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?

No artigo analisado, tanto a implementação da ferramenta quanto a avaliação empírica concentram-se na linguagem Java. Em termos de domínio, os autores testam a abordagem em pequenos sistemas orientados a objetos de propósito geral, típicos de benchmarks didáticos e bibliotecas: um jogo (*Minesweeper*), um blog (*Blog*), um sistema de folha de pagamento (*PayrollSys*), uma e-library (*eLib*) e uma biblioteca de algoritmos (*myAlgsLib*). Esse conjunto de casos sinaliza que a aplicação prática recai sobre programas de escopo reduzido e natureza genérica, sem amarração a um domínio específico de negócio.

Por fim, embora todo o exercício experimental esteja restrito ao ecossistema Java, os autores afirmam que o uso do intermediário J2X foi concebido para "isolar diferenças de linguagem" e, assim, permitir a reutilização do mesmo método de *reverse* em outras linguagens, uma generalidade pretendida no plano conceitual, mas não demonstrada empiricamente no estudo.

Síntese de RQ1: Linguagem aplicada: Java (implementação e avaliação).

Domínio dos estudos: pequenos sistemas OO genéricos (jogo, blog, sistema de folha de pagamento, biblioteca de algoritmos, e-library).

Escopo pretendido: método projetado para ser estendido a outras linguagens via J2X (afirmação conceitual, não demonstrada empiricamente no artigo).

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

O método gera dois artefatos UML de alto nível: (i) diagrama de classes (estrutura) e (ii) diagrama de sequência (comportamento de interação).

Operacionalmente, o framework (a) "first transforms source code to J2X representation", (b) constrói o diagrama de classes com base nas informações extraídas, e (c) combina OFG e CFG para "construct UML sequence diagram". Esses elementos (J2X, OFG, CFG) são artefatos intermediários.

Cadeia que conduz a esses modelos (código → J2X → OFG/CFG → UML) e menções a "construct UML class diagram" e "construct UML sequence diagram": Abstract e Approach Overview.

Síntese para o texto do TCC (sugerido): "O método entrega, como artefatos de alto nível, o UML Class Diagram e o UML Sequence Diagram. A geração é suportada por uma cadeia estática que transforma o código em J2X e analisa OFG/CFG, culminando na construção dos diagramas ('construct UML class diagram'; 'construct UML sequence diagram')."

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

A abordagem é de Estático, pois opera exclusivamente sobre o código-fonte, sem instrumentação nem execução. O objetivo principal é Compreensão e Redocumentação de sistemas orientados a objetos, por meio da elevação do nível de abstração: do código para modelos UML. Em particular, gera-se *UML Classe* (estrutura) e *UML Sequência* (interação) a partir de informações derivadas estaticamente do código, o que apoia entendimento, manutenção e documentação do sistema.

Em síntese, trata-se de uma análise Estático que extrai tanto a estrutura (classes e relacionamentos) quanto a interação (mensagens em sequência) com base em *artefatos derivados do próprio código*, preparando o terreno para as transformações discutidas na RQ4.

Extensão proposta (minha versão). 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?

(1) Código → AST → J2X (IR). O método realiza o *parsing* do código, visita a AST e anota semântica em uma linguagem intermediária XML (*J2X*) que padroniza elementos estruturais e "isola diferenças de linguagem". *Efeito prático:* a J2X viabiliza armazenamento/troca de informações e estabiliza as etapas seguintes sem depender novamente do código-fonte.

(2) J2X → metadados + mapeamentos para UML Classe. A partir do J2X, extrai-se a estrutura básica (classes, interfaces, métodos, campos) e mapeiam-se quatro relações estruturais para o diagrama de classes: generalização, implementação, associação e dependencia. *Efeito prático:* torna explícitas hierarquias (generalização/realização) e acoplamentos (associação/dependência), úteis a Compreensão/Redocumentação.

(3) Refinamento de relações com OFG (sentenças simplificadas → OFG). Para aumentar a precisão de associação/dependência, a representação J2X é reduzida a "sentenças simplificadas" e então usada para construir um *Object Flow Graph* (OFG) que rastreia a propagação de objetos. *Efeito prático:* reduz falsos positivos/negativos como polimorfismo nas relações estruturais ao confirmar tipos e fluxos reais de objetos.

(4) J2X → CFG → (OFG + CFG) → UML Sequência. Para o diagrama de sequência, o método modela chamadas e estruturas de controle em um *Control Flow Graph* (CFG) e o combina com o OFG a fim de identificar objetos participantes, mensagens e condições/loops. *Efeito prático:* mesmo sem execução, a combinação OFG+CFG preserva ordem/condição das interações, permitindo gerar *UML Sequência* com *lifelines*, *messages* e *fragments* (alt/opt/loop).

Síntese de RQ4:

- **Intermediário:** Código → AST → **J2X (XML)**.
- **Classe:** metadados + mapeamentos diretos (generalização, realização, associação, dependência).
- **Precisão de relações:** sentenças simplificadas → **OFG** (fluxo de objetos).
- **Sequência: CFG** (fluxo de controle) + **OFG** ⇒ mensagens/condições sem execução.

6 RQ5. Quais ferramentas/frameworks são utilizados?

Em alinhamento com o pipeline técnico descrito na RQ4, o ecossistema de suporte compreende: (i) a ferramenta de engenharia reversa que orquestra a extração e geração de diagramas; (ii) o gerador de *parser* para obtenção da AST; (iii) a infraestrutura

de processamento XML sobre a linguagem intermediária; e (iv) a própria linguagem intermediária (IR), que, embora não seja uma ferramenta, é o artefato central de representação.

Ferramenta de engenharia reversa (J2UML). Os autores implementam a ferramenta que materializa o processo de *reverse*. *Papel no pipeline:* integra a leitura do J2X, aplica mapeamentos para *UML Classe* e combina OFG/CFG para *UML Sequência*, entregando os artefatos de alto nível úteis a Compreensão/Redocumentação.

Gerador de *parser* (JavaCC). O *frontend* do processo usa um gerador de analisadores para construir a AST: ""[The parser is] automatic generated by JavaCC."" (Zhang (2016)) *Papel no pipeline:* viabiliza *Código* $\rightarrow$ *AST*, primeiro passo para rotular semântica no J2X.

Infraestrutura de processamento XML (DOM4J e equivalentes). Para manipular a representação intermediária, os autores recorrem a bibliotecas maduras *Papel no pipeline:* permite consultar/transformar o **J2X (XML)** com eficiência e baixo acoplamento ao *parser*.

Linguagem intermediária (J2X, DTD/XML). Document Type Definition (DTD) define a estrutura do XML usado como IR. A J2X é a IR que armazena semântica e estrutura do programa. A especificação em DTD padroniza elementos e atributos, sustentando a extração estruturada (*Código* $\rightarrow$ *AST* $\rightarrow$ **J2X**).

Ambiente de execução (contexto experimental). Para reprodutibilidade, os autores reportam o contexto de execução: ""The tool runs on a PC with Windows [...] 32bit, 3G Memory and 2.93Ghz Intel Core 2 Duo CPU."" (Zhang (2016)) *Observação:* trata-se de caracterização do *setup* usado nos experimentos, não de requisito da abordagem.

Síntese de RQ5:

- **Ferramenta central:** J2UML (orquestra extração e geração de UML).
- **Parser/AST:** gerado com **JavaCC**.
- **Processamento da IR:** bibliotecas XML (e.g., **DOM4J**).
- **Representação: J2X (DTD/XML)** como linguagem intermediária.
- **Ambiente de execução:** especificado para reprodutibilidade dos resultados.

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

Fonte no artigo e como extraí: toda a evidência de validação está na seção Experiment and Evaluation e na Tabela 4 do paper; ali os autores descrevem o procedimento, os

estudos de caso, as métricas e interpretam os resultados. Extraí os dados diretamente desse trecho.

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 de RQ6:

- **Desenho**: estudo experimental com 5 sistemas OO pequenos; comparação ferramenta vs. *ground truth* manual.
- **Métricas**: tempo de execução; acurácia de classes e relações (definições explícitas).
- **Resultados**: classes ≈ 96,4%–100%; relações ≈ 65,0%–90,4%; tempos em ms.
- **Qualidade prática**: útil para Compreensão/Redocumentação com custo baixo; OFG melhora relações.
- **Limitações**: casos didáticos; pouca evidência para sistemas grandes; *ground truth* manual; sem análise estatística.

Autores / Referência	**Linguagem / Domínio**	**Modelo Gerado**	**Aspecto**	**Técnica / Transformação**	**Ferramenta / Framework**	**Validação / Estudo de Caso**
Zhang (2016)	Java; pequenos sistemas OO (eLib, Minesweeper, Blog, PayrollSys, myAlgsLib)	UML Classe; UML Sequência	Estático — compreensão, manutenção/redocumentação	Código→AST→JDT; mapeamentos (gen./impl./assoc./dep.); sentenças simplif.→OFG; CFG+OFG→Sequência	J2XML; JavaCC; Doxygen; J2X (DTD/XML)	5 casos pequenos; acurácia: 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.