

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. 16. 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

---

<b>1</b>	<b>Introdução</b>	<b>6</b>
<b>2</b>	<b>Referencial Teórico</b>	<b>8</b>
2.1	Entropia de mudança . . . . .	8
2.1.1	Trabalhos Relacionados . . . . .	9
2.2	Métricas . . . . .	9
2.2.1	Métricas de autoria . . . . .	9
2.2.1.1	Authorship . . . . .	9
2.2.1.2	Ownership . . . . .	11
2.2.1.3	Experience . . . . .	11
2.2.2	Métricas Sociais . . . . .	11
2.2.2.1	Quantidade de mensagens trocadas . . . . .	11
2.2.3	Métricas de processo . . . . .	11
2.2.3.1	Quantidade de autores . . . . .	11
2.2.3.2	Quantidade de commits . . . . .	12
2.2.3.3	Quantidade de defeitos . . . . .	12
2.2.3.4	Idade do repositório . . . . .	12
<b>3</b>	<b>Proposta</b>	<b>13</b>
3.1	Coleta de dados . . . . .	13
3.2	Cálculo das métricas . . . . .	13
3.3	Cálculo da entropia . . . . .	13
3.4	Visualização de dados . . . . .	14



---

# 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 manutenções.

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 monitorea os valores de entropia e a sua relação com os aspectos sociais(), de autoria de mudança(authorship 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.

---

## Referencial Teórico

---

Este capítulo apresenta os conceitos de entropia, métricas sociais 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:

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

### 2.1.1. Trabalhos Relacionados

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 comitters, 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 consequência impacta a qualidade do software. Os resultados de Canfora indicam que a entropia de mudança varia conforme o número de desenvolvedores aumentam, sendo que diferentes sistemas de controle de versão apresentam variações diferentes na entropia.

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. No caso da refatoração, é feito a comparação do valor da entropia antes e depois da refatoração.

## 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 e o ownership é o autor com maior authorship, assim um módulo de software pode apresentar ownership forte ou distribuído entre os contribuidores.

Essas duas medidas podem ser obtidas 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).

No trabalho de Greiler e Kim Herzig (Greiler; Herzig, ) é feito um estudo para relacionar Ownership com a qualidade do código. Para medir a qualidade do código é considerado o número de bugs que foram corrigidos, número de diretórios e arquivos defeituosos e 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. Também foi mostrado que módulos com ownership fraco tendem a apresentar mais defeitos. Nesse mesmo trabalho foi estudado quatro diferentes software para Windows e é possível observar que nos sistemas analisados a métrica de ownership está correlacionado com número de defeitos. Isso é observado na análise feita tanto em nível de arquivo quanto em nível de diretório. Essa pesquisa mostrou que o número de contribuidores e a porcentagem de mudanças aplicadas pelo ownership são bons indicadores de qualidade, mostrando que há relação entre ownership e a qualidade de um projeto apesar da quantidade de erros também depender do tamanho do projeto.

Na pesquisa realizada por Rahman e Devanbu (2011) é analisado a relação entre o número de autores com o ownership e com a qualidade do código. 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 denominado 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.

O artigo de Foucault *et al.* (2015) também estuda como o ownership impacta na qualidade do código. Nele 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%. Após medir as métricas é comparado o número de erros com o a medida de ownership.

Além disso também há pesquisas (Thongtanunam *et al.*, ) que analisaram a diferença entre a contribuição de code authoring e reviewing e como a atividade de code reviewing afetam projetos com vários autores que contribuem pouco o que ajuda a compreender melhor o impacto do ownership na qualidade do código.

#### **2.2.1.2. Ownership**

#### **2.2.1.3. Experience**

A experience do desenvolvedor influencia na produtividade do mesmo e na qualidade do código produzido como foi mostrado em outras pesquisas(Rahman; Devanbu, 2011)(Banker gordon b. davis, 1998) e quanto mais o desenvolvedor trabalha em diferentes componentes do sistema maior será a sua experience.

Já foi feito pesquisas(Rahman; Devanbu, 2011) mostrando a relação entre experience e qualidade do código além disso a experience 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.

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. Além disso Rahman já mostrou que experiência especializada leva a produzir códigos com menor número de erros e ainda não foi possível concluir que a experiência geral contribui para aumentar a qualidade. No trabalho de Rahman foi correlacionado implicated code com experience.

### **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. É calculado o número de commits do projeto em um certo período 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.

canfora e hassan

---

## 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 por....

### 3.2. Cálculo das métricas

É o modulo onde será calculado....

### 3.3. Cálculo da entropia

Módulo que realiza o cálculo da entropia

### 3.4. Visualização de dados

Visualizar os dados gerados pela ferramenta

QP1: Maior entropia tem como consequência débito social?

Abordagem: Utilizando o GHTorrent será extraído o número de commits de um projeto em um determinado período, em seguida será calculado a entropia desse projeto nesse mesmo período.

As informações extraídas do GHTorrent não são suficientes para calcular as métricas sociais então é necessário utilizar também API do github para coletar outros dados.

Resultados esperados:

QP2: Maior entropia degrada a qualidade do projeto?

Abordagem: como será o procedimento para responder a QP2.

Resultados esperados:



# Referências

---

BANKER GORDON B. DAVIS, Sandra A. Slaughter Rajiv D. Software development practices, software complexity, and software maintenance performance: A field study. *Management Science*, INFORMS, v. 44, n. 4, p. 433–450, 1998. ISSN 00251909, 15265501. Disponível em: <http://www.jstor.org/stable/2634607>.

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>.

THONGTANUNAM, Patanamon; MCINTOSH, Shane; HASSAN, Ahmed E; IIDA, Hajimu. Revisiting Code Ownership and its Relationship with Software Quality in the Scope of Modern Code Review. n. 1, [s.d.].