



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

Engenharia de Requisitos

TCC

An Approach for Extracting UML Diagram from Object-Oriented Program Based on J2X

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 Zhang (2016):

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?

A implementação e avaliação concentram-se na linguagem Java. Os autores testam a abordagem em cinco pequenos sistemas orientados a objetos de propósito geral: *Minesweeper*, *Blog*, *PayrollSys*, *eLib* e *myAlgsLib*. Esses casos evidenciam a aplicação em programas de escopo reduzido e natureza genérica, sem vínculo a um domínio específico.

Embora a experimentação esteja restrita ao ecossistema Java, o artigo destaca que o uso do intermediário J2X foi concebido para “isolar diferenças de linguagem” e permitir a extensão do método a outras linguagens.

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

O método gera dois artefatos UML de alto nível: o diagrama de classes (estrutura) e o diagrama de sequência (comportamento de interação). O processo ocorre em três etapas: (i) transformação do código em representação J2X, (ii) extração estrutural para o diagrama de classes e (iii) análise de fluxo (OFG e CFG) para construir o diagrama de sequência. Assim, a cadeia de transformação é: *código* \rightarrow *J2X* \rightarrow *OFG/CFG* \rightarrow *UML*.

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

A abordagem é predominantemente **ESTÁTICO**, operando sobre o código-fonte sem execução. O objetivo central é a **COMPREENSÃO** e **REDOCUMENTAÇÃO** de sistemas OO, elevando o nível de abstração por meio da geração de diagramas UML de classes e sequência.

Extensão proposta (minha proposta). Os autores reconhecem a possibilidade de um futuro híbrido (estático + dinâmico) para aumentar a acurácia. Como extensão, propomos um **HÍBRIDO** que combine a análise estática com uma *análise de intenção* (o que o software está intencionado a fazer), inferida de indícios semânticos do código (nomes, contratos, comentários) e correlacionada a casos de uso. A hipótese é que a integração entre (**ESTÁTICO**) e intenção de alto nível melhore a precisão na identificação de interações relevantes e reduza ambiguidades na geração de UML Sequência, mantendo o baixo custo de coleta (sem execução).

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

O processo segue um pipeline de quatro fases principais:

1. **Código** → **AST** → **J2X (IR)**: parsing e anotação semântica do código em linguagem intermediária XML (J2X), que padroniza elementos e isola diferenças de linguagem.
2. **J2X** → **metadados** → **UML Classe**: extração de classes, interfaces, métodos e campos, mapeando relações de generalização, implementação, associação e dependência.
3. **OFG (Object Flow Graph)**: refinamento das relações por rastreamento de fluxos de objetos, reduzindo falsos positivos/negativos e tratando polimorfismo.
4. **J2X** → **CFG** → **(OFG + CFG)** → **UML Sequência**: Para a geração do diagrama de sequência, o método modela as chamadas de métodos e as estruturas de controle do programa em um *Control Flow Graph* (CFG). Em seguida, esse grafo é combinado ao *Object Flow Graph* (OFG), de modo a associar o fluxo de controle às instâncias de objetos efetivamente envolvidos nas interações. Essa integração permite identificar, de forma estática, os objetos participantes, as mensagens trocadas e as condições e repetições (**alt/opt/loop**) que estruturam o comportamento do sistema.

OFG+CFG, efeito prático: mesmo sem instrumentação ou execução, a combinação preserva a ordem e a lógica das interações entre objetos, possibilitando a geração automática de *UML Sequence Diagrams* completos, contendo *lifelines*, *messages* e *interaction fragments*.

6 RQ5. Quais ferramentas/frameworks são utilizados?

O ecossistema técnico inclui:

- **J2UML:** ferramenta desenvolvida pelos autores que orquestra a extração e geração dos diagramas UML.
- **JavaCC:** gerador de analisadores automáticos usado para construir a AST a partir do código-fonte.
- **DOM4J:** biblioteca para manipulação eficiente do XML que representa o modelo intermediário J2X.
- **J2X (DTD/XML):** linguagem intermediária padronizada para estruturar e armazenar informações semânticas do programa.

Os experimentos foram realizados em um ambiente Windows 32-bit, com 3GB de memória e CPU Intel Core 2 Duo a 2,93GHz, garantindo reprodutibilidade dos resultados.

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

Os autores implementam a ferramenta J2UML e a avaliam em um conjunto de casos de teste de pequeno porte, medindo desempenho (tempo de execução) e exatidão na extração do diagrama de classes. Os dados observados são: número de classes, tempo, e contagens extraídas pela ferramenta.

A “acurácia” é calculada por comparação manual com um conhecimento de referência derivado do código dos cinco casos, tanto para classes quanto para relações.

Interpretação dos autores. A extração de classes é mais precisa que a de relações; os autores consideram os erros “aceitáveis” e afirmam que sua análise de fluxo de objetos ajuda a obter relações mais acuradas.

A validação foi conduzida por estudo experimental com cinco sistemas OO de pequeno porte, medindo tempo e acurácia da extração do diagrama de classes. A acurácia foi definida como a razão entre itens corretamente extraídos e o ground truth manual (classes e relações). Os resultados indicam acurácia de 96,4–100% para classes e 65,0–90,4% para relações, com desempenho considerado aceitável, variando conforme a complexidade das dependências.

Síntese (RQ6). *Desenho:* estudo experimental com 5 sistemas OO. *Métricas:* tempo e acurácia (classes e relações). *Resultados:* classes 96,4–100%; relações 65,0–90,4%. *Qualidade prática:* boa precisão estrutural; limitações de escala e avaliação.

Autores / Referência	Linguagem / Domínio	Modelo Gerado	Aspecto	Técnica / Transforma- ção	Ferramenta / Fra- mework	Validação / Estudo de Caso
Zhang (2016)	Java; peque- nos sistemas OO (eLib, Mineswee- per, Blog, PayrollSys, myAlgsLib)	UML Classe; UML Sequên- cia	Estático — compreensão e redocumen- tação	Código→AST→J2X; sentenças simplif.→OFG; CFG+OFG→Se- quência	J2X; ML; JavaCC; DOM4J; J2X (eLib, XML)	5 casos pe- quenos; acu- rácia: classes 96,4–100%; relações 65,0–90,4%

Tabela 1: Resumo das abordagens

Referências

ZHANG, H. An Approach for Extracting UML Diagram from Object-Oriented Program Based on J2X. Versão inglesa. *In*: INTERNATIONAL Forum on Mechanical, Control and Automation (IFMCA 2016). Changchun, China: Atlantis Press, 2016. p. 266–276. Published in Advances in Engineering Research, vol. 113.