Evento: XXV SEMINÁRIO DE INICIAÇÃO CIENTÍFICA

ANALISANDO A EVOLUÇÃO DAS MÉTRICAS DE MANUTENIBILIDADE DE UM SISTEMA DE SOFTWARE DE INTEGRAÇÃO: MULESOFT[1]
ANALYZING THE EVOLUTION OF MANUTENIBILITY MEASURES FROM A SOFTWARE INTEGRATION SYSTEM: MULESOFT

Rodolfo Berlezi[2], Rafael Z. Frantz[3]

[1] Resumo expandido com os resultados do tempo dedicado ao projeto de pesquisa
[2] Bolsista de Iniciação Cientifica ? PIBIC/UNIJUI, Acadêmico do Curso de Ciências da Computação e Membro do Grupo de Pesquisa em Computação Aplicada (GCA), rodolfo_berlezi@hotmail.com
[3] Orientador, Professor do Curso de Ciências da Computação, Doutor em Engenharia de Software e líder do Grupo de Pesquisa em Computação Aplicada (GCA), rzfrantz@unijui.edu.br

Introdução

Na atualidade os ambientes empresariais possuem diversas aplicações, podendo ser tanto local quanto na nuvem, além de cada setor pode ter diferentes softwares que são responsáveis pela realização e controle de suas respectivas tarefas. Esta diversidade de softwares, tanto os feitos pelas próprias empresas ou por empresas terceirizadas, criam o chamado ecossistema de software da empresa. Cada software vem com diferentes necessidades e compatibilidades e não são pensados para trabalhar em conjunto com outros programas do sistema, por isso necessitam de um meio de comunicação para a troca de informações que são incompatíveis entre eles. Existe uma área da engenharia de software conhecida como Integração de Aplicações Empresarias responsável pelo estudo e desenvolvimento de soluções de integração, e no mercado surgem as plataformas de integração especificamente feitas para desenvolver soluções de integração, tal como o Mulesoft (2014).

Segundo a IEEE (1990), existem três tipos diferente de manutenção: Corretiva, Perfectiva e Adaptativa. A manutenção corretiva é executada com a intenção de corrigir algum eventual erro ou bug do sistema de software. A manutenção perfectiva é feita com a finalidade de otimizar as funcionalidades do software. Já a manutenção adaptativa é aquela executada em ordem para remodelar o sistema de software, adicionando novas possibilidades do mesmo ser executado em diferentes ambientes ou submeter processos que não foram inicialmente configurados para o sistema, sendo esta manutenção a escolhida para avaliação no artigo.

Mulesoft é uma das ferramentas de integração de aplicações citadas no catálogo de integração de Hohpe e Woolf (2004) de código aberto, de propósito geral e concebe todos os padrões de integração considerados pelos autores. Utilizamos distintas versões da plataforma para uma análise comparativa da evolução do software. Visto que Mule é uma plataforma de código aberto e sem um contexto específico de aplicação, uma empresa poderia optar por desenvolver sua própria solução de integração adaptando uma das plataformas existentes no mercado, no caso o Mule, para uma melhor resposta a sua necessidade de integração para as suas aplicações. Assim sendo, aplicariam uma manutenção adaptativa em uma das versões do Mule. Esta pesquisa tem como objetivo demonstrar qual das versões do Mule tem o grau de manutenibilidade mais adequado.

O artigo está organizado em três grandes partes. Primeiramente a metodologia onde será explicado como foi feita a coleta dos dados e sugerir um embasamento teórico. Depois os resultados e discussões sobre o significado de cada dado coletado e uma análise dos resultados obtidos. Por fim uma as considerações finais e conclusões da pesquisa.

Metodologia

As métricas usadas para medir o grau de manutenibilidade foram tiradas da metodologia proposta por Frantz et al (2015), onde este as usava para comparar diferentes plataformas de integração. Para averiguarmos a evolução dos graus de manutenibilidade no Mulesoft foi extraído medidas de distintas versões lançadas em ordem evolutiva da plataforma. Começando da 3.1, 3.2, 3.4, 3.5, 3.7 e a sua última versão lançada no tempo da pesquisa, a 3.8. É também importante especificar que apenas o núcleo do código fonte foi considerado para a coleta das medidas, visto que apenas o núcleo nos garante uma justa comparação entre todas as funcionalidades da plataforma em suas distintas versões.

Em Frantz, as métricas foram classificadas em quatro grandes grupos: Medidas de tamanho, de acoplamento, de complexidade e de herança. As mesmas foram obtidas através do uso do software Metrics (2005), na versão 1.3.6, nos auxiliando na coleta das informações e dados do núcleo do código fonte de cada versão do Mule. Neste artigo demos o foco para as medidas de tamanho e de complexidade que serão explicadas a seguir.

Medidas de tamanho são aquelas de quantidade, sobre quantos pacotes, classes, interfaces, métodos, parâmetros e linhas tem o código do software. Estas medidas nos mostram o quão grande o sistema é. Em especial, destacamos a quantidade de classes, interfaces e métodos do sistema.

NOC – Do inglês *Number of Classes*, ou o número de classes. Como sabemos, a fonte essencial para o príncipio das linguagens orientadas objetos são as classes. Então, quanto mais classes houverem, mais difícil será entender todas as suas funcionalidades.

NOI – Da sigla inglês *Number of Interfaces*, para o número de interfaces de um software. Funciona como o número de classes, porém de forma positiva pois um grande número de interfaces indicam uma maior facilidade para a manutenção adaptativa, visto sua característica de possibilidade de reuso.

NOM - Do inglês *Number of Methods*, para o número de métodos nas classes e interfaces. Uma grande quantidade de métodos em uma classe indica o quão específica uma classe é.

Classes e interfaces podem indicar o quanto de esforço é necessário para entender como o sistema de pacotes é organizado e ainda promove uma visão geral de todo o design do sistema. Já a quantidade de métodos tanto em classes como em interfaces, é uma métrica que pode ser usada para indicar o potencial reuso de uma classe. Contudo como fora citado em Frantz, que uma vasta gama de métodos demonstram o quanto uma classe está para uma aplicação específica, o que limita o seu potencial de reuso para uma manutenção adaptativa.

Com o foco na possibilidade de manutenção adaptativa do software, a complexidade dele nos diz muito, sendo uma medida essencial. O principal meio de medir sua complexidade é pela complexidade ciclomática de McCabe e a soma de seus pesos. Estas medidas indicam o quão complexo o software é, e o quão complexo e difícil pode ser entender e realizar manutenções no seu código.

MCC – Do inglês *McCabe cyclomatic complexity*, logo é a complexidade ciclomática de McCabe. Ela indica o quão complexo é o algoritmo em um método, sendo recomendado que a medida não ultrapasse o valor de dez, já que está diretamente relacionado com a dificuldade de manutenibilidade de um pedaço do software, sendo um fator questionável para a mesma.

WMC – Da singla inglesa *Weight Sum of McCabe cyclomatic complexity*, é a soma dos pesos da complexidade ciclomática de McCabe. Esta medida abrange todos os métodos em uma classe indicando o quão difícil será entender e depois modificar os mesmos. O quão maior for este valor, mais esforço é necessário para a aplicação de manutenção na classe.

Resultados e Discussão

Após a coleta dos dados referentes ao tamanho e a complexidade das diferentes versões em linha evolutiva do Mulesoft, houve a escolha de apresentar os dados em formato de tabela, que será detalhadamente explicada a seguir.

Versão Mule	3.1	3.2	3.4	3.5	3.7	3.8
NOC	766	822	977	1015	1154	1184
NOI	214	238	291	302	349	369
NOM	5390	5915	6963	7142	8140	8309
MCC	33	30	32	32	39	39
WMC	10959	11826	13642	13954	15750	16036

(Tabela 1: Comparação Evolutiva das Versões do Mulesoft)

Averiguando a Tabela 1, podemos ver claramente que o sistema de software do Mule cresceu proporcionalmente a cada versão. Começando com a versão 3.1 onde havia 766 classes, na sua próxima versão, a 3.2, este número já teria aumentado para 822 classes, ou seja, são 56 classes adicionais em apenas uma versão. Esta medida estava aumentando significativamente entre cada versão, chegando a ficar aproximadamente quarenta por cento maior entre as versão 3.1 até a versão 3.8, que chegou a 1184 classes. O mesmo foi constatado com as interfaces as quais na versão 3.1 eram 214, 24 novas interfaces foram adicionadas para a versão 3.2, e continuaram a ser adicionados em média 20 interfaces por versionamento, totalizando 369 interfaces na versão 3.8, o que presenta um crescimento de mais de cinquenta porcento do sistema. Consequentemente ao aumento da quantidade de classes e interfaces, o número de métodos também teve um crescimento equivalente, foram adicionados 2919 novos métodos durante todo a evolução do Mule, totalizando 8309 métodos na versão 3.8, o que representa outro aumento de mais de cinquenta porcento em função aos 5390 métodos da primeira versão. O aumento das medidas de tamanho do Mule indicam que tem ficado mais difícil aplicar a manutenção adaptativa

ao Mule, visto que um elevado número de classes demanda um grande esforço para entender suas funcionalidades, e um grande número de métodos por classes indica alta especificidade do software. Salvo o aumento do número de interfaces, que mesmo sendo necessário empregar o esforço de entendê-las, elas ainda são de grande prestígio por sua capacidade de reutilização.

Em vista a complexidade do sistema de software do Mule, desde o começo a sua complexidade ciclomática de McCabe já ultrapassa em muito o recomendado (10), logo na primeira versão atinge 33 pontos, o que é o triplo do recomendado, nas versões 3.2, 3.4 e 3.5 há uma pequena diminuição do valor, sendo 30, 32 e 32 respectivamente. Entretanto logo na versão 3.7 já ocorre um salto que quase quadruplica o recomendado, chegando a 39 até a última versão. Este alto valor demonstra que os métodos do software estão muito complicados e difíceis para a manutenção ser facilitada, para reduzir a complexidade ciclomática de McCabe recomenda-se que os métodos sejam quebrados em outros métodos, contudo isto também aumentará ainda mais o tamanho do software. Assim sendo o peso da soma da complexidade ciclomática de McCabe que está diretamente relacionada aos métodos, o quão complexos e difíceis são, tem seu aumento de quase cinquenta por cento explicado, com o crescimento significativo do sistema e a alta complexidade, totalizou 16036 na última versão, 5077 a mais que a primeira versão.

As medidas coletadas demonstraram que além de o tamanho do sistema ter aumentado muito ao longo do lançamento das novas versões, a sua complexidade algorítmica continua elevada em cada método, tornando o nível de complexidade do sistema como um todo elevado e dificultando em muito a possibilidade de manutenibilidade adaptativa do software.

Considerações Finais

A integração e suas ferramentas surgem com a necessidade de integrar aplicações novas, o que exige que a ferramenta esteja em constante evolução. Porém a evolução desta ferramenta pode acabar por aumentar o seu grau de dificuldade e complexidade para uma manutenção apropriada aos padrões de softwares recomendados. O uso das métricas possibilitam averiguar o nível de complexidade do código fonte de um software o que permitiu descobrir o nível da plataforma de integração Mulesoft, que está em constante evolução em suas versões e a um nível elevado de complexidade para a manutenibilidade adaptativa.

Palavras-Chave: Medidas de Software; Integração de Aplicações Empresarias; Manutenibilidade de Software; Plataformas de Integração.

Keywords: Software Metrics; Enterprise Application Integration; Software Manutenibility; Integration Platforms.

Referências Bibliográficas

FRANTZ, Rafael et al. **A methodology to evaluate the maintainability of enterprise application integration frameworks**. International Journal of Web Engineering and Technology 10, 4, 334–354, 2015.

HOHPE, Gregor e WOOLF, Bobby. **Enterprise integration patterns: Designing, building, and deploying messaging solutions**. Addison-Wesley Professional. 2004.

LANZA, Michele e MARINESCU, Radu. **Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of objectoriented systems**. Springer Science & Business Media, 2007.

METRICS, **Metrics 1.3.6**, 2005.

DOSSOT, David et al. **Mule in action**. Manning, 2014.

RADATZ, Jane et al. **IEEE standard glossary of software engineering terminology**. IEEE Std 610121990, 121990, 3, 1990.