AVALIAÇÃO DA RECUPERAÇÃO ARQUITETURAL DE VISÕES MODULARES A PARTIR DE TÉCNICAS DE AGRUPAMENTO

Douglas Eder Uno Silva[1]; Roberto Almeida Bittencourt[2]

1. Bolsista PIBIC/FAPESB, Graduando em Engenharia da Computação, Universidade Estadual de Feira de Santana, email: douglas@ecomp.uefs.br

2. Orientador, Departamento de Exatas (DEXA), Universidade Estadual de Feira de Santana, email: roberto@ecomp.uefs.br

PALAVRAS-CHAVE: evolução de software, arquitetura de software, visão modular

1. INTRODUÇÃO

A maioria dos sistemas de software úteis precisam ser alterados ao longo do tempo, tanto para adicionar novas funcionalidades como para corrigir falhas. Muitas vezes, os sistemas legados podem ser modificados através de medidas emergenciais para manter as aplicações funcionando. Neste caso, é comum fazer a manutenção dos sistemas sem o entendimento completo da estrutura e organização do software. Tais modificações, por menores que sejam, podem alterar o funcionamento de outros componentes do software, comprometendo o seu funcionamento, resultando em reaparecimento de bugs antigos ou na criação de novos. A estrutura do sistema pode deteriorar até o ponto onde a organização do código-fonte se torne tão caótica que o software seja radicalmente reformulado ou abandonado (Mancoridis *et. al.* 1998).

Porém, segundo Wiggerts (1997), reescrever todo o sistema, na maioria das vezes, não é uma solução viável. Por essa razão, os engenheiros de software dependem de notações e ferramentas a fim de ajudá-los com a complexidade dos grandes sistemas de *software*. Dependendo do tamanho do sistema, a tarefa de recuperar sua estrutura original pode ser muito árdua. Por isso, várias abordagens e técnicas vêm sendo propostas na literatura para auxiliar na recuperação arquitetural de software.

A recuperação arquitetural possui um papel muito importante, uma vez que busca fornecer respostas para as perguntas dos arquitetos de software de como decompor um sistema em subsistemas menores e recuperar a estrutura original de um sistema legado ou até comparar os modelos documentados com o código-fonte para o aperfeiçoamento do sistema.

Pollet *et. al.* (2007) classificaram as técnicas de recuperação arquitetural em três: quase-manuais; semi-automáticas e quase-automáticas. Na técnica quase-manual, o engenheiro de *software* identifica manualmente os elementos arquiteturais com o auxílio de uma ferramenta. Na técnica semi-automática, o engenheiro de \textit{software} instrui a ferramenta com o intuito de identificar automaticamente elementos arquiteturais. Por fim, na técnica quase-automática, as ferramentas identificam os elementos arquiteturais, de forma automática, através de algoritmos próprios.

Algoritmos de agrupamento são técnicas quase-automáticas que procuram identificar grupos de entidades semelhantes a partir de suas características. De maneira geral, pode-se dizer que o agrupamento de entidades em vários grupos, ou módulos, a partir de uma relação de semelhança entre eles é uma forma de modularizar o sistema. Estes agrupamentos, formados por tais algoritmos de agrupamento, podem ser também chamados de *clusters*.

Contudo, cada algoritmo aplicado produz diferentes resultados. De modo geral, estes resultados podem ser avaliados através de métricas, em especial, por métricas que revelem quão similares os resultados de um algoritmo são em relação a um agrupamento de referência produzido manualmente por arquitetos de um dado sistema em análise.

2. METODOLOGIA

Este estudo foi realizado a partir dos fundamentos da metodologia de engenharia de *software* experimental, tendo como base a mineração de repositórios de *software*, com o intuito de simular a evolução do sistema alvo e tornar suas versões utilizáveis para a aplicação do estudo. A partir da extração das versões procurou-se compará-las para obter valores de estabilidade e autoridade.

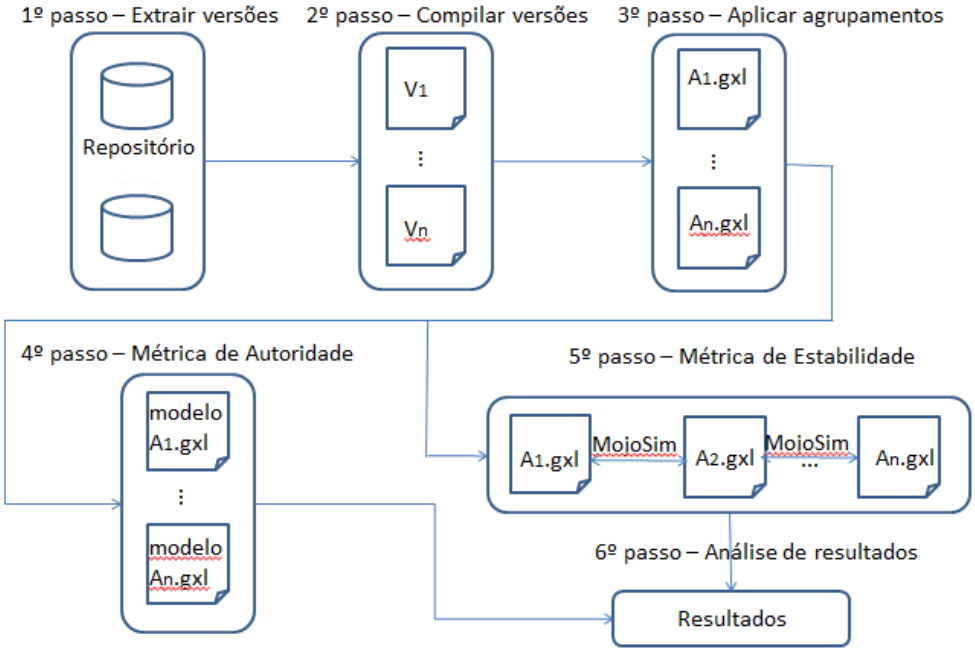

Figura 1 – Design Experimental

A figura 1 ilustra o *design* experimental deste trabalho. Primeiramente, versões sucessivas e semanais foram extraídas de repositórios, aqui chamadas de V_1 até V_n. Em seguida, essas versões foram compiladas para que a *Design Suite*, possa extrair os grafos de *design* das versões dos sistemas. Com os grafos das versões, aplicamos os algoritmos de agrupamento *K-Means*, Matrizes de Estrutura de Design (*DSM*) e Aglomerativos, gerando agrupamentos (ou *clusters*), aqui chamados de A_1 até A_n. Finalmente, empregamos as métricas de autoridade e estabilidade nessas versões para cada sistema, geramos alguns gráficos resultantes dessas métricas e os analisamos. Como o algoritmo *K-means* precisa do número de *clusters* como entrada, decidimos utilizar 10% da quantidade de classes dos sistemas alvo como padrão. O algoritmo alglomerativo foi aplicado em quatro configurações diferentes: SL75, SL90, CL75 e CL90, onde o 75 e o 90 indicam o ponto de corte no dendograma.

Sistemas de código aberto foram utilizados como alvos do estudo, foram eles: ***Lucene***, ***Argouml***, ***Ant*** e ***Sweethome3d***.

3. RESULTADOS

Como resultado, obtemos dois gráficos para cada sistema de acordo com o algoritmo de agrupamento aplicado, um gráfico de estabilidade e o segundo de autoridade. Como quatro sistemas foram analisados então obtemos 48 gráficos (4 sistemas gerando 12 gráficos cada). Alguns gráficos obtidos no estudo serão expostos neste documento.

Para cada gráfico o eixo das abscissas (eixo *x*) representa o número da versão analisada e o eixo das ordenadas (eixo *y*) representa os valores de estabilidade ou autoridade calculados. As figuras 2 e 3 ilustram dois gráficos obtidos a partir das duas métricas de avaliação de agrupamento utilizadas no sistema ***Lucene***. A figura 2 ilustra graficamente a

estabilidade, enquanto na figura 3 a autoridade que está em evidencia.

Os valores tanto de estabilidade quanto de autoridade geram valores entre 0 e 1, onde 0 significa total dissimilaridade e 1 completa similaridade. Foram examinadas versões semanais durante um ano, ou seja, 52 versões foram analisadas para cada sistema, totalizando 52 valores de autoridade e 51 valores de estabilidade.

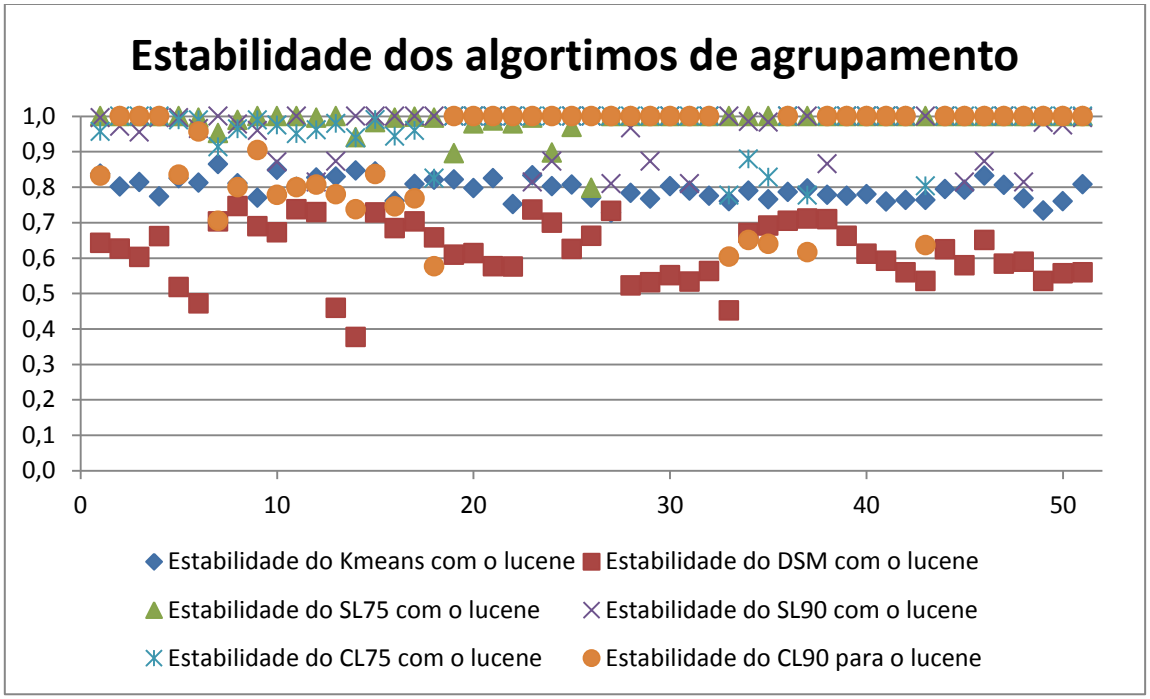

Figura 2 – Estabilidade do sistema *Lucene*

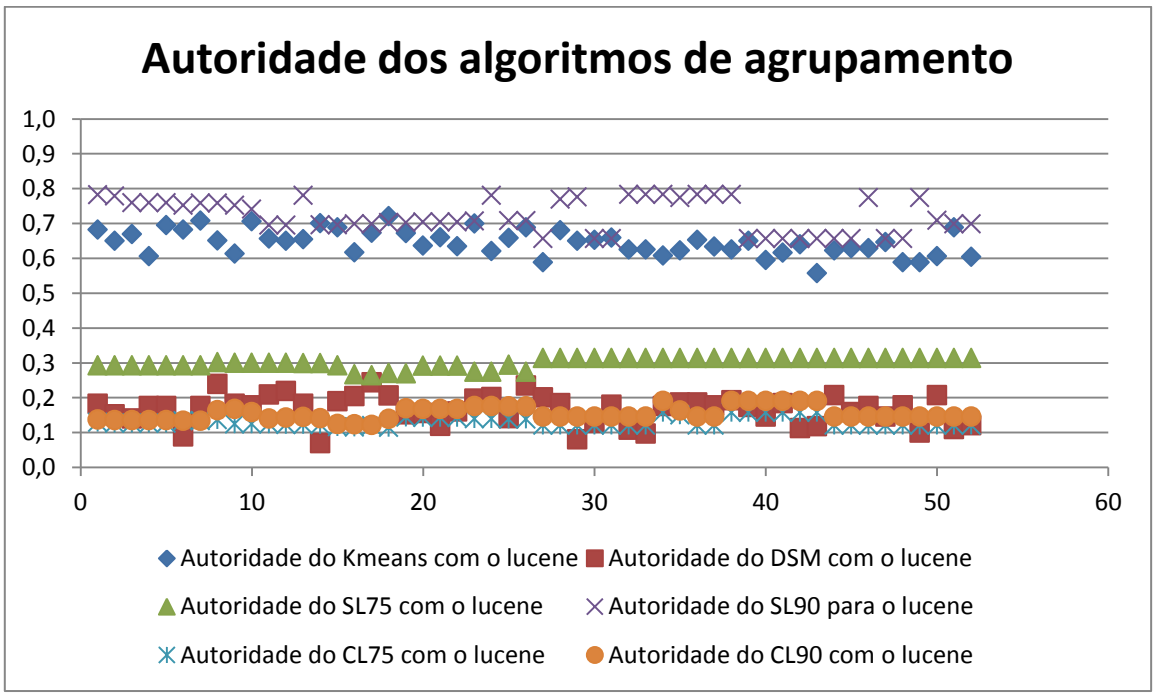

Figura 3 – Autoridade do sistema *Lucene*

Notou-se que o SL75 produziu a maior média de estabilidade e o SL90 gerou a melhor média de autoridade. O algoritmo por DSM produziu a pior média de estabilidade e o CL75 gerou a pior média de autoridade.

4. CONSIDERAÇÕES FINAIS

Este trabalho avaliou versões semanais e consecutivas de quatro sistemas e comparou três algoritmos de agrupamento. Duas métricas de qualidade chamadas de autoridade e estabilidade foram utilizadas, baseadas na métrica de similaridade *MoJoSim*.

A partir dos dados experimentais, pode-se concluir que o *K-Means* possui um melhor desempenho para geração de agrupamentos mais corretos na relação com o DSM, tanto para o critério de estabilidade, quanto para autoridade. O algoritmo aglomerativo, tanto SL quanto CL produziram bons resultados de estabilidade, contudo, o SL ainda foi melhor com relação à autoridade, principalmente o SL90. Entretanto, novos estudos com outras métricas de avaliação devem ser realizados.

Num desenvolvimento posterior nessa linha de estudo, pretende-se adicionar outras métricas de similaridade, como o *EdgeSim* e o *MeCl* apresentados por Mitchell e Mancoridis. Essas duas métricas avaliam outras formas de se identificar similaridade entre agrupamentos e, assim, melhor avaliar os algoritmos de agrupamento.

5. REFERÊNCIAS

MANCORIDIS, S.; B.S MITCHELL; C. RORRES.; Y. CHEN.; E.R. GANSNER. 1998. Using automatic clustering to produce high-level system organizations of source code. Program Comprehension, 1998. IWPC ’98. Proceedings., 6th International Workshop on , vol., no., pp.45-52, 24-26.

WIGGERTS, T.A. Using clustering algorithms in legacy systems remodularization. Reverse Engineering, 1997. Proceedings of the Fourth Working Conference on , vol., no., pp.33-43, 6-8 Oct 1997.

POLLET, D.; S. DUCASSE; L. POYET; I. ALLOUI; S. CIMPAN; H. VERJUS. Towards A Process-Oriented Software Architecture Reconstruction Taxonomy. Software Maintenance and Reengineering, 2007. CSMR ’07. 11th European Conference on , vol., no., pp.137-148, 21-23 March 2007.