FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Software Repository Mining for Estimating Software Component Reliability

André Duarte - ei11044@fe.up.pt

PREPARAÇÃO DA DISSERTAÇÃO

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Rui Maranhão - rma@fe.up.pt

7 de Fevereiro de 2016

Software Repository Mining for Estimating Software Component Reliability

André Duarte - ei11044@fe.up.pt

Mestrado Integrado em Engenharia Informática e Computação

7 de Fevereiro de 2016

Resumo

Com a crescente necessidade de identificar a localização dos erros no código fonte de *software*, de forma a facilitar o trabalho dos programadores e a acelerar o processo de desenvolvimento, muitos avanços têm sido feitos na sua automação.

Existem três abordagens principais: *Program-spectra based* (PSB), *Model-based diagnosis* (MDB) e *Program slicing*.

Barinel, solução que integra tanto o PSB como o MDB, é, até hoje, com base na investigação feita, a que apresenta melhores resultados. Contudo, a ordenação de conjuntos de candidatos (componentes faltosos) não tem em conta a verdadeira qualidade do componente em causa, mas sim o conjunto de valores que maximizam a probabilidade do conjunto (*Maximum Likehood Estimation* - MLE), devido à dificuldade da sua determinação.

Com esta tese pretende-se colmatar esta falha e contribuir para uma melhor ordenação dos conjuntos, classificando, com recurso a técnicas de Machine Learning como *Naive Bayes*, *Support Vector Machines* (SVM) ou *Random Forests*, a qualidade e fiabilidade de cada componente, através das informações disponíveis no sistema de controlo de versões (*Software Repository Mining*), neste caso *Git*, como por exemplo: número de vezes que foi modificado, número de contribuidores, data de última alteração, nome de últimos contribuidores e tamanho das alterações.

A investigação já feita, revelou a existência de algumas soluções de análise preditiva de *software*, como *BugCache*, *FixCache* e *Change Classification*, capazes de identificar componentes com grande probabilidade de falhar e de classificar as revisões (*commits*) como faltosas ou não, mas nenhuma soluciona o problema.

Este trabalho visa também a integração com o *Crowbar* e a contribuição para a sua possível comercialização.

Palavras-chave: *Software-fault Localization*, *Software Repository Mining*, *Machine Learning*, *Classification*

Classificação: *Software and its engineering - Software creation and management - Software verification and validation; Computing methodologies - Machine Learning - Machine Learning Approaches*

"Software is eating the world."

Marc Andreessen

Conteúdo

CONTEÚDO

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

LISTA DE TABELAS

Abreviaturas e Símbolos

MSR	Mining Software Repositories
SFL	Spectrum-based Fault Localization
PSB	Program-spectra based
MDB	Model-based diagnosis
MLE	Maximum Likehood Estimation
SVM	Support Vector Machines

Capítulo 1

Introdução

1.1 Contexto/Enquadramento

Com o elevado crescimento da indústria de desenvolvimento de software, torna-se cada vez mais importante a existência de ferramentas que auxiliem os programadores a desenvolvê-lo mais eficientemente.

Estima-se que a economia dos Estados Unidos perca cerca de 60 mil milhões de dólares por ano em custos associados ao desenvolvimento e distribuição de correções para defeitos de *software* e na sua reinstalação. Pelo que, podemos afirmar que as ferramentas de localização das falhas de *software* (*Software Fault Localization*), ajudando a reduzir o tempo investido nesta tarefa, poderão ter um impacto significativo na economia. Nesta área os avanços são consideráveis. *Ochiai*, *Tarantula*, *Bayes-A* e *Barinel* são apenas algumas das soluções existentes, sendo o algoritmo *Barinel* aquele que apresenta melhores resultados. Apesar dos bons resultados apresentados pelo *Barinel*, este poderá apresentar resultados ainda mais rigorosos se tivermos informações relativas ao projeto, como a probabilidade média de erro ou a probabilidade de dado componente, que o constitui, ter defeitos.

Ferramentas de controlo de versões, como o *Git*, uma vez que mantêm todo o histórico do projeto e informações relacionadas com as diversas alterações (p.e. conteúdo, data e autores), em conjunto com técnicas de *Machine Learning*, poderão ser a chave para a melhoria deste algoritmo.

1.2 Motivação e Objetivos

Tendo em conta as possibilidades que a extração de dados de repositórios de controlo de versões e o *Machine Learning* nos dão, pretende-se com esta dissertação:

- Optimizar a ordenação de resultados candidatos do algoritmo Barinel.
- Ter a capacidade de prever a probabilidade de erro de cada um dos componentes de um dado projecto de *software* que use o *Git* para controlo total de versões, com uma precisão útil

1.3 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais x capítulos. No capítulo 2, é descrito o estado da arte e são apresentados trabalhos relacionados. No capítulo **??**, ipsum dolor sit amet, consectetuer adipiscing elit. No capítulo **??** praesent sit amet sem. No capítulo **??** posuere, ante non tristique consectetuer, dui elit scelerisque augue, eu vehicula nibh nisi ac est.

Capítulo 2

Revisão Bibliográfica

Neste capítulo é feita uma revisão bibliográfica e descrito o estado da arte do *software* de localização de falhas, como o Barinel, das diferentes abordagens de predição de defeitos e ainda de *Software Repository Mining*.

2.1 *Fault Localization Software*

O *Fault Localization Software* auxilia na localização automática do código que origina falhas na sua execução, diminuindo o custo desta identificação que teria de ser feita manualmente pelo programador. Existem duas categorias principais: *Program-spectra based* e *Model-based diagnosis* (MDB).

2.1.1 *Program-spectra based*

2.1.2 *Model-based diagnosis*

2.1.3 *Barinel*

O *Barinel* é um algoritmo que se inspira nos dois métodos descritos anteriormente, *program-spectra based* e *model-based diagnosis*, e que com isto consegue melhores resultados que as outras soluções com um custo pouco superior [AZG09].

O algoritmo começa por analisar uma *hit-spectra matrix*, que representa os testes executados em relação aos componentes que foram executados e ao seu resultado final.

Na tabela 2.1, temos identificados 3 componentes distintos (c_1, c_2 e c_3), 4 testes executados (t_1, t_2, t_3 e t_4) e o respectivo resultado da execução (e). O valor 1 em qualquer uma das colunas das observações (*obs*) indica que o dado componente foi executado nesse teste e o valor 0 indica o contrário, que o componente não foi executado. Na coluna e, o algarismo 1 declara que o teste correspondente falhou. Pelo que, por exemplo, o teste t_4 executou os componentes c_1 e c_3 e foi concluído com sucesso.

	obs			
	c_1	c_2	c_3	e
t_1	1	1	0	1
t_2	0	1	1	1
t_3	1	0	0	1
t_4	1	0	1	0

Tabela 2.1: *Hit-spectra matrix*

2.1.3.1 Geração de candidatos

Com base nesta matriz, uma lista de conjuntos de candidados (d) é gerada, sendo esta reduzida ao número mínimo de candidatos possível.

Neste caso, seriam gerados apenas dois candidatos:

- $d_1 = \{c_1, c_2\}$
- $d_2 = \{c_1, c_3\}$

2.1.3.2 Ordenação de candidatos

Para cada candidato d, é calculada a probabilidade de acordo com a regra de *Naïve Bayes*:

TODO $Pr(d|obs, e)$

$Pr(obs_i)$ é apenas um termo normalizador idêntico para todos os candidatos, pelo que não é usado para proceder à ordenação.

Sendo p_j a probabilidade à *priori* do componente c_j originar uma falha, podemos definir $Pr(d)$, probabilidade do candidato ser responsável pelo erro, não tendo em conta evidências adicionais, como

$$Pr(d) = \prod_{j \in d} p_j \cdot \prod_{j \notin d} (1 - p_j)$$

Sendo g_j (*component goodness*) a probabilidade do componente c_j executar de forma correta, temos que

TODO $Pr(obs_i, e_i|d)$

Tendo em conta o nosso exemplo

TODO $Pr(d_1, obs_i, e_i)$ **TODO** $Pr(d_2, obs_i, e_i)$

Quando existem valores g_j desconhecidos, é maximizado o valor de $Pr(obs, e|d)$ usando o algoritmo *Maximum Likelyhood Estimation* (MLE).

Neste caso, todos os valores de g_j são desconhecidos. Executando o algoritmo MLE para ambas as funções e calculando o resultado final temos que:

- $Pr(d_1, obs, e) = 1.9 \times 10^{-9}$ ($g_1 = 0.47$ e $g_2 = 0.19$)
- $Pr(d_2, obs, e) = 4.0 \times 10^{-10}$ ($g_1 = 0.41$ e $g_3 = 0.50$)

2.1.4 *Crowbar*

2.2 *Software Repository Mining*

Identificação de bugs - Erros libgit2 node-git

2.3 Abordagens à predição de defeitos

2.3.1 *BugCache*

2.3.2 *FixCache*

2.3.3 *Buggy Change Classification*

Time-weighted Risk WhoseFault History slicing/Chronos SZZ Machine Learning Classification: - Naive Bayes - SVM - Random Forests

Anexo A

Loren Ipsum

Depois das conclusões e antes das referências bibliográficas, apresenta-se neste anexo numerado o texto usado para preencher a dissertação.

A.1 O que é o *Loren Ipsum*?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum [**?**].

A.2 De onde Vem o Loren?

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet...", comes from a line in section 1.10.32.

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 from "de Finibus Bonorum et Malorum" by Cicero are also reproduced in their exact original form, accompanied by English versions from the 1914 translation by H. Rackham.

A.3 Porque se usa o Loren?

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using “Content here, content here”, making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for “lorem ipsum” will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

A.4 Onde se Podem Encontrar Exemplos?

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don’t look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn’t anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet. It uses a dictionary of over 200 Latin words, combined with a handful of model sentence structures, to generate Lorem Ipsum which looks reasonable. The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-characteristic words etc.

Referências

[AZG09] Rui Abreu, P Zoeteweij e a J C Van Gemund. Spectrum-Based Multiple Fault Localization. *Automated Software Engineering 2009 ASE 09 24th IEEEACM International Conference on*, pages 88–99, 2009. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5431781, doi:10.1109/ASE.2009.25.