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

TIAGO KENJI UMEMURA

IMPACTO DA ENTROPIA EM FATORES SOCIAIS

MONOGRAFIA

CAMPO MOURÃO

2016

TIAGO KENJI UMEMURA

IMPACTO DA ENTROPIA EM FATORES SOCIAIS

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

. Impacto da Entropia em fatores sociais. 2016. 17. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2016.

A entropia de mudança é uma medida para indicar o quanto um software sofre alterações em um determinado período de tempo e estudos mostraram que o aumento da entropia pode causar desordem no processo de desenvolvimento podendo levar ao aumento no número de defeitos do software. Nesse contexto, é importante observar que as pessoas fazem parte do processo de desenvolvimento do software assim os fatores sociais podem estar relacionados com a entropia.

Contexto: A pesquisa será realizada utilizando os repositórios do sistema de controle de versões Git.
Objetivo: Identificar quais fatores sociais tem relação com a entropia e se existe impacto na qualidade do software.
Método: A entropia e os fatores sociais serão calculados utilizando dados extraídos da base de dados GHTorrent, API Github e a ferramenta change-metrics.
Resultados esperados: É esperado que a entropia esteja relacionada com os aspectos sociais.

Palavras-chaves: Entropia. Fatores Sociais. Qualidade. Defeitos.

Lista de figuras

Lista de tabelas

Sumário

1 Introdução **6**

2 Referencial Teórico **8**

2.1 Entropia de mudança 8

2.1.1 Trabalhos Relacionados 9

2.2 Métricas 11

2.2.1 Métricas de autoria 11

2.2.1.1 Authorship 11

2.2.1.2 Ownership 11

2.2.1.3 Experiência 12

2.2.2 Métricas Sociais 12

2.2.2.1 Quantidade de mensagens trocadas 12

2.2.3 Métricas de processo 12

2.2.3.1 Quantidade de autores 12

2.2.3.2 Quantidade de commits 12

2.2.3.3 Quantidade de defeitos 12

2.2.3.4 Idade do repositório 13

3 Proposta **14**

3.1 Coleta de dados 14

3.2 Cálculo das métricas 14

3.3 Cálculo da entropia 15

3.4 Visualização de dados 15

4 Cronograma **16**

Referências **17**

Capítulo 1

Introdução

Os artefatos de softwares são modificados ao longo do tempo e nesse processo a qualidade do software tende a piorar. Como consequência da mudança contínua de requisitos a complexidade do código tende a aumentar o que dificulta futuras manuntenções.

Para quantificar o impacto das mudanças contínuas os pesquisadores tem utilizado o conceito de entropia de mudança. A entropia é obtida a partir do número de mudanças que ocorrem em um projeto ou arquivo em um determinado período de tempo. Pesquisas anteriores observaram que a quantidade de mudanças está relacionada com o aumento do valor da entropia.

Hassan (2009) mostrou que o aumento da entropia 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 aspectos sociais, como por exemplo, a comunicação dos desenvolvedores, métricas de código e métricas de processo. Além disso não existe uma ferramenta que possibilite os desenvolvedores e os gerentes monitorarem os efeitos da entropia e a relação com outros indicadores

Diante desse contexto sabe-se que o desenvolvimento de software é uma tarefa sócio-técnica. Assim o objetivo desse trabalho é construir uma ferramenta que monitora os valores de entropia e a sua relação com os aspectos sociais(), de autoria de mudança(authorshipm ownership e experience), de métricas de processo(change metrics arrumar depois) e de qualidade(número de defeitos).

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, e as questões de pesquisa que serão usadas como forma de validar 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 e indicadores de qualidade.

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. É utilizada para determinar a quantidade de bits necessários para identificar unicamente um distribuidor de dados, assim quanto maior a entropia maior a incerteza para identifica-lo. A partir disso foi criado a entropia de mudança com o objetivo de calcular o quanto um código está mudando durante um determinado período de tempo.

A entropia de mudança introduzida por Hassan (2009) considera que o software é o emissor de dados e as modificações realizadas são os dados a serem considerados. É uma medida para mensurar a quantidade de mudanças que ocorreram em um determinado espaço de tempo em um arquivo de um projeto, as mudança consideradas podem ser obtidas a partir da quantidade de linhas modificadas em um intervalo de tempo ou utilizando número de commits. Hassan também mostrou que alta entropia está relacionada a maior tendência do projeto apresentar falhas.

A entropia de mudança é definida como:

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

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(Basic Code Change Model - BCC) para mensurar a complexidade de um processo de mudança.

No modelo básico de mudança de código é 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 anteiores e baseados em defeitos anteiores.

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 projetos nos sistemas ArgoUML, Eclipse-JDT, Mozilla e Samba em um período de cerca de 10 anos e foi mostrado como esses fatores se relacionam com a entropia, que por consequencia impacta a qualidade do software.

Nesse estudo, o método para extração de dados é 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, contar número de autores que contribuem para o projeto, identificação dos padrões de projetos e por último é necessário identificar os tópicos que são descritos na mensagem de cada commit.

Após a extração dos dados é feito a análise desses dados para responder cinco questões de pesquisa: Como a refatoração afeta a entropia de mudança(RQ1), como a mudança de entropia está relacionada ao número de contribuidores de um arquivo(RQ2), como a entropia

de mudança varia entre classes que participam ou não dos padrões de projetos(RQ3), como a mudança de entropia se relaciona com os tópicos descritos na mensagem do commit(RQ4) e como todos os fatores descritos nas questões de pesquisa anteriores se relacionam com a entropia de mudança(RQ5).

Para responder a RQ1, é feito a comparação do valor da entropia antes e depois da refatoração.

Para RQ2 é utilizado boxplots onde é contado o número cumulativo de desenvolvedores que, até aquela mudança, tenham alterado o arquivo após isso é feita a correlação entre a mudança de entropia observada em um arquivo para cada conjunto de mudança e o número de desenvolvedores que modificaram o arquivo até aquele conjunto de mudança, considerando todos os arquivos para todos os conjuntos de mudança.

Na RQ3 é utilizado teste de Mann-Whitney e Cliff's delta para comparar mudanças de entropia para classes que participam ou não de padrões de projeto. Então é feito o teste de Kruskal-Wallis para checar se diferentes tipos de padrões de projeto exibem valores diferentes de entropia.

Na RQ4, após classificar o conjunto de mudanças de acordo com os tópicos da mensagem de commit, é comparado a entropia de mudança pertencente a diferentes tópicos utilizando o teste de Kruskal-Wallis.

A RQ5 tem como objetivo analizar a relação entre os diferentes fatores descritos nas questões de pesquisa anteriores e como esses fatores afetam a entropia de mudança. Para responder essa questão foi utilizado Kruskal-Wallis e ANOVA.

Os resultados de Canfora indicam que: a entropia de mudança diminui de forma significante após uma atividade de refatoração(RQ1), o valor da entropia é mais alto para arquivos com maior número de contribuidores(RQ2), classes que participam de diferentes padrões de projetos exibem valores diferentes de entropia(RQ3), mudanças classificadas em diferentes tópicos exibem valores de entropia diferentes(RQ4) e há maior relação entre o valor da entropia, tópicos de mudança e número de desenvolvedores que modificam um arquivo(RQ5).

2.2. Métricas

2.2.1. Métricas de autoria

Esta seção apresenta métricas para calculadas a partir das atividades dos usuário em cada 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 RahmanRahman 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

2.2.2.1. Quantidade de mensagens trocadas

2.2.3. Métricas de processo

Esta seção apresenta as métricas extraídas utilizando a ferramenta Change Metrics desenvolvida por Maurício Aniche.

2.2.3.1. Quantidade de autores

Esta métrica é utilizada para contar a quantidade de autores diferentes que contribuem com o projeto.

2.2.3.2. 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.

2.2.3.3. Quantidade de defeitos

A quantidade de defeitos será calculado com o número de issues do projeto que foram criadas em uma determinada data. Essa data é obtida a partir do GHTorrent no campo created_at.

2.2.3.4. Idade do 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.

Capítulo 3

Proposta

Nos estudos realizados por Canfora e Hassan não foi analisado os efeitos da entropia nas métricas de código 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 forneça o valor da entropia de mudança, das métricas de código, de processo e indicadores de qualidade(número de defeitos) dos projetos, permitindo o monitoramento desses projetos. A ferramenta terá quatro módulos que serão apresentados a seguir:

3.1. Coleta de dados

O módulo de coleta de dados é responsável pela extração de dados dos projetos do Github. Os dados serão obtidos utilizando o GHTorrent, a ferramenta Change Metrics e Github API V3.

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.

3.3. Cálculo da entropia

Este módulo que realiza o cálculo da entropia de cada arquivo do projeto no período que o usuário informar. As outras métricas serão calculadas nesse mesmo período.

3.4. 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.

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`.