

**UMA FERRAMENTA PARA
RECOMENDAÇÃO DE
DEMANDAS SIMILARES**

VAGNER CLEMENTINO DOS SANTOS

**UMA FERRAMENTA PARA
RECOMENDAÇÃO DE
DEMANDAS SIMILARES**

Proposta de dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: RODOLFO F. RESENDE

Belo Horizonte
Novembro de 2015

Lista de Figuras

Sumário

Lista de Figuras	v
1 Introdução	1
2 Justificativa	3
3 Revisão da Literatura	5
4 Metodologia	7
4.1 Revisão Sistemática da Literatura	7
4.2 Construção da Ferramenta	8
4.3 Avaliação	8
5 Conclusão e Trabalhos Futuros	9
Referências Bibliográficas	11

Capítulo 1

Introdução

Dentro do ciclo de vida do produto de software o processo de manutenção tem papel fundamental. Apesar de não ter merecido tanta atenção quanto a parte de projeto e desenvolvimento de software, no últimos anos o processo de manter o software vem ganhando relevância devido, primordialmente, à seu custo associado.

No Capítulo 2 . O Capítulo 3 . No Capítulo 4 é discutida a metodologia a ser aplicada. No Capítulo 5, especialmente na Tabela 5.1 é exibido o cronograma do trabalho.

Capítulo 2

Justificativa

Desde o final da década de 1970 percebe-se o aumento do custo referente as atividades de manutenção de software. Na Tabela 2.1, adaptada de [Koskinen, 2003], é possível verificar esta tendência. Em um trabalho mais recente [Tan & Mookerjee, 2005], Yong & Mookerjee propõe um modelo que reduz o custos de manutenção e reposição durante a vida útil de um sistema de software. O modelo proposto demonstrou quem em algumas situações é *melhor substituir um sistema do que mantê-lo*.

Ano	Proporção	Metodologia	Referência
2000	>90%	$\frac{\text{(Custo com manutenção e evolução do software)}}{\text{(Custo total do software)}}$ (Manutenção de Software)	[Erlikh, 2000]
1993	75%	$\frac{\text{(Orçamento com Sistemas de Informação)}}{\text{(Custo com manutenção de sistema)}}$ (Custo total com Software)	[Eastwood, 1993]
1990	>90%	$\frac{\text{(Orçamento com a operação de Sistemas de Gerenciamento da Informação)}}{\text{(Esforço gasto em manutenção de software)}}$ (Tempo total disponível para esforço em Engenharia de Software)	[Moad, 1990]
1988	60-70%	$\frac{\text{(Tempo total disponível para esforço em Engenharia de Software)}}{\text{(Tempo gasto com manutenção de software)}}$ (Tempo Total)	[Port, 1988]
1984	65-75%	$\frac{\text{(Custo com Manutenção)}}{\text{(Custo Total com Software)}}$	[McKee, 1984]
1981	50%		[Lientz & Swanson, 1981]
1979	67%		[Zelkowitz et al., 1979]

Tabela 2.1. Proporção do Custo de Manutenção de Software. Adaptado de [Koskinen, 2003]

Com o objetivo de mensurar o custo relativo à manutenção de software [Benaroch, 2013, Ren et al., 2011a, Ren et al., 2011b], bem com melhoria as atividades relacionadas ao processo de manutenção mediante a proposição de modelos [April & Abran, 2009], [Kajko-Mattsson, 2001] e [Junio et al., 2011] diversos trabalhos vêm sendo propostos. Nesta mesma linha de pesquisa, alguns estudo focam na melhoria da produtividade do desenvolvedor através de ações como a remoção de Requisições de Mudança - Modification Request (MR) duplicadas [Alipour et al., 2013, Liu et al., 2013, Cavalcanti et al., 2013] ou ainda recomendando MR's similares que reduzem a mudança de contexto (context switch) [Junio et al., 2011, Rocha et al., 2015].

Em geral, afim de remover duplicadas ou sugerir MR's similares são utilizadas de técnicas de Recuperação da Informação - Information Retrieve [Baeza-Yates et al., 1999] tomando como base a descrição textual da Requisição [Rocha et al., 2015, Runeson et al., 2007]. Apesar dos resultados relevantes obtidos por esta técnica verifica-se que ela é fortemente dependente da forma que o solicitante da MR descreve a sua requisição. Além disso há problema dos sinônimos, em que demandas similares mas, escritas com palavras diferentes, podem não ser recuperadas.

Neste contexto é proposto uma ferramenta para recomendação de Requisições de Mudanças similares utilizando Redes Neurais. Uma ferramenta deste tipo trará os seguinte benefícios:

- *Redução da troca de contexto (context switch);*
- *Aumento da produtividade;*
- *Melhorar a qualidade das MR's sugeridas através de uma técnica mais potente do que aquelas da Recuperação da Informação.*

O problema de recomendar Requisições de Mudanças similares pode ser definido formalmente conforme segue:

Seja I o conjunto de Requisições de Mudanças (MR) em aberto para um sistema S qualquer. A cardinalidade de I é dada por n . Seja i um MR, tal que $i \in I$, que foi atribuída para um desenvolver d . Pede-se que seja encontrado um subconjunto $J \subset I$, de tamanho $k \ll n$, tal que para todo j , tal que $j \in J$, seja similar a i em um grau maior ou igual a s . Onde s é limiar inferior de similaridade. Neste caso, o problema se resume em encontrar uma função de similaridade a ser aplicada a cada elemento $m \in I$, onde $m \neq i$.

Capítulo 3

Revisão da Literatura

No trabalho de Junio et al. [Junio et al., 2011] é proposto um processo denominado PASM (Process for Arranging Software Maintenance Requests) que propõe lidar com tarefas de manutenção como projetos de software. Para tanto, utilizou-se técnicas de análise de agrupamento (clustering) a fim de melhor compreender e comparar as demandas de manutenção. Os resultados demonstraram que depois de adotar PASM os desenvolvedores têm dedicado mais tempo para análise e validação e menos tempo para as tarefas de execução e de codificação.

NextBug [Rocha et al., 2015] é uma extensão (plugin) para a ferramenta de Controle de Demanda - Issue Tracking System (ITS) Bugzilla¹ que recomenda novos bugs para um desenvolvedor baseado no bug que ele esteja tratando atualmente. O objetivo da extensão é sugerir bugs com base em técnicas de Recuperação de Informação [Baeza-Yates et al., 1999].

¹<https://www.bugzilla.org/>

Capítulo 4

Metodologia

O processo de desenvolvimento deste trabalho pode ser dividido nas seguintes etapas *I - Revisão Sistemática da Literatura*; *II - Construção da Ferramenta (Prova de Conceito)*; *IV - Avaliação*. Cada uma das etapas é detalhada nas próximas seções.

4.1 Revisão Sistemática da Literatura

Uma *Revisão Sistemática da Literatura* - SLR (do inglês Systematic Literature Review) é uma metodologia científica cujo objetivo é identificar, avaliar e interpretar *toda* pesquisa *relevante* sobre uma questão de pesquisa, área ou fenômeno de interesse [Keele, 2007, Wohlin et al., 2012]. Neste trabalho será utilizada as diretrizes proposta [Keele, 2007] no qual uma Revisão Sistemática deve seguir os seguintes passos:

1. Planejamento

- a) *Identificar a necessidade da Revisão*
- b) *Especificar questões de pesquisa*
- c) *Desenvolver o Protocolo da Revisão*

2. Condução/Execução

- a) *Seleção dos Estudos Primários*
- b) *Análise da qualidade dos Estudos Primários*
- c) *Extração dos Dados*
- d) *Sintetização dos Dados*

3. Escrita/Publicação

- a) *Redigir documento com os resultados da Revisão*
- b) *Redigir documento com lições aprendidas*

A SRL que será realizada visa responder as seguintes questões de pesquisa:

- Q1: Quais são as metodologias/técnicas utilizadas para recomendação de novas demandas?
- Q2: Quais são as metodologias/técnicas utilizadas para remoção de bugs duplicados?
- Q3: Qual procedimento utilizado para avaliar a técnica/metodologia proposta?
- Q4: Em qual tipo de projeto (comercial ou código-aberto) a metodologia/técnica proposto foi aplicada?

4.2 Construção da Ferramenta

Como prova de conceito da técnica proposta será construída uma ferramenta que possibilite a recomendação Requisições de Mudança a serem tratadas a partir de uma demanda base, denominada “demanda-semente”. Esta ferramenta utilizar da técnica de Redes Neurais para encontrar as demais similares. O plano é que a base de dados a ser utilizada seja de uma empresa de desenvolvimento a fim de validarmos a ferramenta em uma situação real. Maiores detalhes sobre o processo de avaliação estão descritos na Seção 4.3.

4.3 Avaliação

Com o objetivo de avaliar a ferramentas proposto neste trabalho será realizado um *Quasi-Experimento* [Wohlin et al., 2012] utilizando a base de dados de demandas de manutenção de uma empresa de software real. O experimento consistirá de dado que uma demanda i foi atribuída a um desenvolvedor d , serão geradas 03 listas de sugestões: (i) lista produzida pelo gerente imediato de d ; (ii) lista feita por um desenvolver do mesmo setor de d ; (iii) gerada pela ferramenta proposta. Naturalmente o desenvolvedor não saberá a origem de nenhum das listas. De posse das três listas pediremos ao desenvolvedor d que informe qual delas pode reduzir a troca de contexto e aumentar sua produtividade. Espera-se a lista proposta pela nossa ferramenta possua o melhor desempenho na maioria dos casos.

Capítulo 5

Conclusão e Trabalhos Futuros

Para tanto, a tabela 5.1 descreve as atividades que serão realizadas para atingir este objetivo.

#	Atividade	Início (MM/AAAA)	Término (MM/AAAA)
01	Revisão da Literatura	10/2015	11/2015
02	Ponto de Controle 01 – Reunião com orientador sobre Revisão da Literatura	12/2016	12/2016
03	Avaliação da Técnica de Rede Neural	01/2016	01/2016
04	Ponto de Controle 02 – Reunião com orientador sobre a Técnica de Rede Neural	02/2016	02/2016
05	Implementação da Ferramenta	02/2016	04/2016
06	Ponto de Controle 03 – Avaliação da Ferramenta Avaliada	05/2016	05/2016
07	Experimento de Avaliação da Ferramenta	05/2016	05/2016
08	Ponto de Controle 04 – Avaliação do Experimento junto com o orientador	05/2016	05/2016
09	Finalização do texto da dissertação	06/2016	07/2016
10	Ponto de Controle 05 – Avaliação do texto da dissertação com o orientador	07/2016	07/2016
11	Defesa da dissertação	07/2016	07/2016

Tabela 5.1. Cronograma de execução do trabalho

Referências Bibliográficas

- [Alipour et al., 2013] Alipour, A.; Hindle, A. & Stroulia, E. (2013). A contextual approach towards more accurate duplicate bug report detection. Em *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pp. 183--192, Piscataway, NJ, USA. IEEE Press.
- [April & Abran, 2009] April, A. & Abran, A. (2009). A software maintenance maturity model (s3m): Measurement practices at maturity levels 3 and 4. *Electronic Notes in Theoretical Computer Science*, 233:73 – 87. ISSN 1571-0661. Proceedings of the International Workshop on Software Quality and Maintainability (SQM 2008).
- [Baeza-Yates et al., 1999] Baeza-Yates, R.; Ribeiro-Neto, B. et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.
- [Benaroch, 2013] Benaroch, M. (2013). Primary drivers of software maintenance cost studied using longitudinal data.
- [Cavalcanti et al., 2013] Cavalcanti, Y.; da Mota Silveira Neto, P.; Lucrédio, D.; Vale, T.; de Almeida, E. & de Lemos Meira, S. (2013). The bug report duplication problem: an exploratory study. *Software Quality Journal*, 21(1):39–66. ISSN 0963-9314.
- [Eastwood, 1993] Eastwood, A. (1993). Firm fires shots at legacy systems. *Computing Canada*, 19(2):17.
- [Erlikh, 2000] Erlikh, L. (2000). Leveraging legacy system dollars for e-business. *IT Professional*, 2(3):17–23. ISSN 1520-9202.
- [Junio et al., 2011] Junio, G.; Malta, M.; de Almeida Mossri, H.; Marques-Neto, H. & Valente, M. (2011). On the benefits of planning and grouping software maintenance requests. Em *Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on*, pp. 55–64. ISSN 1534-5351.

- [Kajko-Mattsson, 2001] Kajko-Mattsson, M. (2001). Motivating the corrective maintenance maturity model (cm3). Em *Engineering of Complex Computer Systems, 2001. Proceedings. Seventh IEEE International Conference on*, pp. 112–117.
- [Keele, 2007] Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Em *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*.
- [Koskinen, 2003] Koskinen, J. (2003). Software maintenance costs. *Information Technology Research Institute, ELTIS-Project University of Jyväskylä*.
- [Lientz & Swanson, 1981] Lientz, B. P. & Swanson, E. B. (1981). Problems in application software maintenance. *Commun. ACM*, 24(11):763--769. ISSN 0001-0782.
- [Liu et al., 2013] Liu, K.; Tan, H. B. K. & Zhang, H. (2013). Has this bug been reported? Em *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pp. 82–91.
- [McKee, 1984] McKee, J. R. (1984). Maintenance as a function of design. Em *Proceedings of the July 9-12, 1984, National Computer Conference and Exposition, AFIPS '84*, pp. 187--193, New York, NY, USA. ACM.
- [Moad, 1990] Moad, J. (1990). Maintaining the competitive edge. *Datamation*, 36(4):61.
- [Port, 1988] Port, O. (1988). The software trap – automate or else. *Business Week*, 9(3051):142--154.
- [Ren et al., 2011a] Ren, Y.; Xing, T.; Chen, X. & Chai, X. (2011a). Research on software maintenance cost of influence factor analysis and estimation method. Em *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pp. 1–4.
- [Ren et al., 2011b] Ren, Y.; Xing, T.; Chen, X. & Chai, X. (2011b). Research on software maintenance cost of influence factor analysis and estimation method. Em *Intelligent Systems and Applications (ISA), 2011 3rd International Workshop on*, pp. 1--4. IEEE.
- [Rocha et al., 2015] Rocha, H.; Oliveira, G.; Marques-Neto, H. & Valente, M. (2015). Nextbug: a bugzilla extension for recommending similar bugs. *Journal of Software Engineering Research and Development*, 3(1).

- [Runeson et al., 2007] Runeson, P.; Alexandersson, M. & Nyholm, O. (2007). Detection of duplicate defect reports using natural language processing. Em *Proceedings of the 29th International Conference on Software Engineering*, ICSE '07, pp. 499–510, Washington, DC, USA. IEEE Computer Society.
- [Tan & Mookerjee, 2005] Tan, Y. & Mookerjee, V. (2005). Comparing uniform and flexible policies for software maintenance and replacement. *Software Engineering, IEEE Transactions on*, 31(3):238–255. ISSN 0098-5589.
- [Wohlin et al., 2012] Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B. & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.
- [Zelkowitz et al., 1979] Zelkowitz, M. V.; Shaw, A. C. & Gannon, J. D. (1979). *Principles of Software Engineering and Design*. Prentice Hall Professional Technical Reference. ISBN 013710202X.