ANÁLISE DE CARACTERÍSTICAS DE SISTEMA LEGADO PARA EVOLUÇÃO DA MOCKUPTOME DSL

Jean Trindade Piagetti*
Prof. Dr. Fábio Paulo Basso**

RESUMO

A evolução constante de *frameworks* e APIs para o desenvolvimento de software web e mobile pode aumentar a complexidade para equipes de desenvolvimento que desejam migrar de uma opção para outra. Para tornar o processo de desenvolvimento mais acessível e independente das habilidades técnicas específicas, foram propostas abordagens de *Model-Driven Web Engineering (MDWE)*, também conhecidas como *Model-Driven Development (MDD)* e *Model-Driven Engineering (MDE)*. Neste estudo, analisamos uma ferramenta de suporte para MDE, chamada *MockupToME DSL*. Tal ferramenta é discutida na ótica de sua evolução como um sistema legado, uma vez que sua última atualização data de mais de dez anos. Para a condução do trabalho de conclusão de curso 1, realizou-se uma revisão da literatura para identificar atributos de qualidade das ferramentas concorrentes de modo à contextualizar a MockupToME no estado da arte. Para a sequência, realizou-se uma análise de características de sistemas legados aplicada sobre o código fonte do software. Com base numa análise de linhas de código e na experiência do proponente com modernização de aplicações Java na indústria de software, dados sobre esforço de evolução são obtidos, gerando como resultado uma base de conhecimento para análise crítica como recomendação sobre o uso futuro da ferramenta: ou se alinha uma evolução para que a MockupToME siga como objeto de novas pesquisas; ou então se encaminha a sua aposentadoria perante sua degradação arquitetural em comparação com o estado da arte.

Palavras-chaves: MDE. MDWE. DSL.Domain Specific Language. Software Evolution. Software Retirement.

ABSTRACT

The constant evolution of frameworks and APIs for web and mobile software development can increase complexity for development teams wishing to migrate from one option to another. To make the development process more accessible and independent of specific technical skills, approaches such as Model-Driven Web Engineering (MDWE), also known as Model-Driven Development (MDD) and Model-Driven Engineering (MDE), have been proposed. In this study, we analyze a support tool for MDE called "MockupToME DSL."This tool is discussed from the perspective of its evolution as a legacy system, given that its last update dates back more than ten years. Previouslly, we conducted a literature review to identify quality attributes of competing tools in order to contextualize MockupToME in the state of the art. Then, in this paper, we presented an analysis of legacy system characteristics applied to MockupToME source

*Aluno do Curso de Engenharia de Software da Universidade Federal do Pampa, Alegrete, Rio Grande do Sul, Brasil
E-mail: jeanpiagetti.aluno@unipampa.edu.br
**Orientador, Professor do Curso de Engenharia de Software da Universidade Federal do Pampa, Alegrete, Rio Grande do Sul, Brasil, E-mail: fabiobasso@unipampa.edu.br

code. Based on an analysis of lines of code and the proponent's experience with Java application modernization in the software industry, data on evolution effort is obtained, resulting in a knowledge base for critical analysis and recommendations regarding the future use of the tool: either an evolution path is aligned for MockupToME to serve as an object of further research, or its retirement is recommended due to its architectural degradation compared to the state of the art.

Keywords: MDE. MDWE. DSL.Domain Specific Language. Software Evolution, Software Retirement.

1. INTRODUÇÃO

O desenvolvimento web moderno se tornou complexo devido ao alto número de *frameworks* disponíveis no mercado, visando aumentar a produtividade. No entanto, esses *frameworks* acabam aumentando a complexidade dos algoritmos, o que resulta em projetos mais caros e um time de desenvolvimento menos produtivo (CUNHA; MOURA, 2007). Neste contexto, a busca por soluções para otimizar o desenvolvimento de software através do uso de ferramentas, como auxílio para o desenvolvedor e/ou a equipe de desenvolvimento desde a concepção do software até sua implantação no cliente, vem sendo encorajada desde muito tempo (HOU; WANG, 2009).

Estudos realizados na década de 80 identificaram problemas comuns nos softwares, tais como a falta de qualidade do código, problemas na análise de requisitos e especificação do software. Para solucionar esses problemas, tornou-se necessário o uso de ferramentas CASE (WEINRICH, 1999). Além disso, surgiram as *Domain Specific Language*, que tornam o desenvolvimento de aplicações mais específicas, aumentando o nível de abstração (JETBRAINS S.R.O.,) e a eficiência no desenvolvimento da aplicação do domínio específico, pois através dos transformadores é gerado o código fonte da aplicação. Recentes abordagens de desenvolvimento de software propostas com base em transformação de modelos independentes de plataforma em modelos específicos de plataforma vem sendo utilizadas em fábricas de software para gerar o código de aplicações (RUSCIO et al., 2022).

Com este mesmo propósito, a *MockupToME DSL* (BASSO et al., 2016) foi desenvolvida. Este software promove a especificação dos modelos do domínio da aplicação e oferece suporte ao refinamento dos modelos até a geração do código fonte. A *MockupToME DSL* é uma abordagem para o *Rapid Application Development*(RAD) e inclui o processo de transformação de modelos flexível, conhecido como *MockupToME Method*. O processo de prototipação da abordagem é dividido em três níveis: Evolutiva, Arquitetural e Funcional. A troca de arquitetura de desenvolvimento ocorre no nível de prototipação arquitetural, enquanto a geração de código é realizada na prototipação funcional. Uma demonstração de menos de dois minutos está disponível em <https://youtu.be/TrjuqLJMy8M>.

Este TCC apresenta um estudo que aborda tais temáticas com o objetivo de identificar o estado da arte no MDE. Nessa temática, a *MockupToME DSL* permite na prototipação funcional a geração de código para plataformas alvo baseada numa arquitetura em três camadas.

1.1. MOTIVAÇÃO

Durante o desenvolvimento do TCC 1, foi proposto um novo recurso para a *MockupToME DSL* com o objetivo de gerar código para a camada de Serviços de uma aplicação MVC. No entanto, antes de fazer essa modernização, era necessário garantir que a *MockupToME DSL* ainda fosse considerada um software de ponta. Como se trata de um software antigo, com sua última manutenção em 2012 e limitações de execução apenas em ambiente *Desktop*, foram exploradas alternativas para modernizar as *APIs* desatualizadas da ferramenta. No entanto, a identificação de dependências problemáticas para a implantação, execução e implementação de novos geradores de código para a camada de serviço foi um obstáculo para a proposta inicial.

Paralelamente a esse estudo, um estudo exploratório de tecnologias de *Domain Specific Language (DSL)* foi realizado pelo grupo de pesquisa (FAVERO et al., 2018), o que levou à identificação da necessidade de evolução da *MockupToME DSL* para tecnologias mais modernas baseadas em serviços, como Software as a Service (SaaS) (ROCCO et al., 2015). Embora tenha sido reconhecido que a modernização da *MockupToME DSL* para operar em *SaaS* poderia ser dispendiosa, não foram coletados dados sobre custo e esforço para essa modernização. Devido à *MockupToME* ter sido mencionada em duas revisões de literatura como um software de ponta para o desenvolvimento web orientado a modelos, demonstrando potencial para modernização, torna-se essencial realizar um estudo de análise de esforço para fornecer dados que auxiliem os pesquisadores na decisão de modernizar ou aposentar o software.

Esse desafio de pesquisa caracteriza uma manutenção adaptativa. De acordo com o SWEBOK(BOURQUE; FAIRLEY, 2021) (no Capítulo 5), manutenção adaptativa é a modificação de um produto de software de modo à mantê-lo utilizável em um ambiente alterado ou em constante mudança.

1.2. OBJETIVOS

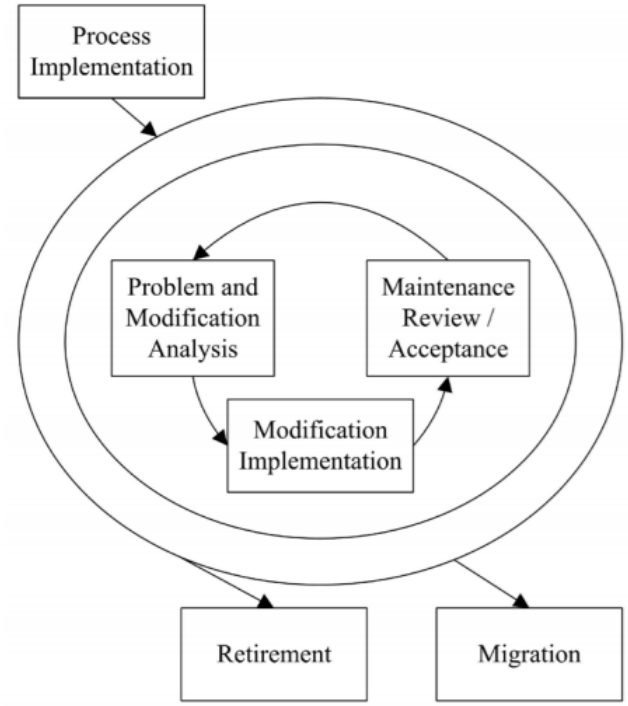
O objetivo deste estudo é analisar um sistema legado para identificar o potencial de modernização e preparar material de suporte para a modernização da MockupToME DSL. O objetivo final é tornar o processo de desenvolvimento de software mais acessível e independente das habilidades técnicas específicas, através de abordagens de Model-Driven Web Engineering.

1.3. JUSTIFICATIVA

A comunidade científica está empenhada em desenvolver ferramentas de suporte que facilitem o processo de desenvolvimento de software usando abordagens orientadas por modelos. No entanto, paradoxalmente, na área de manutenção de software, muitas vezes não exploramos adequadamente o suporte ferramental existente, o que dificulta a identificação de melhorias futuras e a superação de obstáculos para o uso de determinado software. Decidir se devemos evoluir um software por meio de migração ou abandoná-lo completamente faz parte dos processos de manutenção, conforme ilustrado na Figura 1. Essa decisão é baseada em informações relevantes sobre o potencial do produto para o público-alvo.

Para realizar uma análise decisória para a evolução ou aposentadoria da *MockupToME DSL*, este TCC apresenta um estudo de análise estática de código-fonte. Nesta

Figura 1 - As duas possíveis saídas de um estudo analítico de um processo de manutenção para a *MockupToME DSL*



SWEBOK v3

proposta, o foco é aplicar atividades preliminares de manutenção, que permitam identificar o potencial de modernização de um software legado, e tendo potencial, delinear material de estimativa de esforço para conduzir as atividades futuras de evolução.

1.4. ORGANIZAÇÃO DESTE TRABALHO

Este trabalho está organizado da seguinte forma. Na Seção 2 é realizada a fundamentação teórica, que aborda os fundamentos e conceitos que regem este trabalho. Já na Seção 3 é feita uma avaliação da *MockupToME*, no Seção 4 são feitas as considerações finais onde serão discutidos alguns pontos do trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Uma vez que o foco do estudo é dirigido para identificar contribuições relacionadas com a camada de serviços, esta seção abordará todos os fundamentos teóricos que foram utilizados neste trabalho, descrevendo principalmente conceitos de MDE e SOA.

2.1. PROTOTIPAÇÃO RÁPIDA DE APLICAÇÕES WEB

Temos visto cada vez mais esforços para tornar o conteúdo presente na internet acessível aos usuários finais. Porém, mesmo com todos estes esforços, poucos programadores e equipes de desenvolvimento realmente se preocupam em tornar seus web-sites acessíveis em múltiplas plataformas (BRAMBILLA; FRATERNALI, 2014; RIVERO et al., 2017).

Por exemplo, além do conhecimento técnico necessário para desenvolvimento de sistemas de informação (BASSO et al., 2016), desafios para incorporar em aplicativos as novidades do mercado incluem, principalmente, a dificuldade das equipes de

desenvolvimento de cumprir com o cronograma em iterações de um processo de desenvolvimento cada vez mais curtas (OLIVEIRA et al., 2018).

Por conta disso, importantes características de um sistema web como acessibilidade, facilidade de uso e navegabilidade acabam ficando de lado em prol de cumprir com o escasso tempo de mercado. Neste sentido, ferramentas para Prototipação Rápida de Aplicações (RAP) (ELKOUTBI; KHRISS; KELLER, 2006; PLANAS; CABOT; GÓMEZ, 2009) podem contribuir para superar este desafio.

2.2. Principais Conceitos da Temática de Estudo

No desenvolvimento de sistemas de informações da web, os front-ends da web como o layout são compostos pelos componentes para GUI (Graphical User Interface) (MIJAILOVIć; MILIćEV, 2014) e por diagramas comportamentais (NUNES; SCHWABE, 2006). Para permitir a geração de código-fonte completo com uma abordagem MDWE, esses modelos são decorados com a semântica para ações do usuário, fluxos de tela e lógica de negócios. Assim, usando-se uma DSL apropriada, é possível abstrair detalhes de implementação, focando na especificação de semântica em modelos que formalizam o conhecimento sobre requisitos de software (FRANCE; BIEMAN, 2001).

O MDWE permite o desenvolvimento de sistemas de informação para múltiplas plataformas por meio de transformações de modelos (VARA; ESPERANZA, 2012). Outro emprego comum, por exemplo, é no desenvolvimento de serviços que integram múltiplos sistemas (mashups), usando uma arquitetura orientada a serviço (SOA) e também micro-serviços (VRIEZE et al., 2011). Por fim, outro bom exemplo está na automação de processos de negócios, realizada por meio de modelos representados com BPMN (PILLAT et al., 2015).

Uma ampla variedade de abordagens para MDWE foram propostas como produtos derivados de pesquisa desde 2000 (CERI; FRATERNALI; BONGIO, 2000). Atualmente, a falta de um mapeamento das propostas existentes gera uma incerteza sobre o que cada ferramenta oferece para a indústria de software, bem como quais são as lacunas de pesquisa. Assim, este estudo apresenta um mapeamento de propostas para MDWE que introduzem elementos representacionais para o design de GUIs que levam à geração de protótipos. Este (BASSO; URI; PIAGETTI, 2018) é um novo mapeamento, já que estudos existentes focam em plataformas específicas como AJAX (MESBAH; DEURSEN, 2008), com análise de propostas e desafios sobre *frameworks* mais utilizados. Diferentemente, nosso estudo foca em tecnologia existente para gerar código específico de plataforma por meio de modelos independentes delas.

2.3. Conceitos do Objeto de Estudo - *MockupToME* DSL

A *MockupToME DSL* foi criada com base nas técnicas de *design* e MDE, que utilizam modelos de especificação para o desenvolvimento do software propriamente dito. Esses modelos são carregados com semântica o que dá suporte para a geração de código. Quando o modelo é carregado na DSL ele é interpretado pelos transformadores de código e por sua vez o transformam em linguagem de programação. Para cada linguagem de programação há um transformador específico, a DSL em questão faz a transformação para linguagem *Java*. A DSL também faz o uso de prototipações de interface de usuário e a modelagem da aplicação utilizando um ambiente de modelagem UML, que por sua vez faz a geração de código fonte com o padrão DAO (BASSO et

al., 2016). Ao final desse processo temos um software que com toda lógica gerada e a comunicação com banco de dados e todo o fluxo das interfaces mapeados na etapa de design.

Algumas destas ferramentas permitem, por meio de modelos, a geração de sistemas conforme múltiplas plataformas (VARA; ESPERANZA, 2012). Para tanto, MDE (KENT, 2002) é um paradigma para o desenvolvimento de software baseado em transformações de modelos, e vem sendo implementado com algumas técnicas de design que contribuem para a RAP.

Em processos típicos baseados em MDE, as transformações de modelo devem receber um modelo altamente detalhado para gerar partes funcionais de aplicativos (SCHMIDT, 2006). Para a geração de código-fonte completos (KELLY; TOLVANEN, 2008), várias partes de um design de aplicativo são detalhadas em DSLs (Domain Specific Languages) (VOELTER, 2009) e/ ou decoradas com anotações adicionadas aos elementos de modelo representados com a UML (Unified Modeling Language) (ELKOUTBI; KHRISS; KELLER, 2006), uma linguagem de modelagem de propósito geral usada em projetos arquiteturais de alguns sistemas web.

Em qualquer caso, isso torna a construção do software dependente de tarefas de design. Neste contexto, nosso objetivo foi descrever 13 anos de pesquisa no design de front-end web focando em Model-Driven Web-Engineering (MDWE). MDWE inclui desde abordagens para arquitetura de redes de computadores ao desenvolvimento de aplicativos web (BRAMBILLA; FRATERNALI, 2014).

Assim, focamos este estudo nas DSLs propostas para design de front-ends web e nas abordagens que automatizam o design por meio de transformações de modelos (BATORY; LATIMER; AZANZA, 2013), principalmente as direcionadas para a geração da camada de serviços.

2.4. MIGRAÇÃO

De acordo com o o SWEBOK v3 (BOURQUE; FAIRLEY, 2021), a migração de um sistema para algo mais moderno deve ser decidida em um processo de análise rigoroso. Esse processo envolve uma atividade caracterizada como notificação de intenção, que é uma declaração do analista do porquê o ambiente antigo não deve mais ser suportado, seguido por uma descrição do novo ambiente e sua data de disponibilidade. No foco deste TCC, essa atividade diz respeito ao abandono do framework WCTSample em detrimento de outra configuração baseada, preferencialmente, em tecnologias associadas com Microserviços.

(FRANCESCO; LAGO; MALAVOLTA, 2018) recentemente executou um survey na industria para identificar as principais dificuldades dos desenvolvedores na migração de um sistema para microserviços. Ele separou a sua análise das lacunas de pesquisa em um modelo em ferradura que envolve três etapas: engenharia reversa, transformação e engenharia avançada, Esta é a produção de código para uma arquitetura distribuída em microserviços, seja o desenvolvimento manual ou assistido por ferramental. Como mostra a Figura 2, dentre os principais desafios apontados pelos profissionais da prática, pode-se destacar:

a) Configurar a infraestrutura inicial para que os microsserviços funcionem.

b) Compartilhamento de conhecimento numa comunicação eficaz entre desenvolvedor e manutenedor.

Figura 2 - Principais Desafios para a Engenharia Avançada

Challenges	Occurrences
Setting up the initial infrastructure for microservices to work	9
Different thinking for developers	7
Distributed monitoring	6
Knowledge sharing, effective communication	6
Distributed logging	5
Distributed debugging	5
Create uniformity across services	5
Testing the new system	4
Get the initial team to work together	4
Using standards and norms	2
Get the initial prototype working	1
Other	6

Fonte:(FRANCESCO; LAGO; MALAVOLTA, 2018)

2.5. MÉTODOS DE CONTAGEM

Existem várias formas de contagem de software, sendo as mais comuns:

Linhas de código-(LOC) - Técnica de medição por linhas de código, uma das mais antigas formas de medição de software, consiste na contagem das linhas de código do software desenvolvido. Além de ser muito simples, é possível criar rotinas de automação, porém a dependência da linguagem de código e o desenvolvedor se torna uma desvantagem, sendo viável nas etapas de codificação.

Métricas de Hasltead - Métricas proposta por (HALSTEAD, 1977). É uma técnica de medição de software que visa quantificar a complexidade e o tamanho do software por meio da análise de métricas como volume, complexidade, número de palavras-chave, entre outras. Essas métricas são obtidas a partir da contagem de elementos do código-fonte, como linhas, palavras e operadores. O resultado da análise de métricas de Hasltead é útil para avaliar a qualidade do software e identificar possíveis pontos de melhoria.

2.6. ESTUDOS DE ENGENHARIA DE SOFTWARE EXPERIMENTAL

Para realizar uma notificação de intenção, seja pela migração ou pela aposentadoria de um sistema legado, uma alternativa é identificar o potencial científico da *MockupToMDE DSL* para ensino, pesquisa ou extensão. Para a pesquisa, o ideal é a realização de estudos de Mapeamento ou Revisão Sistemático de Literatura (SMS) (KITCHENHAM; CHARTERS, 2007). Tais estudos consistem basicamente em uma investigação para examinar e identificar lacunas de pesquisa, dando uma visão geral sobre o estado da arte e elencando os principais trabalhos publicados na área. Outra alternativa, é testar o software na prática, também fazendo estudos de caso ou experimentos (WOHLIN et al., 2012). Tais alternativas também podem ser utilizadas para avaliar possíveis propostas para superar dois desafios na migração: Configurar

a infraestrutura inicial e o compartilhamento de conhecimento, no que é denominado como transferência de conhecimento.

2.7. O Estado da Arte em Design de Front-End Web em MDWE

A Tabela 1 apresenta os artigos encontrados em ordem cronológica. Na condução do estudo, após uma revisão de literatura ad-hoc, executou-se a primeira atividade incluindo manualmente os artigos {S01-S04, S08-S11, S22-S27, S30-S32 e S34}. Ou seja, estes foram incluídos antes da execução das buscas nas bases elencadas e, portanto, caracterizando uma "adição por heurística".

Em seguida, as bases foram consultadas, os pacotes com referências gerados e o filtro dos artigos por *"keyword"* e leitura do *"abstract"* foi executado. Também por "adição por heurística", os artigos {S14, S19, S36 e S38} foram inseridos de um modo conveniente, durante a formação do pacote de leitura, utilizando-se das recomendações das bases após a busca de cada artigo individualmente para download.

Tabela 1 - Propostas para design, refinamento e geração de front-end web no MDWE

Id	Título e Referência para o Estudo	Ano
S01	A mda-compliant environment for developing user interfaces of information systems (VANDERDONCKT, 2005)	2005
S02	Rapid prototyping of web applications combining domain specific languages and model driven design (NUNES; SCHWABE, 2006)	2006
S03	Automated prototyping of user interfaces based on uml scenarios (ELKOUTBI; KHRISS; KELLER, 2006)	2006
S04	A model-driven approach to generating user interfaces (KAVALDJIAN, 2007)	2007
S05	A uml profile for modeling framework-based web information systems (SOUZA; FALBO; GUIZZARDI, 2007)	2007
S06	Model-Based development of user interfaces a new paradigm in useware engineering (ZUEHLKE, 2007)	2007
S07	A component-and push-based architectural style for ajax applications (MESBAH; DEURSEN, 2008)	2008
S08	A model-driven development for gwt-based rich internet applications with ooh4ria (MELIA et al., 2008)	2008
S09	A language for high-level description of adaptive web systems (SADAT-MOHTASHAM; GHORBANI, 2008)	2008
S10	Wysiwyg development of data driven web applications (YANG et al., 2008)	2008
S11	Oblivious integration of volatile functionality in web application interfaces (GINZBURG; ROSSI; URBIETA, 2009)	2009
S12	A model-driven approach to building modern semantic web-based user interfaces (CHAVARRIAGA; MACÍAS, 2009)	2009
S13	Model-driven development of composite context-aware web applications (KAPITSAKI et al., 2009)	2009
S14	From Mockups to User Interface Models: An Extensible Model Driven Approach (RIVERO et al., 2010)	2010
S15	Generating blogs out of product catalogues: An mde approach (DÍAZ; VILLORIA, 2010)	2010
S16	Transformation templates: adding flexibility to model-driven engineering of user interfaces (AQUINO; VANDERDONCKT; PASTOR, 2010)	2010
S17	A DSL toolkit for deferring architectural decisions in DSL-based software design (ZDUN, 2010)	2010
S18	Specification of personalization in web application design (GARRIGóS; GOMEZ; HOUBEN, 2010)	2010
S19	Using HCI Patterns within the Model-Based Development of Run-Time Adaptive User Interfaces (SEISSLER; MEIXNER; BREINER, 2010)	2010
S20	Building enterprise mashups (VRIEZE et al., 2011)	2011
S21	Towards model-driven development of access control policies for web applications (BUSCH et al., 2012)	2012
S22	A framework for model-driven development of information systems: Technical decisions and lessons learned (VARA; ESPERANZA, 2012)	2012
S23	Towards agile model-driven web engineering (RIVERO et al., 2012)	2012
S24	Ciat-gui: A mde-compliant environment for developing graphical user interfaces of information systems (MOLINA et al., 2012)	2012
S25	From requirements to web applications in an agile model-driven approach (GRIGERA et al., 2012)	2012
S26	Visually modelling data intensive web applications to assist end-user development (DEUFEMIA; D'SOUZA; GINIGE, 2013)	2013
S27	Teaching model driven engineering from a relational database perspective (BATORY; LATIMER; AZANZA, 2013)	2013
S28	Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering (RIVERO et al., 2014)	2014
S29	Seamless composition and reuse of customizable user interfaces with Spec (RYSEGHEM; DUCASSE; FABRY, 2014)	2014
S30	Assisted tasks to generate pre-prototypes for web information systems (BASSO et al., 2014)	2014
S31	Large-scale model-driven engineering of web user interaction: The webml and webratio experience (BRAMBILLA; FRATERNALI, 2014)	2014
S32	Combining mde and scrum on the rapid prototyping of web information systems (BASSO et al., 2015)	2015
S33	A model-driven development for creating accessible web menus (ANTONELLI; SILVA; FORTES, 2015)	2015
S34	Automated design of multi-layered web information systems (BASSO et al., 2016)	2016
S35	An approach to build xml-based domain specific languages solutions for client-side web applications (CHAVARRIAGA; JURADO; DÍEZ, 2017)	2017
S36	DataMock: An Agile Approach for Building Data Models from User Interface Mockups (RIVERO et al., 2017)	2017
S37	Application of Kroki Mockup Tool to Implementation of Executable CERIF Specification (FILIPOVIć et al., 2017)	2017
S38	BRCode: An interpretive model-driven engineering approach for enterprise application (OLIVEIRA et al., 2018)	2018

2.8. Discussão sobre os Temas dos Mapeamentos Sistemáticos de Literatura

Nos mapeamentos de literaturas foram explorados estudos que discutem a flexibilidade das abordagens de desenvolvimento orientadas a modelos, especialmente em relação à sua aplicação em larga escala na indústria. Alguns estudos utilizam a combinação de BPMN *(Business Process Model and Notation)* com MDE para realizar

transformações. No entanto, é destacado que, quando se trata do cenário industrial, existem desafios, como a falta de estabilidade linear nas regras de negócio, que estão em constante variação, e a limitação de flexibilidade de algumas ferramentas nesse aspecto. Além de avaliar abordagens para MDWE, com foco no desenvolvimento de camadas de aplicativos web por meio de Domain-Specific Languages (DSLs) dedicadas a recursos estruturais para o *front-end*. Foram apresentadas várias ferramentas propostas para auxiliar os engenheiros de software nesse processo.

Esses mapeamentos sistemáticos de literatura fornecem uma base sólida para entender o estado atual da pesquisa nessas áreas e identificar possíveis lacunas e direções futuras para o desenvolvimento de abordagens orientadas a serviços e MDE no desenvolvimento de software.

2.9. Lições do Capítulo

A relação dos conceitos mais relevantes para este estudo foram apresentados neste capítulo. Se fez necessário a investigação dos grandes domínios sendo um deles o MDE e área de serviços. Dentre as abordagens se destaca a seção de DSL, pois traz conceitos mais importantes para o esclarecimento a respeito das DSL. Avaliação de Características de Sistema Legado Este capítulo tem por objetivo fazer a avaliação da aplicação da metodologia MockupToMe. Para realizar essa avaliação, é importante analisar a literatura existente sobre o tema, buscando referências e estudos anteriores que abordem a modernização de sistemas legados e a eficácia de abordagens automatizadas para o design de mockups.

3. Estudo Sobre Modernização da *MockupToMe*

De acordo com o trabalho de (CHERVENSKI, 2019), a *MockupToMe* pode ser considerada um sistema legado não devido ao seu tempo de uso, mas sim porque seus componentes e bibliotecas já não possuem mais suporte. A ferramenta é valiosa do ponto de vista comercial, pois suas regras são usadas em ferramentas *low-code* atuais. Dentre elas, segundo o blog CronApp

- Desenvolvimento Visual: As plataformas *low-code* oferecem interfaces de arrastar e soltar visuais que permitem que desenvolvedores criem aplicativos sem escrever código complexo. Isso facilita a construção de aplicativos, mesmo para usuários não técnicos.
- Desenvolvimento baseado em modelos: As plataformas *low-code* usam uma abordagem baseada em modelos para o desenvolvimento de software, onde o foco é na lógica de negócios e não no código. Essa abordagem permite o desenvolvimento rápido de aplicativos fáceis de manter e modificar.

No entanto, as qualidades técnicas da ferramenta não são adequadas para os padrões atuais de desenvolvimento.

Foi feita uma tentativa de atualização da infraestrutura da ferramenta, tornando-a uma biblioteca importada através do arquivo 'pom.xml', montamos uma estrutura de gerenciamento de artefatos utilizando o *Sonatype Nexus*, onde a *MockupToMe* seria armazenada e utilizada como uma dependência para projetos. No entanto, mesmo com essa abordagem, não foi possível atualizar a biblioteca devido à falta de manutenção das suas dependências.

3.1. Comparação do Potencial da *MockupToMe* com o Estado da Arte

A comparação do potencial da *MockupToMe* com o estado da arte em Design de *Front-End Web* é importante para avaliar sua eficiência e capacidade de atender às demandas do mercado. A revisão de literatura efetuada permite concluir que a *MockupToMe* apresenta características que a destacam em relação aos demais softwares de design de *front-end*, como a facilidade de uso e a rapidez na criação de protótipos. No entanto, é importante levar em conta que o estado da arte em Design de *Front-End Web* está em constante evolução, com o surgimento de novas ferramentas e técnicas, sendo necessário avaliar periodicamente o potencial da *MockupToMe* em relação aos avanços da área.

Porém, existem outras aplicações no mercado, como o *JHipster*, que utilizam modeladores de domínio *low-code* e geram artefatos completos, incluindo testes, com tecnologias modernas. Essas opções são mais viáveis e econômicas para suprir algumas de nossas necessidades atuais de desenvolvimento de software.

3.2. Recomendações Quanto à Aposentadoria ou Modernização

A partir das informações apresentadas no artigo de (CHERVENSKI, 2019), é recomendável a modernização do software *MockupToMe*. O fato de que seus componentes e bibliotecas já não possuem mais suporte torna a manutenção e atualização do software difícil e custosa. Além disso, outras plataformas *low-code*, como o *JHipster*, já utilizam tecnologias atuais e são capazes de gerar registros completos com testes unitários.

A modernização do software exigiria um alto investimento em tempo e dinheiro, pois ela é complexa e precisaria ser atualizada para as tecnologias atuais, além de replicar suas regras de negócio.

Uma possível abordagem para modernizar o software seria o desenvolvimento de uma nova versão a partir do zero, utilizando tecnologias modernas e mantendo as regras de negócio. Isso permitiria que o software atendesse aos padrões atuais e oferecesse uma melhor experiência para os usuários. Outra possibilidade seria a migração gradual do software para tecnologias mais modernas, o que reduziria o impacto financeiro e permitiria uma transição mais suave.

Existem diversas técnicas disponíveis para medição de software, e uma delas é a contagem de linhas de código (LOC) - uma das mais antigas e simples de utilizar. Por essa razão, é comum a criação de rotinas automatizadas para implementação. Além disso, a abordagem LOC pode ser útil na manutenção de projetos legados, ajudando a entender a complexidade e o tamanho do código para priorizar a alocação de recursos de maneira adequada.

3.3. Análise da complexidade da aplicação

Nesta seção mostraremos nossa análise da complexidade da aplicação e também explicar como foi feito a análise, quais indices utilizados para medição.

Escolhemos como ferramenta para análise de linhas de código o Eclipse Metrics Reloaded que é uma ferramenta *open-source* de análise de código-fonte para

o Eclipse. Ela fornece uma série de métricas de software para ajudar os desenvolvedores a avaliar a qualidade do código de um projeto.

Um dos critérios para utilizar a ferramenta possui o conjunto de métricas que são baseadas no livro de (SELLERS, 1996), tais métricas incluem, por exemplo, a complexidade ciclomática de McCabe, o número de linhas de código, o número de classes, o número de métodos, a profundidade de herança, entre outras métricas.

A ferramenta também permite que os desenvolvedores criem suas próprias métricas personalizadas usando uma linguagem de *script* específica. Além disso, é possível configurar regras de qualidade de código e estabelecer limites para as métricas. Se o código-fonte não atender a essas regras ou limites, o *Eclipse Metrics Reloaded* pode gerar avisos ou erros para indicar potenciais problemas no código. Além da facilidade de uso, basta abrir o projeto a ser medido e abrir a perspectiva de medição, após isso ele irá mostrar todas as métricas e os arquivos mais relevantes à aquela métrica.

Em resumo, o *Eclipse Metrics Reloaded* é uma ferramenta útil para analisar a qualidade do código de um projeto Java e para identificar áreas do código que precisam de atenção ou refatoração.

Métricas relacionadas à complexidade e estrutura do sistema foram analisadas. Essas métricas fornecem informações valiosas sobre diferentes aspectos do código-fonte. A seguir, estão as métricas e seu significado:

- **VG** (McCabe Cyclomatic Complexity): É uma medida de complexidade de controle de fluxo em um método.
- **PAR** (Number of Parameters): Indica o número de parâmetros em um método.
- **NBD** (Nested Block Depth): Representa a profundidade de blocos aninhados em um método.
- **CA** (Afferent Coupling) e **CE** (Efferent Coupling): Representam a quantidade de acoplamento aferente e eferente, respectivamente, em pacotes.
- **RMI** (Instability), **RMA** (Abstractness) e **RMD** (Normalized Distance): São métricas relacionadas à arquitetura e ao design do sistema.
- Métricas relacionadas às classes e tipos:
  - **DIT** (Depth of Inheritance Tree): Profundidade da árvore de herança.
  - **WMC** (Weighted Methods per Class): Número ponderado de métodos por classe.
  - **NSC** (Number of Children): Número de filhos.
  - **NORM** (Number of Overridden Methods): Número de métodos sobrescritos.
  - **LCOM** (Lack of Cohesion of Methods): Falta de coesão entre métodos.
  - **NOF** (Number of Attributes): Número de atributos.
  - **NSF** (Number of Static Attributes): Número de atributos estáticos.
  - **NOM** (Number of Methods): Número de métodos.
  - **NSM** (Number of Static Methods): Número de métodos estáticos.

  - **SIX** (Specialization Index): Índice de especialização.
  - **NOC** (Number of Classes): Número de classes.
  - **NOI** (Number of Interfaces): Número de interfaces.

- **NOP** (Number of Packages) e **TLOC** (Total Lines of Code): São métricas relacionadas à estrutura geral do sistema. **NOP** indica o número de pacotes no sistema, enquanto **TLOC** indica o número total de linhas de código.
- **MLOC** (Method Lines of Code): Indica o número de linhas de código em cada método.

Essas métricas fornecem *insights* sobre diferentes aspectos da complexidade e estrutura do sistema, auxiliando na compreensão da qualidade e manutenibilidade do código-fonte.

3.4. ANÁLISE DE CÓDIGO

Para se ter uma noção do tamanho da ferramenta, foi feita uma contagem das linhas de código totais de seus módulos principais. Os módulos apresentados na tabela são os módulos principais da ferramenta.

Projeto	**Total**
org.wct.fomda	22.474
org.wct.fomda.plugin	40.310
org.wct.mockuptome.guidsl	27.831
org.wct.sourcecodeutil.reveng	17.602
org.wct.uml.view	41.409
org.wct.sourcecodeutil.m2t.cplusplus	2.128
org.wct.sourcecodeutil.m2t.csharp	1.512
org.wct.sourcecodeutil.m2t.java	2.743
org.wct.sourcecodeutil.m2t.php	1.557
org.wct.sourcecodeutil.m2t.python	1.484
org.wct.uml.view.main	2.000
org.wct.uml.xmi.exporters.custom	351
Total	**161.501**

Tabela 2 - Total de linhas de código por projeto

Os projetos **org.wct.fomda** e **org.wct.fomda.plugin** estão diretamente relacionados aos modeladores utilizados na abordagem MockupToMe, com o objetivo de fornecer adaptabilidade e flexibilidade à metodologia de engenharia web orientada a modelos proposta pelos autores (BASSO et al., 2016). Esses projetos fazem parte da abordagem, conhecida como *FOMDA*, é caracterizada por sua flexibilidade e adaptabilidade, permitindo que os modelos sejam modificados e atualizados ao longo do processo de desenvolvimento de software. Isso possibilita o uso da abordagem *FOMDA* em diferentes contextos de desenvolvimento de software, incluindo tanto o desenvolvimento ágil quanto o desenvolvimento em cascata. Além disso, a proposta visa permitir mudanças na arquitetura e nas tecnologias subjacentes ao longo do processo de desenvolvimento.

Por outro lado, o projeto **org.wct.mockuptome.guidsl** faz parte da abordagem de design de interfaces MockupToME, que possibilita trabalhar com quatro níveis de abstração de artefatos associados às interfaces de usuário. Esses níveis incluem o protótipo em papel, o modelo de mockup, os modelos de GUI concretos específicos da plataforma e o protótipo funcional.

O projeto **org.wct.sourcecodeutil.reveng** refere-se à abordagem de Engenharia Reversa, que é frequentemente utilizada para obter informações sobre um sistema legado, como sua arquitetura, design e implementação. Essas informações são obtidas com o intuito de realizar atividades como manutenção, reengenharia ou migração para uma nova plataforma.

Os projetos **org.wct.uml.view** estão relacionados às abordagens na camada de apresentação em uma arquitetura de software da MockupToME. É por meio dela que ocorre a interação do usuário com os modelos.

Por fim, os projetos **org.wct.sourcecodeutil.m2t** referem-se aos transformadores Model-to-Text, que são ferramentas utilizadas para gerar automaticamente código-fonte a partir de modelos. Esses transformadores são usados em conjunto com a abordagem *FOMDA* para automatizar a geração de código-fonte, reduzindo o tempo e o esforço necessários para implementar um sistema de software. Além disso, eles possibilitam a geração de código-fonte para diferentes plataformas, como dispositivos móveis e web.

O projeto **org.wct.uml.xmi.exporters.custom** é responsável por exportar dados em formatos específicos, como XML, CSV, JSON ou qualquer outro formato desejado. Essa funcionalidade é útil para extrair informações de um sistema e convertê-las em um formato adequado para uso em outros sistemas.

3.5. MÉTRICAS DE CÓDIGO

Esta seção tem o objetivo de exibir as métricas do código e analisar os dados relacionados a ele, a fim de obter *insights* sobre a qualidade e desempenho do software. Isso permite identificar áreas que requerem melhorias, identificar possíveis problemas de manutenção e tomar decisões informadas sobre a evolução do código.

Durante essa análise, serão consideradas diversas métricas, como complexidade do código, tamanho do código, acoplamento entre módulos, entre outras. Essas métricas fornecem uma visão quantitativa do código e ajudam a identificar possíveis problemas, como código complexo e desafios de manutenções, dependências excessivas entre os módulos.

A Figura 3 mostra um gráfico com os valores médios de complexidade VG dos projetos. Analisando a média de complexidade, observamos uma variedade de níveis de complexidade. A média de complexidade é de aproximadamente 5,814. Essa média serve como uma referência geral para a complexidade média dos projetos.

Os valores de complexidade VG variam significativamente entre os projetos, abrangendo uma faixa ampla. Enquanto alguns projetos possuem uma complexidade relativamente baixa, como "uml.view.main"com um valor de 1,494, outros apresentam uma complexidade mais alta, como "org.wct.sourcecodeutil.reveeng"com 11,833.

A Figura 4 apresenta dados de valores médios de acoplamento. Tais dados são avaliados através das métricas de *Afferent Coupling* (CA) e *Efferent Coupling* (CE).

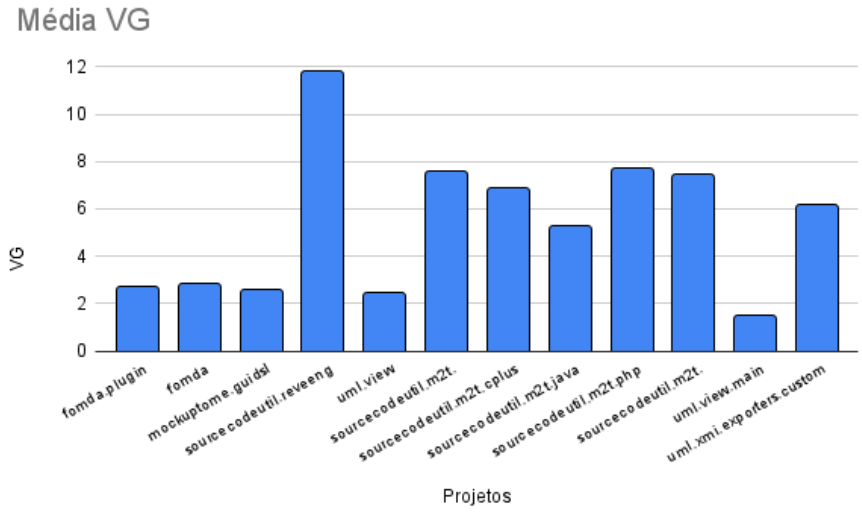


Figura 3 - Média McCabe Cyclomatic Complexity(VG)

A maioria dos projetos possui um alto acoplamento entre componentes, indicado pelos valores de "CE"(*Efferent Coupling*) relativamente altos. Isso sugere que há uma interdependência significativa entre os diferentes componentes desses projetos.

Alguns projetos têm um alto número de componentes acoplados, conforme indicado pelos valores de CA (*Afferent Coupling*) relativamente altos. Isso pode implicar em uma maior complexidade e dificuldade de manutenção desses projetos.

Alguns projetos têm um acoplamento baixo ou mesmo nulo, com valores de CA próximos ou iguais a zero. Isso pode ser positivo em termos de modularidade e facilidade de manutenção, uma vez que as alterações em um componente específico não afetariam outros componentes.

A Figura 5 apresenta dados de estabilidade (RMI - *Relative Maintenance Impact*) e abstração (RMA -*Relative Modularity Architecture*) dos projetos.

Parte dos projetos apresentam um RMI relativamente baixo, indicando um impacto moderado nas atividades de manutenção. Isso sugere que as modificações e atualizações nesses projetos podem ser realizadas de forma mais controlada e com menor risco de efeitos indesejados.

Quanto à abstração (RMA), a maioria dos projetos também possui valores relativamente baixos, o que indica uma menor modularidade e maior complexidade em sua arquitetura. Isso pode tornar mais difícil entender, modificar e reutilizar partes específicas dos projetos.

Alguns projetos apresentam um RMI próximo de 1, o que significa que qualquer modificação ou manutenção pode ter um impacto significativo em todo o sistema. Isso pode indicar uma maior dependência entre os componentes e uma maior dificuldade em realizar alterações sem efeitos colaterais.

Em relação à abstração (RMA), alguns projetos têm valores próximos de 0, o que sugere uma arquitetura mais modular e bem dividida em componentes independentes. Isso facilita a compreensão, manutenção e reutilização desses projetos.

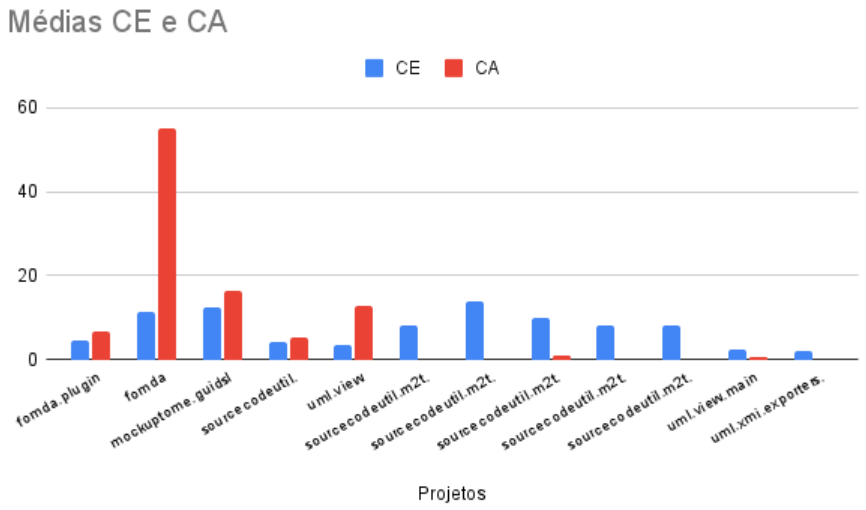


Figura 4 - Média CE e CA

A Figura 6 apresenta dados referente a herança e reutilização através da métrica *Depth of Inheritance Tree* (DIT) dos projetos. A maioria dos projetos possui valores relativamente altos, o que indica um potencial significativo para herança e reutilização de código. Isso sugere que esses projetos podem ser estruturados de maneira a permitir a criação de subclasses e a reutilização de componentes em diferentes contextos.

Alguns projetos apresentam valores mais baixos em relação à herança e reutilização. Isso pode indicar que esses projetos possuem uma estrutura mais linear, com menos oportunidades para aproveitar a herança e reutilização de código.

Projetos com valores mais altos indicam que podem existir classes ou componentes bem definidos e modularizados, que podem ser facilmente estendidos e reutilizados em outros contextos.

A Figura 7 apresenta dados referente a Tamanho e Complexidade através das métricas *Weighted Methods per Class*(WMC), *Number of Children*(NSC), *Number of Overridden Methods*(NORM), *Lack of Cohesion of Methods*(LCOM), *Number of Attributes*(NOF), *Number of Static Attributes*(NSF), *Number of Methods*(NOM),*Number of Static Methods*(NSM), *Specialization Index*(SIX). O tamanho dos projetos, medido pelo WMC (Weighted Methods per Class), varia consideravelmente, com valores que vão desde cerca de 6 a mais de 150. Quanto maior o valor do WMC, maior é a quantidade de métodos definidos nas classes do projeto.

A complexidade dos projetos também varia, com valores de LCOM (*Lack of Cohesion in Methods*) e SIX (*Specialization Index*) indicando diferentes níveis de coesão e especialização. Valores mais altos de LCOM sugerem uma menor coesão entre os métodos de uma classe, enquanto valores mais altos de SIX podem indicar uma maior especialização entre as classes.

Alguns projetos apresentam um número considerável de *namespaces*(NSC), arquivos(NOF) e métodos(NOM). Esses valores indicam uma maior estruturação e organização dos componentes nos projetos, o que pode contribuir para a manutenibilidade e escalabilidade dos sistemas.

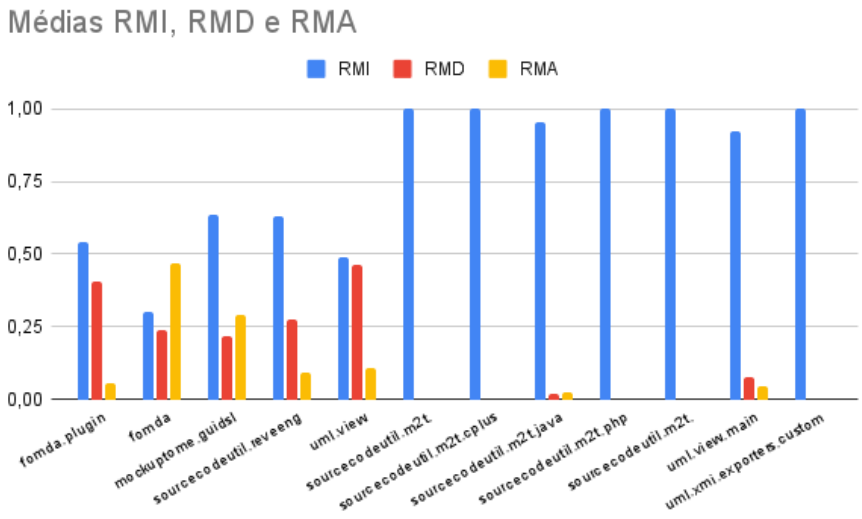


Figura 5 - Média RMI, RMD e RMA

Alguns projetos possuem um número relativamente alto de heranças (NORM), indicando a presença de classes que herdam características e comportamentos de outras classes. Isso pode contribuir para a reutilização de código e a modularidade dos projetos.

É importante ressaltar que o tamanho e a complexidade dos projetos podem afetar sua manutenibilidade, testabilidade e compreensão. Projetos com valores muito altos em WMC, LCOM e outros métricas podem ser mais difíceis de entender e modificar.

A Figura 8 mostra a Organização do Projeto através das métricas Número de classes (NOC), Número de interfaces (NOI) e Número de pacotes (NOP), sendo interpretada como segue:

- Número de classes (NOC): Os projetos têm diferentes quantidades de classes, variando de cerca de 2 a mais de 17. Isso indica o número de classes definidas nos projetos que representam os componentes principais do sistema.
- Número de interfaces (NOI): Alguns projetos possuem um número considerável de interfaces, o que sugere o uso de abstrações e contratos para definir comportamentos e interações entre os componentes.
- Número de pacotes (NOP): Os projetos possuem uma estrutura de organização em pacotes, que pode variar de 0 a aproximadamente 77. Esses pacotes são utilizados para agrupar e organizar as classes e interfaces, oferecendo uma divisão lógica dos componentes do sistema.

Essas métricas ajudam a entender como os projetos estão organizados em termos de estrutura de classes, uso de interfaces e divisão em pacotes. Uma organização clara e bem definida pode facilitar o desenvolvimento, a manutenção e a reutilização de código nos projetos.

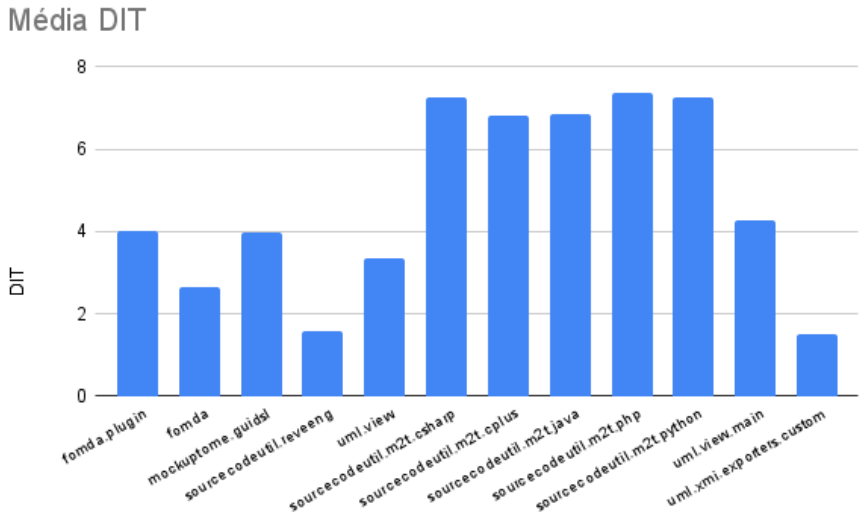


Figura 6 - Média DIT

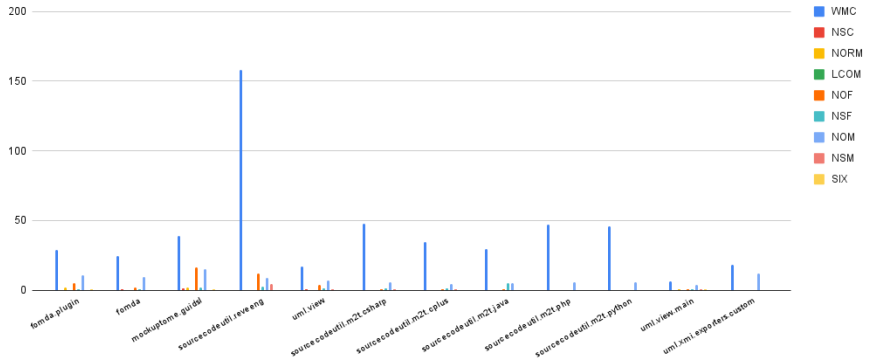


Figura 7 - Médias WMC, NSC, NORM, LCOM, NOF, NSF, NOM, NSM, SIX

3.6. FATORES QUE DIFICULTAM A EVOLUÇÃO DA FERRAMENTA

A ferramenta atualmente possui uma falta significativa de documentação, em termos técnicos de cada projeto. Isso desempenha um papel crucial na decisão de evoluir ou aposentar a ferramenta. Portanto, é um fator importante a ser considerado nesta análise.

Além disso, a curva de aprendizado da ferramenta é bastante íngreme, devido à sua capacidade de personalização em várias formas, múltiplos usos e fluxos de trabalho. Isso torna necessário investir tempo e esforço consideráveis para se tornar proficientes na sua utilização.

O código da ferramenta é altamente acoplado e complexo, como identificado nas análises realizadas. Essa complexidade dificulta trabalhar diretamente no código, uma vez que qualquer alteração requer extrema responsabilidade. O código contém trechos extensos que executam várias tarefas, o que torna a manutenção e modificação uma tarefa desafiadora.

Embora a técnica *Extract-method* possa ser aplicada em trechos menos extensos do código para separar as responsabilidades, existe o risco de gerar comportamentos

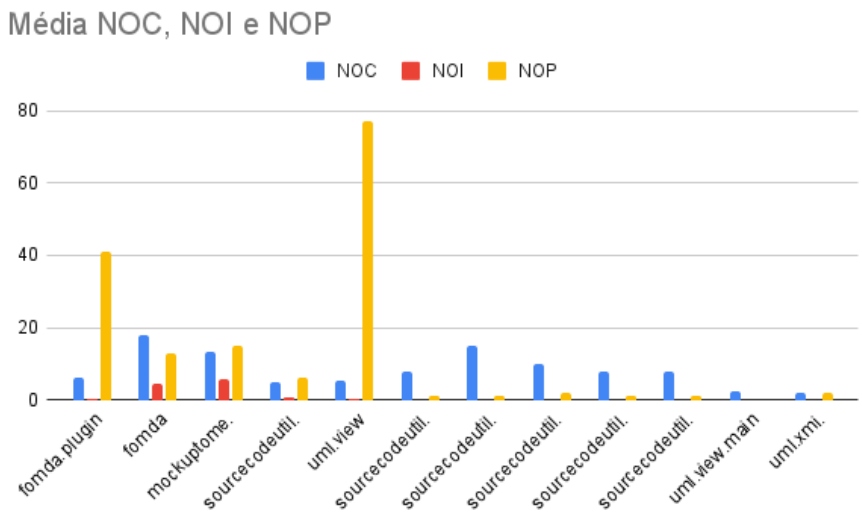


Figura 8 - Média NOC, NOI e NOP

imprevisíveis. Além disso, essa abordagem pode resultar na inutilização de módulos específicos, uma vez que o código compartilha várias ações entre eles.

3.7. Conclusões a partir das análises

De acordo com as análises conduzidas juntamente com as métricas coletadas, é possível concluir que a MockupToME apresenta algumas características que podem indicar desafios e possíveis problemas de manutenção como segue:

- Linguagem defasada: A ferramenta esta utilizando uma versão mais antiga da linguagem Java. Isso pode resultar em limitações em termos de recursos e funcionalidades disponíveis nas versões mais recentes do Java.
- Bibliotecas não mantidas: O fato de o projeto utilizar bibliotecas não mantidas pelos desenvolvedores sugere que essas bibliotecas podem estar desatualizadas e não receberem suporte e correções de *bugs*. Isso pode levar a problemas de compatibilidade e vulnerabilidades de segurança.
- Código fora dos padrões atuais: A presença de código que não está em conformidade com os padrões atuais pode dificultar a leitura, a compreensão e a manutenção do código. Isso pode levar a problemas de legibilidade, dificuldades na identificação de *bugs* e na introdução de novos recursos.

Além destes fatores, as métricas fornecidas também dão algumas indicações sobre a complexidade e a estrutura do projeto. Por exemplo, a complexidade de controle de fluxo medida pelo índice *"McCabe Cyclomatic Complexity"* tem uma média moderada, mas com uma variação considerável. A quantidade média de parâmetros por método é baixa, mas também apresenta uma variação significativa. A profundidade de blocos aninhados é moderada, com uma média razoável, mas também com variação considerável.

Essas métricas podem sugerir que o projeto possui um nível moderado de complexidade, com algumas partes do código potencialmente mais complexas que outras.

A variação nas métricas pode indicar uma falta de consistência na estrutura e na qualidade do código.

Em resumo, com base nas condições do projeto, na linguagem defasada, nas bibliotecas não mantidas e nas métricas analisadas, é possível concluir que o projeto pode exigir esforços significativos para atualização, refatoração e manutenção.

4. CONSIDERAÇÕES FINAIS

O uso de DSLs no desenvolvimento de software é motivado na literratura para otimizar o processo de desenvolvimento, melhorar a qualidade do código e aumentar a eficiência da equipe de desenvolvimento. DSLs oferecem um nível mais alto de abstração, permitindo que os desenvolvedores se concentrem mais nos requisitos do domínio específico e menos nos detalhes técnicos.No entanto, é importante ter em mente que a complexidade do código gerado por essas ferramentas pode apresentar desafios adicionais durante a manutenção e modificação do software. Portanto, é fundamental analisar o suporte ferramental existente e buscar melhorias contínuas para superar essas barreiras.

O projeto de pesquisa motivado neste TCC partia de um ponto em que era essencial identificar se a MockupToME DSL ainda apresenta potencial para a pesquisa e utilização futura, sem em ambiente acadêmico ou industrial. Neste rumo, duas revisões de literatura ajudaram à caracterizar a DSL como parte do estado da arte, apesar de ser um sistema legado. A partir daí, o foco desse trabalho foi o de preparar o ambiente e pretende para o TCC 2 gerar como contribuição a transferência de conhecimento, que hoje é atrapalhada por uma série de fatores, dentre eles a pouca documentação da MockupToME DSL e a depreciação das bibliotecas que ela é dependente em decorrência da evolução da linguagem de programação Java. Assim, o trabalho apresentado neste documento foi importante para estimar o esforço de modernização antes apenas subjetivas aos membros do grupo de pesquisa.

Com base nos mapeamentos de literatura conduzidos até o trabalho de conclusão de curso 1, chegamos à seguinte conclusão: as abordagens em MDWE, principalmente no design de componentes, são maduras em termos representativos e de semântica de ações que permitem a geração de código de qualidade. As DSLs utilizadas pelas propostas cobrem grande parte das necessidades para o design de componentes de front-ends para webapps atuais. Porém encontramos que as propostas de design automatizado, que utilizam de assistentes no refinamento de modelos, ainda apresentam boas oportunidades de pesquisa e contribuições na área, uma vez que não possuem bom aceite na indústria.

Com base na primeira revisão de literatura, que focou em abordagens de design *front-end*, buscou-se executar uma segunda revisão de literatura dirigida para o desenvolvimento de back-end em abordagens orientadas à serviço. Em especial, buscamos identificar se abordagens de MDE orientadas à dados ainda são novidades na pesquisa. Neste segundo estudo, selecionou-se um conjunto delimitado de artigos para formar a base conceitual do MDE aplicado ao designs de características de *back-end*. Identificou-se que abordagens orientadas à dados, como a MockupToME, são ainda novidades na pesquisa, e especial empregadas para o desenvolvimento de aplicações baseadas em Microserviços.

No entanto, é importante lembrar que o uso de DSLs para desenvolvimento web

pode apresentar desafios adicionais durante a manutenção e modificação do software. A complexidade do código gerado por essas ferramentas pode dificultar a compreensão do programa e a identificação de problemas. Portanto, é fundamental avaliar cuidadosamente as ferramentas disponíveis e buscar melhorias contínuas para superar essas barreiras.

Quanto à modernização da MockupToME, uma vez que o esforço de evolução é alto e que existem ferramentas em utilização atualmente com propostas equivalentes, recomenda-se que o sistema seja aposentado. Sabe-se que modernização de software legado traz benefícios importantes para as empresas, como melhoria da eficiência, qualidade e facilidade de manutenção. Além disso, permite a adoção de novas tecnologias e integração com outros sistemas. No entanto, para garantir uma modernização eficiente, é crucial investir em atividades preliminares de manutenção, como documentação e compreensão de programa, o que tornará o esforço de manutenção ainda maior do que o discutido neste documento.

REFERÊNCIAS

ANTONELLI, H. L.; SILVA, E. A. N. da; FORTES, R. P. M. A model-driven development for creating accessible web menus. **Procedia Computer Science**, Elsevier, v. 67, p. 95–104, 2015.

AQUINO, N.; VANDERDONCKT, J.; PASTOR, O. Transformation templates: adding flexibility to model-driven engineering of user interfaces. In: **2010 ACM Symposium on Applied Computing**. [S.l.: s.n.], 2010. (SAC '10), p. 1195–1202. ISBN 978-1-60558-639-7.

BASSO, F.; URI, E.; PIAGETTI, J. A summary of toolboxes scoping mdwe front-ends. In: . [S.l.: s.n.], 2018.

BASSO, F. P. et al. Assisted tasks to generate pre-prototypes for web information systems. In: **16th International Conference on Enterprise Information Systems.** [S.l.: s.n.], 2014. (ICEIS'14), p. 14–25.

BASSO, F. P. et al. Automated design of multi-layered web information systems. **Journal of Systems and Software**, v. 117, p. 612 – 637, 2016. ISSN 0164-1212.

BASSO, F. P. et al. Combining mde and scrum on the rapid prototyping of web information systems. **International Journal of Web Engineering and Technology**, v. 10, n. 3, p. 214–244, 2015.

BATORY, D.; LATIMER, E.; AZANZA, M. Teaching model driven engineering from a relational database perspective. In: **16th International Conference on Model Driven Engineering Languages and Systems**. [S.l.: s.n.], 2013. (MODELS'13), p. 121–137.

BOURQUE, P.; FAIRLEY, R. E. **SWEBOK V3 - Guide to The Software Engineering Base of Knowledge 2021**. 2021.

BRAMBILLA, M.; FRATERNALI, P. Large-scale model-driven engineering of web user interaction: The webml and webratio experience. **Science of Computer Programming**, v. 89, Part B, p. 71 – 87, 2014. Special issue on Success Stories in Model Driven Engineering.

BUSCH, M. et al. Towards model-driven development of access control policies for web applications. In: **Workshop on Model-Driven Security**. [S.l.: s.n.], 2012. (MDsec '12), p. 4:1–4:6.

CERI, S.; FRATERNALI, P.; BONGIO, A. Web modeling language (webml): a modeling language for designing web sites. **Computer Networks**, v. 33, n. 1-6, p. 137–157, 2000. ISSN 1389-1286.

CHAVARRIAGA, E.; JURADO, F.; DÍEZ, F. An approach to build xml-based domain specific languages solutions for client-side web applications. **Computer Languages, Systems & Structures**, Elsevier, v. 49, p. 133–151, 2017.

CHAVARRIAGA, E.; MACÍAS, J. A. A model-driven approach to building modern semantic web-based user interfaces. **Advances in Engineering Software**, Elsevier, v. 40, n. 12, p. 1329–1334, 2009.

CHERVENSKI, A. S. Entendimento sobre sistemas legados à luz da teoria fundamentada em dados. Universidade Federal do Pampa, 2019.

CUNHA, J. A. O. G. da; MOURA, H. P. de. Fatores que afetam a produtividade em projetos de software: Uma visão geral. 2007.

DEUFEMIA, V.; D'SOUZA, C.; GINIGE, A. Visually modelling data intensive web applications to assist end-user development. In: **6th International Symposium on Visual Information Communication and Interaction**. [S.l.: s.n.], 2013. (VINCI'13), p. 17–26.

DÍAZ, O.; VILLORIA, F. M. Generating blogs out of product catalogues: An mde approach. **Journal of Systems and Software**, Elsevier, v. 83, n. 10, p. 1970–1982, 2010.

ELKOUTBI, M.; KHRISS, I.; KELLER, R. K. Automated prototyping of user interfaces based on uml scenarios. **Automated Software Engg.**, v. 13, n. 1, p. 5–40, jan. 2006. ISSN 0928-8910. Disponível em: <http://dx.doi.org/10.1007/s10515-006-5465-5>.

FAVERO, E. et al. Estudo exploratório no refinamento de uma dsl para versões baseadas em emf, emf forms e angular. In: **Anais da II Escola Regional de Engenharia de Software**. Porto Alegre, RS, Brasil: SBC, 2018. p. 49–56. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/eres/article/view/10056>.

FILIPOVIć, M. et al. Application of kroki mockup tool to implementation of executable cerif specification. **Procedia Computer Science**, v. 106, p. 245 – 252, 2017.

FRANCE, R. B.; BIEMAN, J. M. Multi-view software evolution: A UML-based framework for evolving object-oriented software. In: **ICSM**. [S.l.: s.n.], 2001. p. 386–395.

FRANCESCO, P. D.; LAGO, P.; MALAVOLTA, I. Migrating towards microservice architectures: An industrial survey. In: **2018 IEEE International Conference on Software Architecture (ICSA)**. [S.l.: s.n.], 2018. p. 29–2909.

GARRIGóS, I.; GOMEZ, J.; HOUBEN, G.-J. Specification of personalization in web application design. **Information and Software Technology**, v. 52, n. 9, p. 991 – 1010, 2010.

GINZBURG, J.; ROSSI, D. D. G.; URBIETA, M. Oblivious integration of volatile functionality in web application interfaces. **Journal of Web Engineering**, v. 8, n. 1, p. 25–47, 2009.

GRIGERA, J. et al. From requirements to web applications in an agile model-driven approach. In: **Web Engineering**. [S.l.]: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7387). p. 200–214.

HALSTEAD, M. H. **Elements of Software Science (Operating and Programming Systems Series)**. USA: Elsevier Science Inc., 1977. ISBN 0444002057.

HOU, D.; WANG, Y. An empirical analysis of the evolution of user-visible features in an integrated development environment. **In: IBM CORP**, v. 21, n. 38, p. 122–135, 2009.

JETBRAINS S.R.O. **Linguagens específicos de domínio**. [S.l.].

KAPITSAKI, G. M. et al. Model-driven development of composite context-aware web applications. **Information and Software Technology**, v. 51, n. 8, p. 1244 – 1260, 2009. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584909000354>.

KAVALDJIAN, S. A model-driven approach to generating user interfaces. In: **The 6th Joint Meeting on European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering: companion papers**. [S.l.: s.n.], 2007. p. 603–606.

KELLY, S.; TOLVANEN, J.-P. **Domain Specific Modeling: Enabling Full Code Generation**. [S.l.]: IEEE Computer Society - John Wiley & Sons, 2008.

KENT, S. Model driven engineering. In: **Integrated Formal Methods**. [S.l.: s.n.], 2002. p. 286–298.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. 2007.

MELIA, S. et al. A model-driven development for gwt-based rich internet applications with ooh4ria. In: **Web Engineering, 2008. ICWE '08. Eighth International Conference on**. [S.l.: s.n.], 2008. p. 13–23.

MESBAH, A.; DEURSEN, A. V. A component-and push-based architectural style for ajax applications. **Journal of Systems and Software**, Elsevier, v. 81, n. 12, p. 2194–2209, 2008.

MIJAILOVIć Žarko; MILIćEV, D. Empirical analysis of gui programming concerns. **International Journal of Human-Computer Studies**, v. 72, n. 10, p. 757 – 771, 2014.

MOLINA, A. I. et al. Ciat-gui: A mde-compliant environment for developing graphical user interfaces of information systems. **Advances in Engineering Software**, v. 52, p. 10 – 29, 2012.

NUNES, D. A.; SCHWABE, D. Rapid prototyping of web applications combining domain specific languages and model driven design. In: **6th international conference on Web engineering**. [S.l.: s.n.], 2006. p. 153–160.

OLIVEIRA, A. et al. Brcode: An interpretive model-driven engineering approach for enterprise applications. **Computers in Industry**, v. 96, p. 86 – 97, 2018.

PILLAT, R. M. et al. BPMNt: A BPMN extension for specifying software process tailoring. **Information and Software Technology**, v. 57, n. 0, p. 95 – 115, 2015.

PLANAS, E.; CABOT, J.; GÓMEZ, C. Verifying action semantics specifications in uml behavioral models. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 5565 LNCS, p. 125–140, 2009.

RIVERO, J. M. et al. Datamock: An agile approach for building data models from user interface mockups. **Software & Systems Modeling**, Feb 2017.

RIVERO, J. M. et al. Towards agile model-driven web engineering. In: **IS Olympics: Information Systems in a Diverse World**. [S.l.: s.n.], 2012. v. 107, p. 142–155. ISBN 978-3-642-29748-9.

RIVERO, J. M. et al. Mockup-driven development: Providing agile support for model-driven web engineering. **Information and Software Technology**, v. 56, n. 6, p. 670–687, 2014.

RIVERO, J. M. et al. From mockups to user interface models: An extensible model driven approach. In: **Current Trends in Web Engineering**. [S.l.: s.n.], 2010. p. 13–24.

ROCCO, J. D. et al. Collaborative repositories in model-driven engineering [software technology]. **Software, IEEE**, v. 32, n. 3, p. 28–34, May 2015. ISSN 0740-7459.

RUSCIO, D. D. et al. Low-code development and model-driven engineering: Two sides of the same coin? **Softw. Syst. Model.**, Springer-Verlag, Berlin, Heidelberg, v. 21, n. 2, p. 437–446, apr 2022. ISSN 1619-1366.

RYSEGHEM, B. V.; DUCASSE, S.; FABRY, J. Seamless composition and reuse of customizable user interfaces with spec. **Science of Computer Programming**, v. 96, p. 34 – 51, 2014.

SADAT-MOHTASHAM, S. H.; GHORBANI, A. A. A language for high-level description of adaptive web systems. **Journal of Systems and Software**, Elsevier, v. 81, n. 7, p. 1196–1217, 2008.

SCHMIDT, D. C. Guest editor's introduction: Model-driven engineering. **IEEE Computer**, v. 39, n. 2, p. 25–31, 2006.

SEISSLER, M.; MEIXNER, G.; BREINER, K. Using hci patterns within the model-based development of run-time adaptive user interfaces. **IFAC Proceedings Volumes**, v. 43, n. 13, p. 477 – 482, 2010.

SELLERS, B. H. Object-oriented metrics: measures of complexity. **PH PTR, New Jersey**, 1996.

SOUZA, V. E. S.; FALBO, R. D. A.; GUIZZARDI, G. A uml profile for modeling framework-based web information systems. In: **12th International Workshop on Exploring Modelling Methods in Systems Analysis and Design EMMSAD2007**. [S.l.: s.n.], 2007. p. 153–162.

VANDERDONCKT, J. A mda-compliant environment for developing user interfaces of information systems. In: **17th international conference on Advanced Information Systems Engineering**. [S.l.: s.n.], 2005. p. 16–31.

VARA, J. M.; ESPERANZA, M. A framework for model-driven development of information systems: Technical decisions and lessons learned. **Journal of Systems and Software**, v. 85, n. 10, p. 2368 – 2384, 2012.

VOELTER, M. Best practices for dsls and model-driven development. **Journal of Object Technology**, v. 8, n. 6, p. 79–102, 2009.

VRIEZE, P. de et al. Building enterprise mashups. **Future Generation Computer Systems**, v. 27, n. 5, p. 637 – 642, 2011. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X10001974>.

WEINRICH, J. Software de apoio à avaliação e seleção de ferramentas case baseado na norma iso/iec 14102. **Trabalho de Conclusão de Curso, Universidade Regional de Blumenau. Blumenau Citado**, v. 2, 1999.

WOHLIN, C. et al. **Experimentation in Software Engineering**. [S.l.]: Springer, 2012.

YANG, F. et al. Wysiwyg development of data driven web applications. **Proc. VLDB Endowment.**, v. 1, n. 1, p. 163–175, 2008.

ZDUN, U. A dsl toolkit for deferring architectural decisions in dsl-based software design. **Information and Software Technology**, v. 52, n. 7, p. 733 – 748, 2010.

ZUEHLKE, D. Model-based development of user interfaces a new paradigm in useware engineering. **IFAC Proceedings Volumes**, v. 40, n. 16, p. 31 – 38, 2007.