

PREDIÇÃO DE MUDANÇAS EM SISTEMAS COM ARQUITETURA DE LINHA DE PRODUTO DE SOFTWARE

Kevin Grünfeld Strey, Paulo Roberto Farah

INTRODUÇÃO

O aumento da demanda por soluções de software personalizadas e adaptáveis tem impulsionado a adoção de arquiteturas de Linha de Produto de Software (LPS). Esse paradigma permite que uma família de produtos relacionados seja desenvolvida a partir do reuso de partes comuns compartilhadas para atender a um mercado ou domínio específico (Clements; Northrop, 2002; Linden; Schmid; Rommes, 2007). A construção de sistemas personalizados se dá por meio da integração entre um núcleo comum e funcionalidades específicas do sistema (do inglês *features*). Uma *feature* é um aspecto, qualidade ou funcionalidade de um software, visível ou percebida pelo usuário do sistema (Kang et al., 1990). Desse modo, a seleção dessas *features* compõe a versão final de uma Linha de Produto de Software.

Esse tipo de arquitetura de software promove ganhos significativos em termos de produtividade, qualidade e redução de custos, pois incorpora o reuso sistemático no processo de desenvolvimento de software (Galimberti; Wazlawick, 2015). No entanto, a gestão de mudanças nesses sistemas se torna progressivamente complexa à medida que o número de produtos e sua variabilidade aumenta, exigindo abordagens eficazes para antecipar e controlar os impactos e modificações na arquitetura e nos componentes associados. Nesse contexto, a predição de mudanças surge como uma estratégia promissora para apoiar o processo de evolução de sistemas com arquitetura de LPS. Ao identificar, de forma antecipada, os artefatos e componentes mais suscetíveis a modificações, é possível planejar intervenções de maneira mais precisa, reduzir riscos de falhas e otimizar o esforço de manutenção (Farah, 2023).

DESENVOLVIMENTO

A pesquisa se iniciou com a escolha de um repositório LPS e análise do seu modelo de variabilidade, seguiu com a coleta de suas métricas estruturais e evolutivas e sucedeu com análise do projeto e resposta das perguntas de pesquisa.

O repositório escolhido como estudo de caso foi o Eclipse IDE, devido ao seu histórico de desenvolvimento com controle de versões e na sua utilização como projeto LPS em trabalhos relacionados. O mecanismo de implementação de variabilidade do Eclipse IDE, é a variabilidade em tempo de execução, dado que é uma IDE baseada em plug-ins. Conforme documentado pela (ECLIPSE FOUNDATION, 2026), não existe um único projeto monolítico do Eclipse IDE, o ambiente é organizado em múltiplos projetos independentes, onde cada projeto pode contribuir com múltiplos componentes para a IDE. Para isso, o repositório central Eclipse Simultaneous Release¹ atua como um manifesto de integração entre os repositórios independentes das outras *features* que irão compor a release final do Eclipse IDE.

Para identificar as principais *features* do sistema, nos baseamos no trabalho de (Johansen et al., 2012), que mostra um conjunto de 23 *features*, das quais, foram extraídas 19, pois somente estas eram estavam disponíveis em repositórios compatíveis com o sistema de controle de versão e Github.

¹ <https://github.com/eclipse-simrel/simrel.build>

Após a identificação do conjunto de features a ser coletada, foi desenvolvida uma ferramenta de coleta que parte do Eclipse Simrel organizada em um pipeline de dois estágios: o Repository_mapper (Python) e o EvoMetrics Collector (Java). O primeiro estágio é responsável por construir um mapeamento temporal entre as releases do repositório central e os commits dos repositórios individuais de cada feature. O segundo estágio consome esse mapeamento para realizar a extração evolutiva de métricas no escopo de método sobre o código-fonte de cada módulo (Strey; Mendonça; Farah, 2026).

As métricas coletadas pelas ferramentas são 12 métricas evolutivas, tais como Lines Of Code (LOC), média de LOC por método ativo, Birth Of Method (BOM), Changes since the birth (CSB), dentre outras. Também foram coletadas 29 métricas estruturais CK tais como Coupling Between Objects (CBO), variablesQty, parametersQty, dentre outras. A lista completa de todas as métricas coletadas está disponível no Readme² do repositório.

RESULTADOS

A ferramenta desenvolvida permitiu coletar o código das features fragmentadas em múltiplos repositórios, sendo capaz de extrair um conjunto de 12 métricas evolutivas e 29 métricas estruturais CK de 19 features ao longo de 61 releases do projeto Eclipse IDE. Além da extração ela permite a visualização, agrupamento e análise de métricas por meio de gráficos, revelando padrões de manutenção e evolução nas diferentes features.

A partir disso, realizaram-se análises do Eclipse IDE para investigar a evolução do projeto ao longo do tempo, classificar features com características comuns e identificar como refatorações impactam nas métricas.

Em relação a evolução do Eclipse IDE, maioria as features do foram desenvolvidas nas primeiras releases e se mantiveram estáveis ao longo do tempo. No entanto, os valores absolutos de LOC das features EclipseLink, JDT e DataTools aumentaram, indicando a inclusão de novas funcionalidades durante a evolução do projeto.

A partir dessas tendências percebidas, realizou-se a aplicação do algoritmo K-Means, o qual revelou a formação de três perfis evolutivos distintos: C1 (Enxutas e Estáveis), com bases de código reduzidas e um valor baixo de LOC por método ativo; C2 (Médias), com tamanho moderado e manutenção ativa; e C3 (Gigantes Voláteis), caracterizadas por alto volume de código e frequentes refatorações estruturais.

Por fim, constatou-se que as fatoraões que modificam o nome dos métodos ou que alteram sua localização diluem artificialmente o CSB e podem simular uma reinicialização no histórico de alteração dos métodos. Já quando há aumento de LOC e BOM, isso costuma indicar adição de métodos novos no sistema. Diante disso, a análise isolada de métricas pode ser enganosa e a comparação do comportamento de múltiplas métricas é essencial.

Portanto, a capacidade de identificar padrões de manutenção e agregar as features de um sistema em clusters tem implicações importantes para pesquisadores em áreas como testes e predição de defeitos. Modelos preditivos de defeitos baseiam-se fortemente no histórico de mudanças do código para identificar áreas de risco e isso pode enviesar a predição de bugs caso eventos como refatorações não sejam discriminadas das verdadeiras expansões funcionais (Mendonça; Assunção; Vergilio, 2024)

² <https://github.com/KevinStrey/feature-model-eclipse/tree/main>

CONSIDERAÇÕES FINAIS

O presente trabalho apresentou uma ferramenta para coletar e analisar métricas de softwares LPS que implementam um modelo de variabilidade em tempo de execução e possuem suas features distribuídas em múltiplos repositórios. Sua utilização no projeto Eclipse IDE permitiu a identificação de padrões de evolução e a categorização de features em grupos com base nesses padrões, sendo útil para pesquisadores em áreas de priorização de testes e predição de defeitos.

Além dos resultados já obtidos, as métricas coletadas ainda serão utilizadas em trabalhos futuros para treinar os modelos de predição de mudanças através do treinamento de modelos de Machine Learning, empregando algoritmos clássicos como Random Forest, SVM, entre outros, seguida de sua avaliação utilizando métricas como Precisão, Recall e F1-Score.

Palavras-chave: Linha de Produto de Software; Mineração de Repositórios de Software; Predição de Mudanças.

ANEXOS

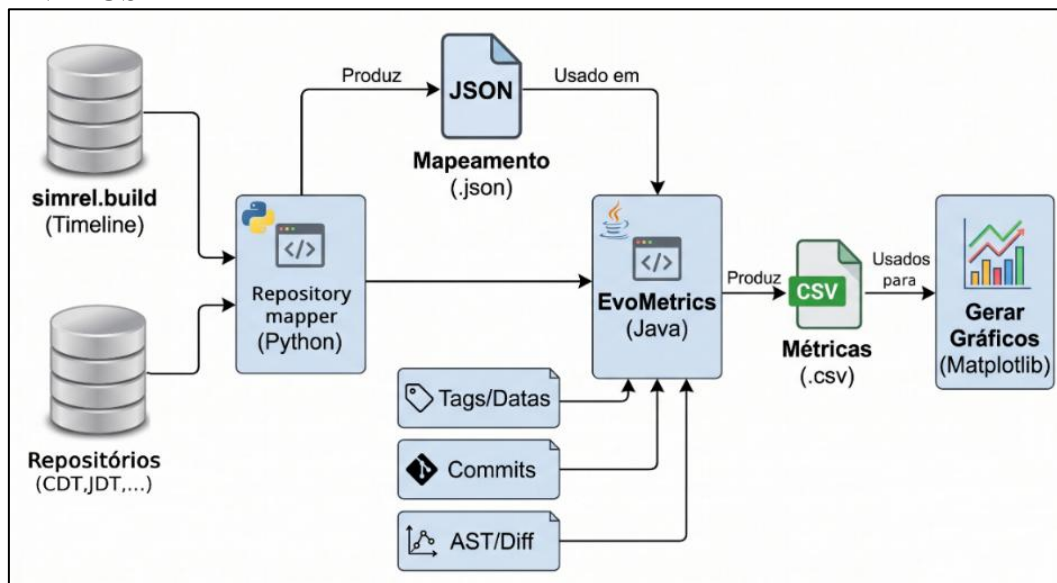


Figura 1: *Diagrama do fluxo de coleta da ferramenta*

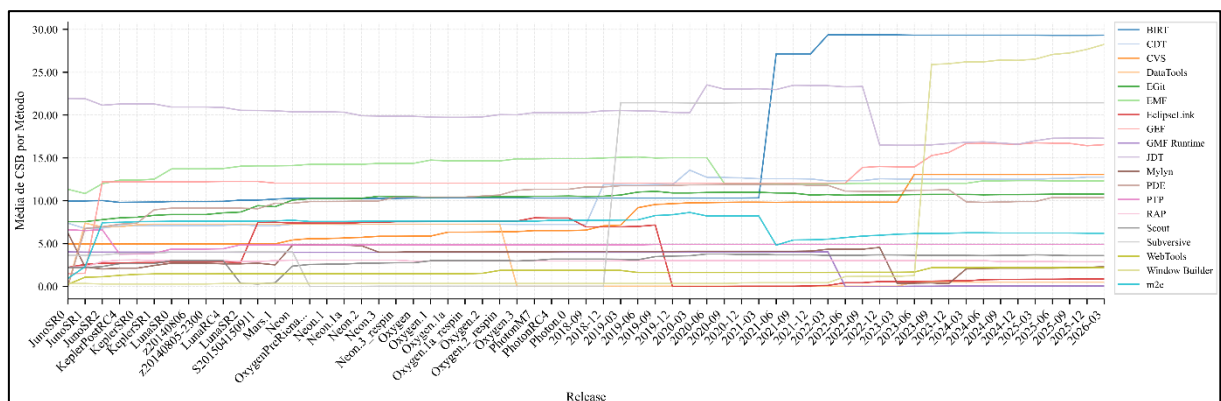


Figura 2: Média de mudanças desde o nascimento (CSB) por método para todas as features.

REFERÊNCIAS BIBLIOGRÁFICAS

CLEMENTS, Paul; NORTHROP, Linda. **Software Product Lines: Practices and Patterns**. [S.l.]: Addison-Wesley, 2002.

ECLIPSE FOUNDATION. **Eclipse IDE: The Eclipse IDE repository**. Ottawa, Canadá, 2026. Disponível em: <<https://github.com/eclipse-ide>>. Acesso em: 5 jun. 2026

FARAH, Paulo Roberto. **Machine learning prediction of change-prone methods and performance impactful changes**. Disponível em: <<https://repositorio.udesc.br/entities/publication/33ef5ed9-ce0d-4546-9b1c-92a129486100>>. Acesso em: 18 ago. 2026.

GALIMBERTI, Maurício Floriano; WAZLAWICK, Raul Sidnei. Active Internationalization of Small and Medium - Sized Software Enterprises - Cases of French Software Companies. **Journal of technology management & innovation**, v. 10, n. 4, p. 99–108, dez. 2015.

JOHANSEN, Martin Fagereng *et al.* A Technique for Agile and Automatic Interaction Testing for Product Lines. *In*: NIELSEN, Brian; WEISE, Carsten (Orgs.). **Testing Software and Systems**. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. v. 7641 p. 39–54.

KANG, Kyo *et al.* Feature-Oriented Domain Analysis (FODA) feasibility study. 1 jan. 1990.

LINDEN, Frank J. van der; SCHMID, Klaus; ROMMES, Eelco. **Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering**. [S.l.]: Springer Science & Business Media, 2007.

MENDONÇA, Willian Douglas Ferrari; ASSUNÇÃO, Wesley K. G.; VERGILIO, Silvia. Feature-oriented Test Case Prioritization Strategies: An Evaluation for Highly Configurable Systems. *In*: : SPLC '24. New York, NY, USA: Association for Computing Machinery, 2 set. 2024. Disponível em: <<https://dl.acm.org/doi/10.1145/3646548.3672592>>. Acesso em: 25 ago. 2026

STREY, Kevin G.; MENDONÇA, Willian D. F.; FARAH, Paulo Roberto. Extração e Visualização de Métricas Evolutivas em LPS: um Estudo de Caso com o Eclipse IDE. *In*: 14TH WORKSHOP ON SOFTWARE VISUALIZATION, EVOLUTION AND MAINTENANCE (VEM26). **São Paulo**. 2026. Disponível em: <<https://vemworkshop.github.io/vem2026/acceptedPapers.html>>. Acesso em: 26 ago. 2026