Modalidade do trabalho: Relatório técnico-científico
Evento: VI Seminário de Inovação e Tecnologia

COMPARAÇÃO E ANÁLISE DA EVOLUÇÃO DO GRAU DE MANUTENIBILIDADE DA FERRAMENTA SPRING INTEGRATION[1]

Thainan H. Feistel[2], Fabricia Roos-Frantz[3].

[1] Pesquisa de Iniciação Tecnológica desenvolvida no Grupo de Pesquisa em Computação Aplicada da Unijui.
[2] Bolsista de Iniciação Tecnológica PIBITI/CNPq, qthainan@gmail.com
[3] Professora Orientadora, frfantz@unijui.edu.br

Introdução

Quando uma organização cresce em tamanho, são criados diferentes departamentos com áreas de foco distintas. O particionamento dos funcionários em diferentes setores dentro de organização normalmente é necessário. Mesmo dividida em diversos subgrupos, uma organização necessita da troca de informações entre seus setores, o compartilhamento de processos e dados se faz necessário para a execução de tarefas que envolvam dois ou mais setores. Mas como cada parte da organização tem uma aplicação diferente executando os seus processos, normalmente não se tem como fazer a troca de processos e dados entre esses setores, pois são diversos os fatores que fazem com que as aplicações sejam incompatíveis.

É a partir de um vasto ecossistema de softwares de uma empresa que a integração de aplicações empresariais se faz necessária. Uma das principais vantagens de se ter um ecossistema de softwares é o custo reduzido, pois o desenvolvimento de uma única aplicação que abrange todas as necessidades da empresa é significativamente maior [3].

A evolução faz parte do ciclo de vida de todo o software que se mantém útil ao longo do tempo. Esta evolução também pode ser vista como a implementação de novas funcionalidades ou apenas como a realização de simples manutenções. Consequentemente, esta evolução resulta em distintas versões de um software, sendo que cada versão subsequente pode apresentar um nível de complexidade maior. A manutenção de um software faz parte do seu processo de evolução e se caracteriza por qualquer alteração realizada no sistema do software que ocorre após ele ser entregue.

Se não controlada, o processo de evolução de um software pode ficar cada vez mais complexo de se implementar. Uma forma de medir o grau de manutenção de um software é por meio do uso de métricas voltadas a manutenção de software. As métricas têm como objetivo extrair medidas do código fonte e identificar em quais pontos o software está ficando mais difícil de manter, sendo que cada uma destas métricas aborda um aspecto em especifico do software [2]. A partir do conhecimento destas medidas, torna-se possível realizar uma análise das mesmas para decidir o que irá se fazer para diminuir o nível de complexidade da manutenibilidade do software. Este processo permite com que se tenha uma evolução no software sem que nível de complexidade do código fonte chegue a um ponto inviável de se manter.

A integração de aplicações pode ser desenvolvida a partir de diferentes ferramentas, dentre elas destacam-se Apache Camel, Spring Integration, Mule e Guaraná. Sendo que Spring Integration foi a ferramenta escolhida para estudo e análise neste artigo.

Modalidade do trabalho: Relatório técnico-científico
Evento: VI Seminário de Inovação e Tecnologia

Metodologia

Realizou-se um levantamento de todas as versões existentes da ferramenta Spring Integration com a finalidade de identificar em quais versões as mudanças foram mais significativas, o que resultou na escolha de quatro versões distintas. Após a seleção das versões, se fez necessária a utilização do software Metrics, que permite a extração das medidas necessárias para estudo a partir do código fonte do Spring Integration. O resultado desta extração foi uma tabela contendo os números de cada versão com base nas métricas utilizadas, onde é possível identificar se o software está crescendo controladamente, ou se, a cada versão, ele está ficando mais complexo de implementar.

Spring Integration

A ferramenta de integração Spring Integration providencia uma extensão do modelo de programação utilizado por outras aplicações da plataforma Spring para dar suporte aos clássicos padrões de integração propostos por Gregor Hohpe e Bobby Woolf [4]. É introduzido um sistema baseado em mensagens que possibilita a integração com sistemas externos por meio da utilização de adaptadores, tais adaptadores proporcionam um alto nível de abstração, o que deixa menor a complexidade das implementações. Segundo Mark Fisher [1], o principal objetivo da ferramenta de integração Spring Integration é providenciar um modelo simples para a implementação de soluções de integração empresariais complexas.

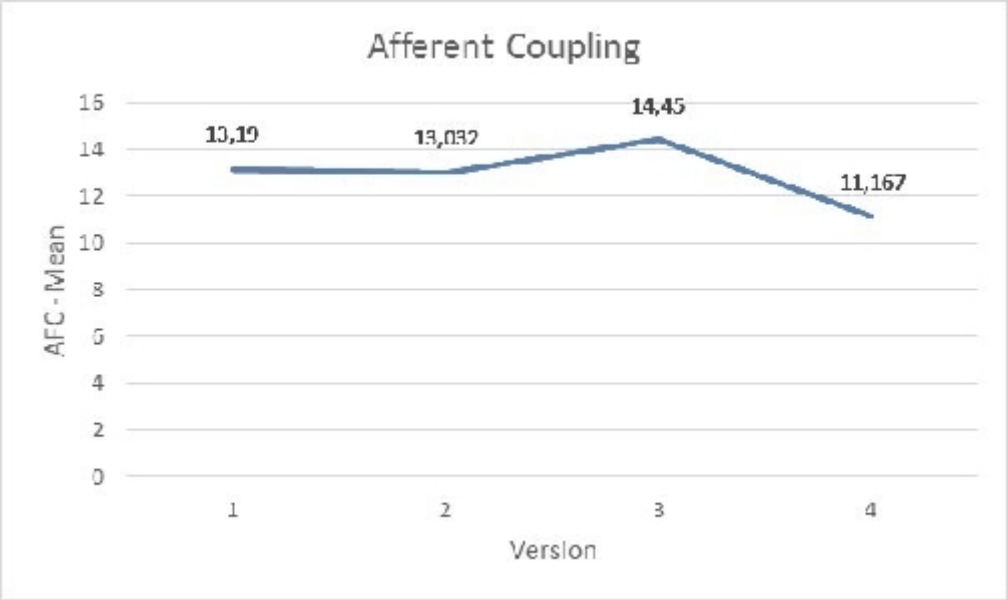

Figura 1. Gráfico da Métrica Afferent Coupling.

Modalidade do trabalho: Relatório técnico-científico
Evento: VI Seminário de Inovação e Tecnologia

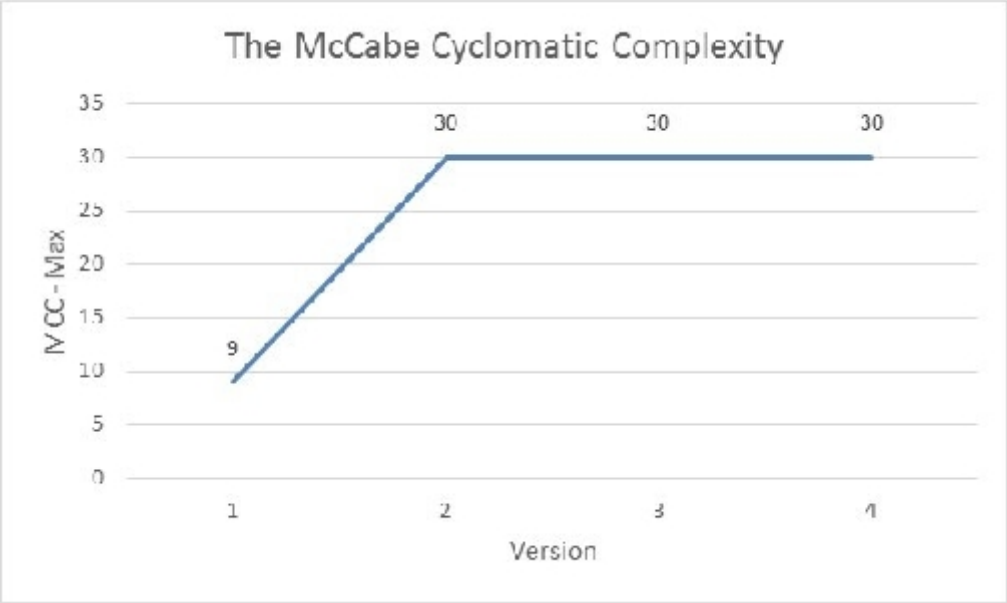

Figura 2. Gráfico da Métrica McCabe Cyclomatic Complexity.

Análise dos Resultados

A partir dos dados extraídos das distintas versões da ferramenta Spring Integration, realizou-se uma análise com o objetivo de identificar possíveis problemas que poderiam vir a existir com o crescimento da mesma. Como já ressaltado anteriormente, cada uma das métricas existentes aborda um aspecto em especifico da ferramenta [2], a partir disso, torna se possível a realização de uma análise detalhada de cada parte da ferramenta com base na métrica utilizada.

A utilização das métricas tem como objetivo indicar se a ferramenta está aumentando seu nível de complexidade em versões subsequentes, ou se está aprimorando sua implementação para que o nível de complexidade baixe. Com a análise realizada é possível notar que em um ponto a complexidade está aumentando, e que em outro ela está diminuindo. O que pode ser visualizado nos gráficos 1 e 2.

A métrica Afferent Coupling pode ser definida como o número de classes fora de um pacote que dependem de uma ou mais classes de dentro deste mesmo pacote [2]. Quanto maior o valor desta medida, mais complexo será de se realizar a manutenção da ferramenta, devido ao fato de existir um maior número de dependências entre as classes [5].

Observando o gráfico 1, pode-se notar que houve uma queda da versão 3 para a versão 4, isso acontece devido a divisão realizada em um pacote que estava com um número grande de classes, fazendo com que a dependência que existia das outras classes quanto a este pacote ficasse menor.

A métrica McCabe Cyclomatic Complexity pode ser utilizada como um indicador de quão complexo um método está [2]. De acordo com McCabe [6], esse valor não deve ser maior do que dez.

Observando o gráfico 2, pode-se notar que ocorreu um crescimento somente da versão um para a dois, e que nas outras versões não ocorreram mudanças. Como o número desta medida não deve ser maior que dez, pode-se dizer que existem métodos na ferramenta que estão com um alto nível de complexidade, e que ainda não foi realizado nada para tentar baixar este nível.

Modalidade do trabalho: Relatório técnico-científico
Evento: VI Seminário de Inovação e Tecnologia

Conclusão

As ferramentas de integração surgiram a partir da necessidade de se integrar aplicações. Estas ferramentas estão em constante evolução, com a finalidade de estarem sempre aptas a atender distintos tipos de problemas. O que acaba resultando na existência de distintas versões de uma ferramenta, sendo que a cada versão subsequente a ferramenta terá uma eficiência maior. Esta evolução pode fazer com que a complexidade de implementação da ferramenta chegue a um nível muito alto, fazendo com que se torne impossível realizar até uma simples manutenção. A utilização de métricas que determinam o nível de complexidade da evolução de um software permitiu averiguar o grau de complexidade atingido pela ferramenta Spring Integration, a qual está em constante evolução.

Palavras-chave: Integração de Aplicações Empresariais; Manutenibilidade de Software; Métricas.

Referências

[1] Mark Fisher, Jonas Partner, Marius Bogoevici, and Iwein Fuld. Spring integration in action. Manning Publications Co., 2012.

[2] Rafael Z Frantz, Rafael Corchuelo, and Fabricia Roos-Frantz. A methodology to evaluate the maintainability of enterprise application integration frameworks. International Journal of Web Engineering and Technology, 10(4):334–354, 2015.

[3] Rafael Z. Frantz, Antonia M. Reina Quintero, and Rafael Corchuelo. A domain-specific language to design enterprise application integration solutions. International Journal of Cooperative Information Systems, 20(02):143–176, 2011.

[4] Gregor Hohpe and Bobby Woolf. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional, 2004.

[5] Robert Cecil Martin. Agile software development: principles, patterns, and practices. Prentice Hall PTR, 2003.

[6] Thomas J McCabe. A complexity measure. Software Engineering, IEEE Transactions on, (4):308–320, 1976.