMI) e mapeamentos IM→UML (classe/atividade)	Eclipse + UML2/EMF; MoDisco (XMI); (Papyrus/StarUML/Rational Rose na validação manual)	Comparação especialista (modelos manuais vs. gerados), 5 estudos de caso; ATM e Amadeus descritos

Tabela 1: Resumo das abordagens

Referências

SABIR, U. *et al.* A Model Driven Reverse Engineering Framework for Generating High Level UML Models from Java Source Code. *IEEE Access*, v. 7, p. 158931–158950, 2019.