



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

Engenharia de Requisitos

TCC

**A Model Driven Reverse
Engineering Framework for
Generating High Level UML
Models From Java Source Code**

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

Neste trabalho, o domínio é Java, a abordagem extrai e transforma código-fonte Java em modelos estruturais e comportamentais UML.

3 Quais modelos/artefatos de alto nível são gerados?

São gerados os modelos de alto nível finais Class Diagram e Activity Diagram, o resultado é entregue como um UML package com esses modelos. Também são gerados artefatos de suporte, o Intermediate Model (XML) viabiliza as transformações T2M/M2M e cada Function Behavior serve de ponte para construir o activity por operação.

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

O aspecto privilegiado é o estático (parsing do código -> AST -> modelo intermediário -> UML), todo o pipeline é T2M estático (texto -> modelo). Com objetivo principal de compreensão e redocumentação.

5 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 Quais ferramentas/frameworks são utilizados?

A implementação do *Src2MoF* apoia-se em três pilares tecnológicos: (i) **Eclipse** (IDE e ecossistema MDE) para implementar o gerador de modelos e as transformações *Text-to-Model* (*A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code*: “In *Src2MoF*, Eclipse tool is used, which facilitates the Intermediate Model Discoverer (IMD) implementation and UML2 modeling. Our model generator is using Text-to-model transformations which is also implemented in Eclipse.” :contentReference[oaicite:6]index=6); (ii) o **parser de código-fonte Java** (JavaParser) integrado ao *Intermediate Model Discoverer* (IMD) para produzir uma AST e um *Intermediate Model* textual em XML (*A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code*: “Source code parser uses an open source tool ‘Java Parser’ ... it parses the source code and passes an abstract syntax tree AST to model discovery phase ... Implemented IMD is generic and language independent ... This intermediate representation is an XML like depiction of source code.” :contentReference[oaicite:7]index=7 :contentReference[oaicite:8]index=8); e (iii) a plataforma **UML2** sobre **Eclipse/EMF** para materializar os modelos UML de alto nível (diagramas de classes e de atividades) e para a qual os dois *templates* (*Class Diagram Creator* e *Activity Diagram Creator*) foram escritos (*A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code*: “UML2 is an Eclipse Modeling Framework (EMF) based implementation of UML 2.x ... Transformation templates i.e. class diagram creator and activity diagram creator are written for UML2.” :contentReference[oaicite:9]index=9).

Especificamente, o *model generator* “calls two templates i.e. class diagram creator and activity diagram creator” e seu “Implementation ... is carried out using Eclipse tool” (*A Model Driven Reverse Engineering Framework for Generating High Level UML Models From Java Source Code*). :contentReference[oaicite:10]index=10

Na avaliação, os autores também mencionam ferramentas de modelagem usadas para a comparação manual (Papyrus, StarUML, Rational Rose, editor UML do Eclipse), ainda que não façam parte do *pipeline* automático do *Src2MoF* (*A Model Driven Reverse Engineering Framework for Generating High Level UML Models*

From Java Source Code: “experts have manually performed modeling ... by means of different software design tools i.e. Eclipse, papyrus, starUML, rationalrose”.
:contentReference[oaicite:11]index=11

7 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* (IM) 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*) Sabir *et al.* (2019). 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 Sabir *et al.* (2019). 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*.

Síntese (RQ5). Eclipse + UML2/EMF (T2M/M2M); JavaParser (código → AST → IM/XML); Eclipse, Papyrus, StarUML, Rational Rose (apoio à validação).

8 Como as abordagens são validadas e com que qualidade prática?

Na validação, os autores comparam a saída do *Src2MoF* com modelos elaborados manualmente por especialistas em ferramentas como Eclipse, Papyrus, StarUML e Rational Rose. Trata-se de uma evidência *empírica e qualitativa* baseada em julgamento de especialistas, sem métricas quantitativas de acurácia (p. ex., precisão/recall) ou tempo de execução.

A avaliação ocorre em duas etapas: (1) modelagem manual (diagramas de classes e de atividades) por engenheiros de software; (2) execução do *Src2MoF* nos mesmos casos; (3) comparação entre as saídas. São cinco estudos de caso de referência; dois são detalhados no artigo: *Automated Teller Machine (ATM)* e *Amadeus Hospitality*.

Os diagramas de classes gerados refletem atributos, operações e relacionamentos observados no código.

Para cada método, é criada uma *Activity*; o corpo da operação é traduzido para *OpaqueAction*, com uso de *CallOperationAction*, *CreateObjectAction*, nós inicial/final, nós condicionais e fluxos de controle/objeto.

Síntese (RQ6). *Método:* comparação especialista (modelos manuais vs. gerados). *Escopo:* 5 estudos de caso (ATM e Amadeus descritos). *Qualidade prática:* conformi-

dade percebida de estrutura e comportamento; *sem* métricas quantitativas. *Ameaças*: linguagem (Java), suporte parcial em *Activity*, generalização limitada.

9 Como as abordagens são validadas e com que qualidade prática?

A validação do *Src2MoF* baseia-se na comparação entre os modelos gerados automaticamente e modelos elaborados manualmente por especialistas, configurando uma evidência empírica e qualitativa, sem reporte de métricas quantitativas de acurácia ou de tempo de execução. Nessa configuração, engenheiros de software primeiro constroem, com ferramentas de modelagem (Eclipse, Papyrus, StarUML e Rational Rose), diagramas de classes e de atividades a partir do mesmo código que, em seguida, é processado pelo *Src2MoF*; a análise confronta as duas saídas. O protocolo é aplicado a cinco estudos de caso de referência, dos quais dois são detalhados no artigo: *Automated Teller Machine (ATM)* e *Amadeus Hospitality*. Permitindo observar recorrências e limites no comportamento do framework.

Nos resultados, os diagramas de classes produzidos refletem, de forma consistente, atributos, operações e relacionamentos observáveis no código, sugerindo boa correspondência estrutural. Do ponto de vista comportamental, o framework deriva uma *Activity* para cada método, traduzindo o corpo da operação em *OpaqueAction* e mobilizando *CallOperationAction* e *CreateObjectAction*, além de nós iniciais e finais, nós condicionais e fluxos de controle e de objeto; tal mapeamento sustenta a reivindicação de que estrutura e comportamento são gerados de maneira conjunta a partir do código.

Contudo, a qualidade prática reportada permanece no domínio da conformidade percebida: não há estimativas de precisão/recall, nem análise de concordância entre avaliadores, escalabilidade ou custo temporal. Além disso, a implementação divulgada restringe-se à linguagem Java, foca *Activity* como diagrama comportamental e não cobre certos construtos (por exemplo, nós *merge* e *fork*).

Autores / Referência	Linguagem / Domínio	Modelo Gerado	Aspecto	Técnica / Transformação	Ferramenta / Framework	Validação / Estudo de Caso
Sabir <i>et al.</i> (2019)	Java (sistemas legados orientados a objetos)	UML <i>Class Diagram</i> + <i>Activity Diagram</i> (em pacote UML)	Estático; objetivo: compreensão/redocumentação	T2M/M2M em duas fases: Parser → AST → IN (UML) → M2M (mapeamento IM → UML (classe/atividade))	Eclipse + UML2/EMF; (UML)ser (IMD); (Papyrus/StarUML/Rational Rose na validação manual)	Comparação especialista (modelos manuais vs. automáticos), 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.