UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TIAGO KENJI UMEMURA

UMA FERRAMENTA PARA MONITORAMENTO DA ENTROPIA DE MUDANÇA E SUA RELAÇÃO COM MÉTRICAS DE SOFTWARE

MONOGRAFIA

CAMPO MOURÃO

2016

TIAGO KENJI UMEMURA

UMA FERRAMENTA PARA MONITORAMENTO DA ENTROPIA DE MUDANÇA E SUA RELAÇÃO COM MÉTRICAS DE SOFTWARE

Proposta de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Igor Scaliante Wiese

CAMPO MOURÃO

2016

Resumo

. Uma ferramenta para monitoramento da entropia de mudança e sua relação com métricas de software. 2016. 16. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2016.

Contexto: A entropia de mudança é uma medida para indicar o quanto um software sofre alterações em um determinado período de tempo. Estudos mostraram que o aumento da entropia pode causar desordem no processo de desenvolvimento podendo levar ao aumento no número de defeitos do software. Além disso não há uma ferramenta que monitore a relação entre a entropia de mudança e métricas de softwares, por exemplo, número de autores que modificaram um arquivo, número de *commits*, *authorship* e *ownership*
Objetivo: Implementar uma ferramenta que possibilite o monitoramento da entropia e das métricas de softwares de projetos armazenados no Github para ajudar os desenvolvedores no gerenciamento de projeto.
Método: A ferramenta é dividida em coleta de dados, cálculo da entropia, cálculo das métricas, análise estatística, visualização de dados e avaliação. Na coleta de dados os dados serão extraídos do GHTorrent e da API Github e após a coleta será realizado o cálculo da entropia e das métricas de software utilizando principalmente a ferramenta Change Metrics. Na etapa de análise estísticas será feito a comparação e a correlação entre as métricas e em seguida será gerado um relatório sobre os resultados para o usuário. Na etapa de visualização de dados, a entropia de mudança será exibida utilizando o *Treemapping* e serão contruídos gráficos que mostram o comportamento das métricas ao longo do tempo. Para a avaliação da ferramenta será convidado desenvolvedores para testar a ferramenta e depois serão feitos questionários sobre a usabilidade da ferramenta para esses desenvolvedores.
Resultados esperados: É esperado que a ferramenta ajude os desenvolvedores a gerenciar melhor os projetos oferecendo relatórios estatísticos e visualizações da relação da entropia com as métricas sociais, de processo e de autoria.

Palavras-chaves: Entropia. Fatores Sociais. Defeitos.

Lista de figuras

Lista de tabelas

Sumário

Capítulo 1

Introdução

Os artefatos de softwares são modificados ao longo do tempo e nesse processo a qualidade do software tende a piorar (Hassan, 2009), uma vez que existe a necessidade da mudança contínua de requisitos durante a evolução do software.

Para quantificar o impacto das mudanças contínuas os pesquisadores (Hassan, 2009) tem utilizado o conceito de entropia de mudança. Esta medida pode ser obtida a partir do número de mudanças que ocorrem em um projeto ou arquivo em um determinado período de tempo. Hassan (2009) observou que maior quantidade de mudanças está relacionada com o aumento do valor da entropia e consequentemente está relacionado com maior tendência do software apresentar defeitos.

O impacto da entropia na qualidade do projeto também é investigado na pesquisa de Canfora *et al.* (2014), que analisou a relação entre a entropia e atividades de desenvolvimento, como a refatoração, padrões de projetos e a quantidade de desenvolvedores que mudam um determinado arquivo.

Estes estudos não consideram uma grande quantidade de métricas de software, por exemplo, não foi analisado a comunicação dos desenvolvedores, número de *commits* e número de arquivos que são alterados. Portanto não existe uma ferramenta que possibilite os desenvolvedores e os gerentes monitorarem os efeitos da entropia e sua relação com métricas sociais, de autoria e de processo.

Diante desse contexto, sabe-se que o desenvolvimento de software é uma tarefa sócio-técnica, pois é essencial a comunicação entre os desenvolvedores para coordenar as atividades de desenvolvimento durante a evolução do software. Assim o objetivo deste trabalho é construir uma ferramenta que monitora os valores de entropia e a sua relação com as métricas sociais, de autoria de mudança (authorship, ownership e experience), métricas de processo

(quantidade de commits, quantidade de defeitos, quantidade de linhas removidas, quantidade de linhas adicionadas, Code Churn, quantidade de refatorações, max change set e average change set) e número de defeitos.

A ferramenta irá gerar relatórios estatísticos sobre a entropia e as métricas e fornecerá a visualização desses dados para auxiliar os desenvolvedores na tomada de decisões durante o desenvolvimento de um projeto. O relatório estatístico será feito utilizando as técnicas de *Wilcoxon*, *ANOVA* e *Cliffs's test* e irá sugerir quais arquivos do projeto possuem maior tendência para apresentar defeitos. Na visualização de dados será utilizado o *Treemapping* para mostrar quais arquivos do projeto possuem maior entropia e para as métricas de software serão utilizados gŕaficos que mostrem os valores das métricas ao longo do tempo.

A última etapa será a avaliação da ferramenta que será feita com desenvolvedores que utilizarão a ferramenta e depois responderão um questionário sobre as funcionalidades e usabilidadade da ferramenta.

Esta proposta está organizada da seguinte forma. O capítulo 2 apresenta as definições das métricas sociais, de processos e os trabalhos relacionados. O capítulo 3 apresenta a proposta de construção da ferramenta, como a ferramenta será validada e uso da ferramenta. O capítulo 4 apresenta o cronograma.

Capítulo 2

Referencial Teórico

Este capítulo apresenta os conceitos de entropia, métricas socias e métricas de processo.

2.1. Entropia de mudança

A entropia de Shannon (2001) é uma medida para mensurar a incerteza associada a uma variável que quantifica uma informação contida em uma mensagem produzida por um emissor de dados. A partir dessa definição foi criado o conceito de entropia de mudança que tem como objetivo indicar o quanto um código está mudando durante um determinado período de tempo.

Na entropia de mudança o software é considerado o emissor de dados e as modificações realizadas são os dados de entrada. A entropia é uma medida para mensurar a quantidade de mudanças que ocorreram em um determinado espaço de tempo em um arquivo de um projeto. As mudanças consideradas podem ser obtidas a partir da quantidade de linhas modificadas em um intervalo de tempo ou utilizando número de *commits*.

A entropia de mudança é definida como (Hassan, 2009):

$$H(S) = \sum_{n=1} \frac{chg(f_i)}{chg(S)} log_2(\frac{chg(f_i)}{chg(S)}) \tag{2.1}$$

A figura 2.1 mostra um exemplo do cálculo da entropia dos arquivos de um sistema(Hassan, 2009). Nesse exemplo é considerado os arquivos A, B, C e D de um sistema dado um período de tempo qualquer e as estrelas indicam quando ocorreu uma mudança no arquivo.

Para cada arquivo no sistema, é feito a contagem de quantas ele foi modificado em um

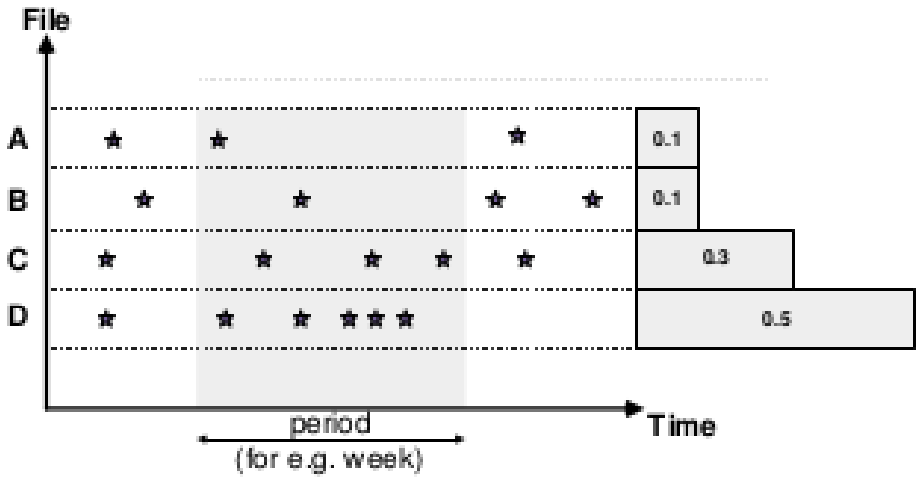

Figura 2.1. Exemplo de cálculo de entropia de mudança por arquivo.

período e depois é dividido pelo total de mudanças que ocorreram nesse mesmo período considerando todos os arquivos. Por exemplo, como ocorreram dez mudanças no período selecionado e uma mudança no arquivo A nesse mesmo paríodo então a entropia de mudança para esse arquivo é 0,1.

2.1.1. Trabalhos Relacionados

No trabalho de Hassan (2009) foi introduzido o conceito de entropia de mudança do software e foi feito um modelo básico de mudança de código(BCC - *Basic Code Change Model*) para mensurar a complexidade de um processo de mudança.

No modelo básico de mudança de código são utilizado arquivos como unidade de código para mensurar a complexidade de código, o intervalo de tempo para o cálculo da entropia é fixo e é considerado que os arquivos do projeto sejam sempre os mesmos. Para eliminar a limitação do tempo e a limitação alteração no número arquivos é utilizado o modelo extendido de mudança de código(ECC).

No ECC o histórico de mudanças é dividido em períodos de tamanhos iguais. Essa divisão pode ser feita com base em diferentes critérios: com base no tempo, no número de modificações ou em períodos onde ocorrem mais modificações(*Burst Based Period*). A entropia será calculada com base no número de modificações que ocorreram no período definido.

Hassan (2009) aplicou o modelo ECC para 6 projetos: NetBSD, FreeBSD, OpenBSD, Postgres, KDE e KOffice. Nesse estudo foi comparado modelos de predição de defeitos baseados na entropia, modelo baseados em defeitos anteriores e modelo baseado em modificações anteriores.

Os resultados indicam que modelos baseados na métrica de entropia tem o desempenho igual ou superior aos modelos baseados em modificações anteriores e baseados em defeitos anteriores.

A pesquisa de Canfora *et al.* (2014) relaciona a entropia de mudança com características do software e atividades de desenvolvimento. As características analisadas foram: refatoração, número de commiters, padrões de projetos e nome de tópicos no projeto. Na pesquisa foram analisados os projetos ArgoUML, Eclipse-JDT, Mozilla e Samba em um período de cerca de 10 anos.

O método para extração de dados foi dividido em 6 passos: extração das métricas de mudança do sistema de controle de versão, cálculo da entropia de mudança, identificação das mudanças relacionadas a refatoração, contagem número de autores que contribuem para o projeto, identificação dos padrões de projetos e por último foi necessário identificar os tópicos que são descritos na mensagem de cada *commit*.

Os resultados de Canfora indicam que: a entropia de mudança diminui de forma significante após uma atividade de refatoração, o valor da entropia é mais alto para arquivos com maior número de contribuidores, classes que participam de diferentes padrões de projetos exibem valores diferentes de entropia, mudanças classificadas em diferentes tópicos exibem valores de entropia diferentes e há maior relação entre o valor da entropia, tópicos de mudança e número de desenvolvedores que modificam um arquivo.

2.2. Métricas

Métricas de software são medidas para mensurar características presentes no desenvolvimento de software e servem para auxiliar na tomada de decisões durante o desenvolvimento do projeto.

2.2.1. Métricas de autoria

Esta seção apresenta métricas calculadas a partir das contribuições dos usuários em cada arquivo de um projeto.

2.2.1.1. Authorship

Authorship é uma medida para mensurar o quanto um desenvolvedor contribuiu para um determinado módulo de software.

Essa medida pode ser obtida de várias formas: contando número de arquivos que o desenvolvedor modificou, número de commits e outra possibilidade é contar número de linhas modificadas pelo contribuidor, também chamada de code churn(Munson; Elbaum, 1998).

Na pesquisa realizada por Rahman e Devanbu (2011) o authorship é calculado utilizando o número de linhas modificadas no código pelo desenvolvedor dividido pelo número total de linhas do arquivo, e o autor com a maior contribuição é denomidado ownership. Também é definido implicated code, que é o código modificado quando é corrigido um determinado erro no módulo de software. O trabalho de Rahman investiga a relação entre ownership, authorship e experience com implicated code. Para cada linha de código modificado é utilizado o comando blame para identificar o autor responsável por essa mudança. O resultado fornece evidências indicando que implicated code tende a ser mais frequentemente gerado por poucos autores, vários fragmentos de códigos modificados tem apenas um único autor.

2.2.1.2. Ownership

No trabalho de Greiler e Kim Herzig(Greiler; Herzig,) o Ownership é medido considerando número de contribuidores de um arquivo e também é verificado se existe um contribuidor principal, nesse caso a medida foi calculada com base no número de commits do autor em relação ao total de commits para aquele componente.

No artigo de Foucault *et al.* (2015) os contribuidores são classificados como owner, minor e major. Owner é o contribuidor com maior valor de contribuição, minor o desenvolvedor que contribuiu com menos de 5% e major contribuiu com mais de 5%.

A métrica social ownership será usada para se referir ao authorship do desenvolvedor que mais contribuiu com o projeto.

2.2.1.3. Experiência

A experience é a medida para calcular o nível de experiência do contribuidor, essa medida é computada analisando o número de linhas(Rahman; Devanbu, 2011) deltas comitadas pelo contribuidor em determinado espaço de tempo.

Na pesquisa de Rahman e Devanbu (2011) a experiencia é dividida em dois tipos: a experience especializada e experience geral. A experience especializada é medida considerando o quanto um indivíduo contribui em um determinado arquivo e a experiência geral é medida conseiderando um projeto inteiro.

2.2.2. Métricas Sociais

Esta seção apresenta métricas calculadas a partir da comunicação entre os desenvolvedores a partir das issues e de Pull Requests.

2.2.3. Métricas de processo

As métricas de processo são métricas para mensurar as características relacionadas aos artefatos produzidos durante o desenvolvimento do projeto. As métricas de processo serão descrita na tabela 2.1 e serão calculadas utilizando a ferramenta Change Metrics:

Tabela 2.1. Tabela de métricas de processo

Nome da métrica	Descrição
Quantidade de commits	A quantidade de commits representa o nível de atividade do projeto em termos de número de commits feitos. È calculado o número de commits do projeto em um certo perído de tempo.
Quantidade de defeitos	A quantidade de defeitos será calculado com o número de issues do projeto que foram criadas em uma determinada data.
Idade de repositório	A métrica idade do repositório representa o tempo de existência do projeto. O cálculo é utilizado medindo a diferença de tempo entre o primeiro e último commit.
Quantidade de linhas removidas	Essa medida representa a quantidade de linhas removidas de um arquivo até o momento.
Quantidade de linhas adicionadas	Essa medida representa a quantidade de linhas adicionadas de um arquivo até o momento.
Code Churn	A métrica Code Churn representa a soma de todas as linhas de código removidas e adicionas no arquivo.
Quantidade de refatorações	Essa medida representa a quantidade de refatorações ocorridas até o momento, se a refatoração é citada no commit.
Max Change Set	A métrica Max Change Set representa o número máximo de arquivos que foram alterados junto com o arquivo em questão.
Average Change Set	As métricas Average Change Set representa a média de número de arquivos alterados juntos.

Capítulo 3

Proposta

Nos estudos realizados por Canfora e Hassan não foi analisado os efeitos da entropia nas métricas sociais e de processo. Além disso não há uma ferramenta que monitore a entropia e as métricas do projeto.

O objetivo é construir uma ferramenta que monitore o valor da entropia de mudança e sua relação com as métricas sociais, de autoria e de processos dos projetos. A ferramenta terá seis módulos que serão apresentados a seguir:

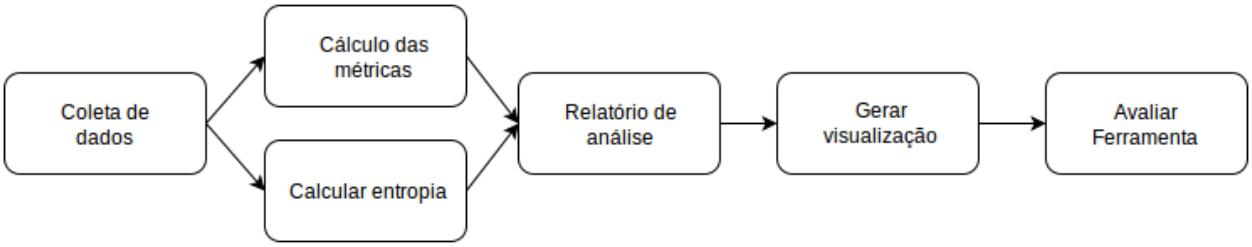

Figura 3.1. Fluxograma do funcionamento da ferramenta

3.1. Coleta de dados

O módulo de coleta de dados é responsável pela extração de dados dos projetos do Github e Git que serão persistidos em um banco de dados local. Os dados serão obtidos utilizando o GHTorrent e a ferramenta Change Metrics.

Utilizando a base de dados do GHTorrent será extraído do MySQL todos os *Hash* dos *commits* para assim ter acesso à todas as versões do projeto. E como a ferramenta Change Metrics calcula as métricas de processo do ínicio do projeto até a data da versão que está sendo analisada, é necessário guardar os dados das métricas de processo de todas as versões.

Juntamente com os *commits* é necessário extrair os autores de cada commit e sua data para realizar o cálculo do *authorship* e *ownership*

3.2. Cálculo das métricas

O módulo de cálculo das métricas irá realizar o cálculo após a extração dos dados no módulo de coleta de dados. Nessa etapa o usuário deverá informar quais métricas ele deseja calcular e uma data anterior a data atual, o período definido será entre a data atual e a data definida pelo usuário.

Para utilizar a ferramenta Change Metrics será necessário realizar clones das diferentes versões do projeto para assim comparar as métricas de processo entre as versões.

3.3. Cálculo da entropia

Este módulo que realiza o cálculo da entropia de cada arquivo do projeto no período que o usuário informar. O valor da entropia será determinado utilizando como medida o número de commits que teve no arquivo durante esse período de tempo. Será necessário calcular a diferença da métrica Revisões(número total de commits) gerada pela ferramenta Change Metrics entre duas versões diferentes do projeto e assim obter o número de commits de cada arquivo entre uma versão e outra.

3.4. Relatório de análise

Como foi feito na pesquisa de Canfora *et al.* (2014), para comparar uma métrica entre uma versão antiga do projeto e outra mais nova será feita uma comparação estatística utilizando *Wilcoxon* e para analisar a interação entre diferentes métricas será utilizada o ANOVA.

Após a análise estatísticas será gerado um relatório informando o usuário identificando quais métricas está relacionada com a entropia e quais arquivos possuem maior entropia.

3.5. Visualização de dados

Este módulo irá mostrar os resultados com gráficos bidimensionais e uma visualização utilizando o método Treemapping, exibindo uma hierarquia de dados utilizando retângulos aninhados. Será feito um gráfico para cada métrica selecionada no módulo de cálculo de métricas, onde o eixo x será o tempo e o eixo f(x) será o valor da métrica. O Treemapping

será utilizado para visualizar o valor da entropia em cada arquivo do projeto, quanto maior a entropia maior será o retângulo.

Figura 3.2. Exemplo de visualização utilizando Treemapping.

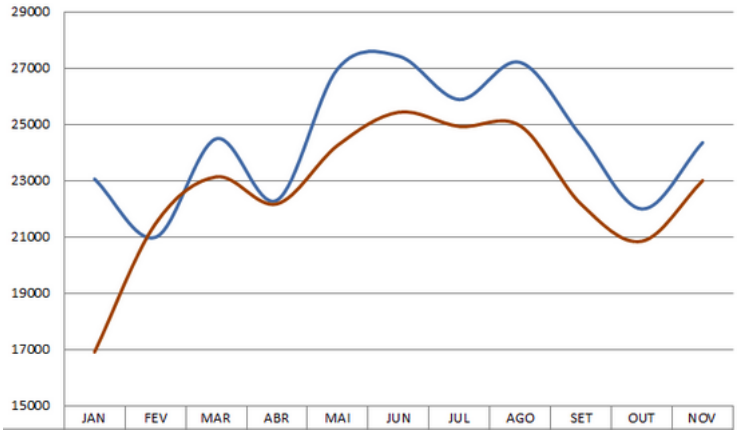

Figura 3.3. Exemplo de gráfico de duas métricas ao longo do tempo

Capítulo

4

Cronograma

Referências

CANFORA, Gerardo; CERULO, Luigi; CIMITILE, Marta; PENTA, Massimiliano Di. How changes affect software entropy: an empirical study. *Empirical Software Engineering*, v. 19, n. 1, p. 1–38, 2014. ISSN 1573-7616. Disponível em: `http://dx.doi.org/10.1007/s10664-012-9214-z`.

FOUCAULT, Matthieu; TEYTON, Cédric; LO, David; BLANC, Xavier; FALLERI, Jean-rémy. On the usefulness of ownership metrics in open-source software projects. *Information and Software Technology*, Elsevier B.V., v. 64, p. 102–112, 2015. ISSN 0950-5849. Disponível em: `http://dx.doi.org/10.1016/j.infsof.2015.01.013`.

GREILER, Michaela; HERZIG, Kim. Code Ownership and Software Quality : A Replication Study. [s.d.].

HASSAN, Ahmed E. Predicting faults using the complexity of code changes. In: *Proceedings of the 31st International Conference on Software Engineering*, 2009. (ICSE '09), p. 78–88. ISBN 978-1-4244-3453-4. Disponível em: `http://dx.doi.org/10.1109/ICSE.2009.5070510`.

MUNSON, J. C.; ELBAUM, S. G. Code churn: A measure for estimating the impact of code change. In: *Proceedings of the International Conference on Software Maintenance*, 1998. (ICSM '98), p. 24–. ISBN 0-8186-8779-7. Disponível em: `http://dl.acm.org/citation.cfm?id=850947.853326`.

RAHMAN, Foyzur; DEVANBU, Premkumar. Ownership , Experience and Defects :. 2011.

SHANNON, C. E. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 5, n. 1, p. 3–55, jan. 2001. ISSN 1559-1662. Disponível em: `http://doi.acm.org/10.1145/584091.584093`.