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

2017

TIAGO KENJI UMEMURA

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

Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, 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

2017

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

Às **14:00** do dia **27 de novembro de 2017** foi realizada na sala **B101** da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação do(a) acadêmico(a) **Tiago Kenji Umemura** com o título **Uma ferramenta para monitoramento da entropia de mudança e sua relação com métricas de software**. Estavam presentes, além do(a) acadêmico(a), os membros da banca examinadora composta por: **Prof. Dr. Igor Scaliante Wiese** (orientador), **Prof. Dr. Lucio Geronimo Valentin** e **Profa. Dra. Aretha Barbosa Alencar**. Inicialmente, o(a) acadêmico(a) fez a apresentação do seu trabalho, sendo, em seguida, arguido(a) pela banca examinadora. Após as arguições, sem a presença do(a) acadêmico(a), a banca examinadora o(a) considerou _____ na disciplina de Trabalho de Conclusão de Curso **2** e atribuiu, em consenso, a nota _____ (_____). Este resultado foi comunicado ao(à) acadêmico(a) e aos presentes na sessão pública. A banca examinadora também comunicou ao acadêmico(a) que este resultado fica condicionado à entrega da versão final dentro dos padrões e da documentação exigida pela UTFPR ao professor Responsável do TCC no prazo de **onze dias**. Em seguida foi encerrada a sessão e, para constar, foi lavrada a presente Ata que segue assinada pelos membros da banca examinadora, após lida e considerada conforme.

Observações: _____

_____

_____

_____

_____

_____

Campo Mourão, **27 de novembro de 2017**

Prof. Dr. Lucio Geronimo Valentin	**Profa. Dra. Aretha Barbosa Alencar**
Membro 1	Membro 2

Prof. Dr. Igor Scaliante Wiese
Orientador

A ata de defesa assinada encontra-se na coordenação do curso.

Resumo

. Uma ferramenta para monitoramento da entropia de mudança e sua relação com métricas de software. 2017. 41. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2017.

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. Dado esse contexto, não existem ferramentas que ofereçam suporte para monitoramento da relação entre entropia e diferentes métricas de software, como por exemplo, número de autores que modificaram um arquivo, número de *commits*, *authorship* e *ownership*.
Objetivo: Implementar e avaliar 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.
Ferramenta: A ferramenta é dividida em coleta de dados, cálculo da entropia e das métricas, cálculo da correlação de *Spearman* e visualização de dados. Na coleta de dados, os dados são extraídos por meio da API Github e em seguida é realizado o cálculo da entropia, métricas de software e da correlação *Spearman* entre a entropia e as métricas. Na etapa de visualização de dados, os valores da entropia de mudança e das métricas de software são exibidos utilizando gráficos de *Treemap*, *Heat Map* e séries temporais. A ferramenta foi avaliada por meio da análise das visualizações geradas sobre o histórico do projeto Angular.
Resultados: Na análise da ferramenta, foi possível localizar arquivos mais relevantes do projeto, medindo o valor de entropia de mudança e gerando a visualização de *Treemaps* em diferentes períodos. Calculando a correlação de *Spearman* e analisando a visualização de *Heat Map* em diferentes períodos foi possível observar quais métricas apresentam maior nível de correlação com o valor de entropia dos arquivos.
Conclusões: Foi concluído que a ferramenta pode ser utilizada para encontrar os arquivos que merecem maior atenção dos desenvolvedores no projeto, devido ao valor alto da entropia e das métricas correlaciondas a entropia.

Palavras-chaves: Entropia. Fatores Sociais. Defeitos.

Abstract

. A tool for monitoring the change entropy and its relationship with software metrics. 2017. 41. f. Monograph (Undergradute Program in Computer Science), Federal University of Technology – Paraná. Campo Mourão, PR, Brazil, 2017.

Context: The change entropy is a measure to indicate how much a software suffers changes over a certain period of time. Researches have shown that the increase in entropy can cause disorder in the development process and may lead to an increase in number of software defects. Given this context, there are no tools support for monitoring the relationship between entropy and different software metrics, such as number of authors who have modified a file, number of commits, authorship and code ownership.
Objective: Implement and evaluate a tool that enables the monitoring of entropy and software metrics from projects stored on Github to help developers in project management.
Tool: The tool is divided into data extraction, calculation of entropy and metrics, calculation of the correlation of Spearman and data visualization. In data extraction, the data is extracted from the Github API and then the entropy and software metrics are calculated. Then Spearman correlation between entropy and metrics are calculated. In the data visualization step, the change entropy values and the software metrics will be displayed using Treemap, Heat Map and time series. The tool was evaluated by analyzing the views generated from the Angular project data, available on Github.
Results: In the analysis of the tool it was possible to find more relevant files of the project, measuring the value of change entropy and generating the visualization of Treemap in different periods of time. Calculating the correlation of Spearman and analyzing the Heatmap view in different periods it was possible to observe which metrics present a higher level of correlation with the entropy value of the files.
Conclusions: It was concluded that the tool can be used to find the files which deserve more attention from the developers in the project, due to the high entropy value and metrics correlated with entropy.

Keywords: Entropy. Social Metrics. Defects

Lista de figuras

Lista de tabelas

Sumário

Capítulo 1

Introdução

Projetos de *softwares* de grande porte, como por exemplo, Angular e Ruby On Rails, são desenvolvidos por várias pessoas e são modificados ao longo do tempo, uma vez que mudanças contínuas são necessárias para corrigir defeitos ou para atender a novos requisitos do projeto.

Para quantificar o impacto das mudanças contínuas, os pesquisadores tem utilizado o conceito de entropia de mudança (Hassan, 2009). 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 uma maior quantidade de mudanças está relacionada com o aumento do valor da entropia e consequentemente está relacionado com maior tendência do software apresentar defeitos. Canfora *et al.* (2014) 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 e concluiu que esses fatores estão relacionados ao aumento na entropia.

Dada a importância da entropia de mudança, é necessário que desenvolvedores e gerentes possam monitorar os valores de entropia dos arquivos e a possível relação do aumento da entropia com as métricas de software. Uma vez que métricas de software são usadas como indicadores de avaliação da qualidade e evolução do projeto.

Na revisão da literatura não foram encontrados estudos que tenham propostos ferramentas que possibilitem o monitoramento da relação da entropia com as métricas de softwares. Portanto, não existe uma ferramenta que possibilite o monitoramento dos efeitos da entropia e a sua relação com métricas sociais, de autoria e de processo.

Assim, o objetivo deste trabalho é construir uma ferramenta para coletar dados sobre o histórico dos projetos e permitir o monitoramento dos valores de entropia e a sua relação com as métricas sociais (número de comentários em *Pull Request* e número de *Pull Request*), de

autoria de mudança (*authorship*, *ownership* e experiência) e métricas de processo (quantidade de *commits*, quantidade de defeitos, quantidade de linhas removidas, quantidade de linhas adicionadas, *Code Churn*).

A ferramenta disponibiliza visualizações das métricas de software para auxiliar os desenvolvedores na tomada de decisões durante o desenvolvimento de um projeto. Foi utilizado o *Treemap* para mostrar quais arquivos do projeto possuem maior entropia, o *Heat Map* para visualizar o nível de correlação entre as métricas e a entropia, e gráficos de séries temporais que mostram o comportamento dos valores das métricas ao longo do tempo.

A avaliação da ferramenta foi feita no projeto Angular[1]. Foram analisados como os valores das métricas são apresentados nas visualizações implementadas e como esses valores se alteram em diferentes períodos da evolução do projeto. Foram considerados períodos de 2 anos, 4 anos e 7 anos. Os períodos de 2 anos e 4 anos foram divididos em 8 intervalos iguais de tempo e o período de 7 anos foi dividido em 7 intervalos.

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 ferramenta e o método de avaliação da ferramenta. O capítulo 4 apresenta os resultados. Por fim, o capítulo 5 apresenta a conclusão.

[1] O projeto angular é encontrado no endereço https://github.com/angular/angular.js

Capítulo 2

Referencial Teórico

Este capítulo apresenta os conceitos de entropia, métricas sociais, métricas de autoria, métricas de processo e visualização de dados.

2.1. Entropia de mudança

A entropia definida por 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 de mudança é uma medida para mensurar a quantidade de mudanças que ocorreram em um determinado espaço de tempo nos arquivo de um projeto.

Dado um sistema de software S composto por um conjunto de arquivos $f_1, f_2, ..., f_n$, a entropia de uma atividade de mudança em que cada arquivo f_i sofreu um certo número de mudanças $chg(f_i)$ é definido como (Hassan, 2009):

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

Assim, a probabilidade de um arquivo mudar em determinada atividade é estimada pela razão entre o número de mudanças no arquivo $chg(f_i)$ e o número total de mudanças na atividade.

As mudanças consideradas podem ser obtidas a partir da quantidade de linhas modificadas em um intervalo de tempo ou utilizando número de *commits*.

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, na qual os círculos pretos indicam quando ocorreu uma mudança no arquivo.

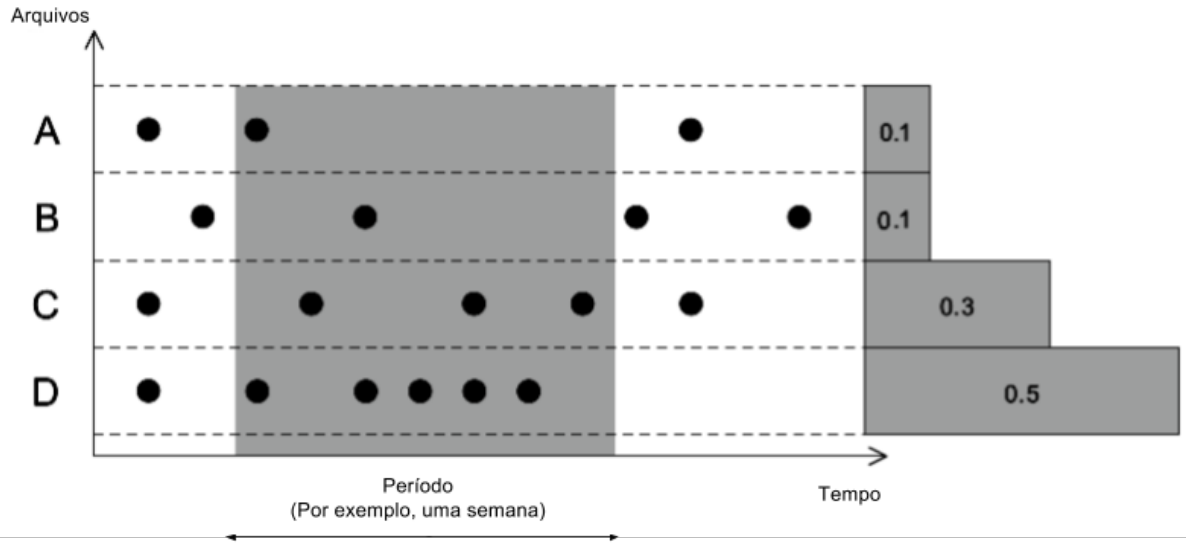

Figura 2.1. Exemplo de cálculo da entropia de mudança por arquivo extraído de Hassan (2009).

Para cada arquivo no sistema, é feito a contagem de quantas vezes ele foi modificado em um período e depois é dividido pelo total de mudanças que ocorreram nesse mesmo período considerando todos os arquivos. Por exemplo, na figura 2.1 como ocorreram dez mudanças no período selecionado e uma mudança no arquivo A nesse mesmo período, 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 de software e foi feito um modelo básico de mudança de código (BCC - *Basic Code Change Model*) para mensurar a complexidade de mudança em um tempo fixo, com um número fixo de arquivos.

No modelo básico de mudança de código são utilizados 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 fixo e a limitação do número fixo de arquivos é utilizado o modelo estendido de mudança de código (ECC).

No ECC, o histórico de mudanças é dividido em períodos de tamanhos iguais de 3 meses. Entretanto, a 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*).

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 *et al.* (2014) indicam que a entropia de mudança está relacionada com atividades de refatorações, número de contribuidores de cada arquivo e com padrões de projeto. A entropia 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.

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 (Koscianski; Soares, 2007). Existem várias métricas propostas na literatura e para este trabalho foram selecionadas métricas relacionadas aos *pull requests*, *commits* e linhas modificadas dos arquivos. As métricas foram categorizadas em métricas de autoria, sociais e de processo.

As métricas de autoria medem a contribuição do desenvolvedor nos arquivos do projeto, enquanto as métricas sociais medem a interação entre os desenvolvedores durante a modificação dos artefatos. Já métricas de processo, mensuram as mudanças que ocorreram em relação aos artefatos criados 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 em um projeto de software. Essa medida pode ser calculada por arquivo ou pacote do projeto.

O *Authorship* pode ser mensurado de várias formas: contando número de arquivos que o desenvolvedor modificou, número de *commits* e ou 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. Na pesquisa de Rahman e Devanbu (2011), é utilizado o termo *Implicated code* para indicar o número de linhas modificadas para corrigir um determinado erro no módulo de software.

2.2.1.2. *Ownership*

No trabalho de Greiler e Herzig (2015), 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* aquele que contribuiu com mais de 5%.

Considerando os valores de *authorship* de todos os autores de um arquivo do projeto, o maior valor de *authorship* será o valor de *ownership* do arquivo (Foucault *et al.*, 2015).

2.2.1.3. Experiência

Para calcular o nível de experiência do contribuidor é computada o número de linhas modificadas pelo contribuidor em um determinado espaço de tempo (Rahman; Devanbu, 2011).

Na pesquisa de Rahman e Devanbu (2011), a experiência é dividida em dois tipos: a experiência especializada e experiência geral. A experiência especializada é medida considerando o quanto um indivíduo contribui em um determinado arquivo e a experiência geral é a mesma medida,porém, considerando 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*.

As *issues* são criadas pelos desenvolvedores e tem como objetivo relatar problemas encontrados no projeto. Os *pull requests* são sugestões para resolver uma determinada *issue* e podem assumir dois estados: *merged* e *unmerged*. O *pull request unmerged* é uma contribuição que está em análise e pode ou não ser aceito e o *pull request merged* a contribuição foi aceita.

A Tabela 2.1 contém os nomes das métricas sociais utilizadas e suas descrições.

Tabela 2.1. Tabela de métricas sociais

Métricas	**Descrição**
Número de *Pull Requests merged* por arquivo	Representa o número total de *Pull Request merged* para cada arquivo modificado do projeto.
Número de *Pull Request unmerged* por arquivo	Representa o número total de *Pull Requests unmerged* para cada arquivo modifcado do projeto.
Número de comentários em *Pull Request unmerged*	Representa o número total de comentários em todos os *Pull Request unmerged* para cada arquivo modificado do projeto
Número de comentários em *Pull Request merged*	Representa o número total de comentários em todos os *Pull Request merged* para cada arquivo modificado do projeto

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. A Tabela 2.2 possui o nome das métricas de processo e as suas descrições, é descrita a seguir:

Tabela 2.2. Tabela de métricas de processo

Métricas	**Descrição**
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 perído de tempo.
Quantidade de defeitos	A quantidade de defeitos é calculada analisando a mensagem do *commit*, se a mensagem possui palavras relacionadas a defeito, então é contado um defeito para todos os arquivos do *commit*.
Quantidade de linhas removidas	Essa medida representa a quantidade de linhas removidas em cada arquivo.
Quantidade de linhas adicionadas	Essa medida representa a quantidade de linhas adicionadas em cada arquivo.
Quantidade de linhas modificadas	Essa medida representa a quantidade de linhas modificadas em cada arquivo.

2.3. Visualização de dados

A visualização de dados é uma área da computação que estuda maneiras de representar dados visualmente.

Um volume grande de dados apresenta dificuldades para se obter um entendimento abrangente e mostrar de forma acessível o seu significado. Assim, as visualizações são necessárias, como uma forma de exibição de dados, para fornecer respostas claras e acessíveis sobre um conjunto complexo de dados (Fry, 2008).

As visualizações de dados, foram utilizadas nesse trabalho com o objetivo de facilitar a compreensão sobre o comportamento dos valores das métricas de software ao longo do tempo. As visualizações foram geradas com os dados coletados sobre o histórico de 7 anos do projeto Angular.

Foram utilizadas 3 tipos de visualizações, para mostrar os valores das métricas ao longo do tempo, a *Treemap*, séries temporais e o *Heat map*.

2.3.1. *Treemap*

A *Treemap* é uma forma de visualização de dados hierárquica e utiliza a estrutura de árvore, em que cada elemento pode ter vários sub-elementos (Fry, 2008). Arquivos e diretórios, são os exemplos mais comuns de estrutura de árvore. Cada diretório pode possuir vários itens, que podem ser arquivos ou outros diretórios.

Na *Treemap*, a estrutura de árvore é mapeada para uma representação em duas dimensões. O projeto *Newsmap* de Marcos Weskamp[1] é um exemplo recente do uso da técnica, em que utiliza a *Treemap* para agrupar notícias em grupos facilmente reconhecíveis.

Uma das principais características da *Treemap* é que o usuário pode clicar em qualquer parte do *display* para ver mais detalhes de uma parte específica da estrutura. Além disso, os elementos da *Treemap* apresentam o tamanho relativo a alguma característica dos dados que estão sendo representados. Além disso, também é possível mapear a cor dos elementos da *Treemap* para um determinado atributo dos dados. Por exemplo, a figura 2.2 mostra o total em dinheiro de vendas de bicicletas por país, quanto maior valor das vendas maior será o elemento na *Treemap*. Nesse caso, cada cor representa um país[2].

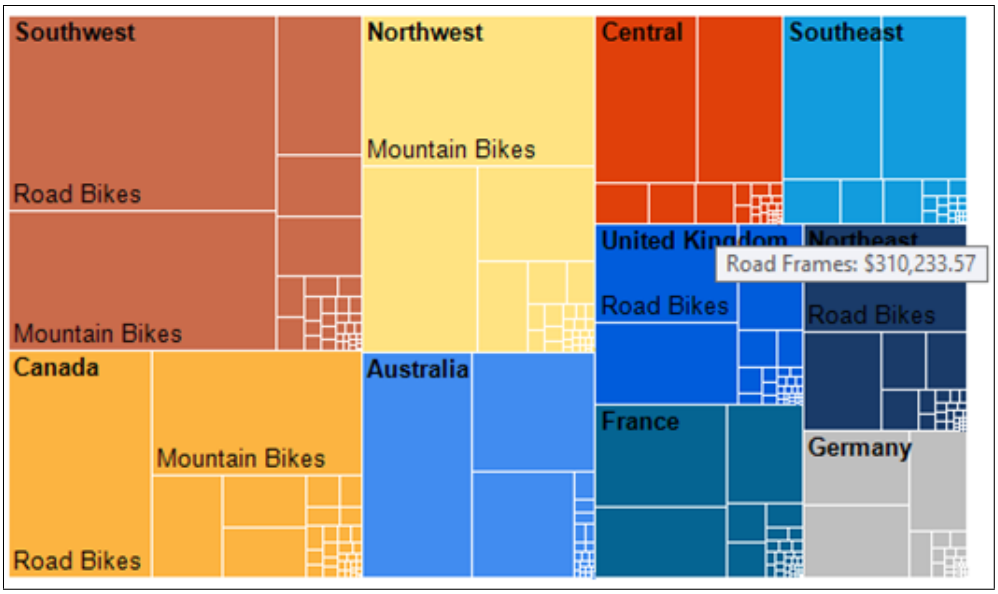

Figura 2.2. Exemplo de *Treemap*

Para esse trabalho, a *Treemap* foi utilizada para representar o valor de entropia de arquivos e pacotes de projetos em intervalos selecionados pelo usuário. Portanto, o tamanho dos elementos da visualização são relativos ao valor da entropia, quanto maior o valor da

[1] http://newsmap.jp/
[2] https://docs.microsoft.com/pt-br/sql/reporting-services/report-design/tree-map-and-sunburst-charts-in-reporting-services

entropia maior será o espaço que o elemento ocupa na representação.

2.3.2. Séries temporais

Séries temporais são um tipo de gráficos de linha que descreve como uma função mensuravél, por exemplo, população ou número de vendas, mudou durante um determinado período de tempo (Fry, 2008). A figura 2.3 mostra um exemplo de série temporal[3].

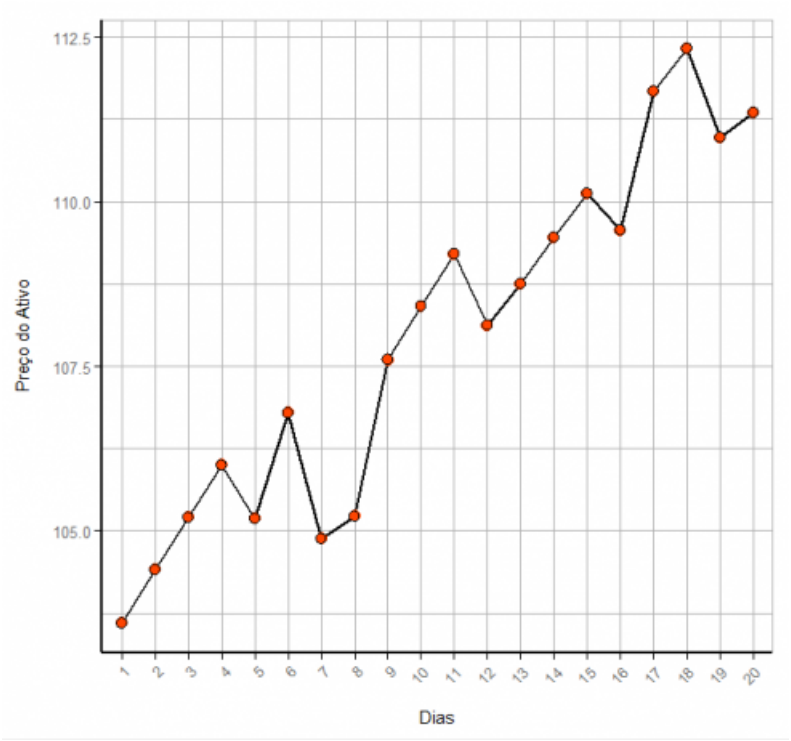

Figura 2.3. Exemplo de séries temporais

No trabalho, as séries temporais foram utilizadas para monitorar os valores das métricas a cada 15 dias, dentro de um período determinado pelo usuário. Para evitar a oclusão visual foram utilizados filtros para limitar a quantidade de métricas que são representadas no gráfico.

2.3.3. *Heat Map*

A visualização *Heat Map* é uma representação gráfica que utiliza cores em uma tabela, para mostrar as relações entre os valores dos dados[4]. Essa visualização será utilizada em forma de matriz, onde as linhas e colunas representarão as métricas e as cores nas intersecções

[3] https://support.minitab.com/pt-br/minitab/18/help-and-how-to/modeling-statistics/time-series/how-to/decomposition/before-you-start/example/
[4] http://www.businessdictionary.com/definition/heatmap.html

representarão o nível de correlação entre as métricas e entre a entropia de mudança e as métricas. No *Heat Map*, a cor vermelha é utilizada para correlações fortes e crescentes, a cor azul para correlações fortes e decrescentes e a cor cinza para o nível de correlação baixo. A figura 2.4 mostra um exemplo de *Heat Map*[5].

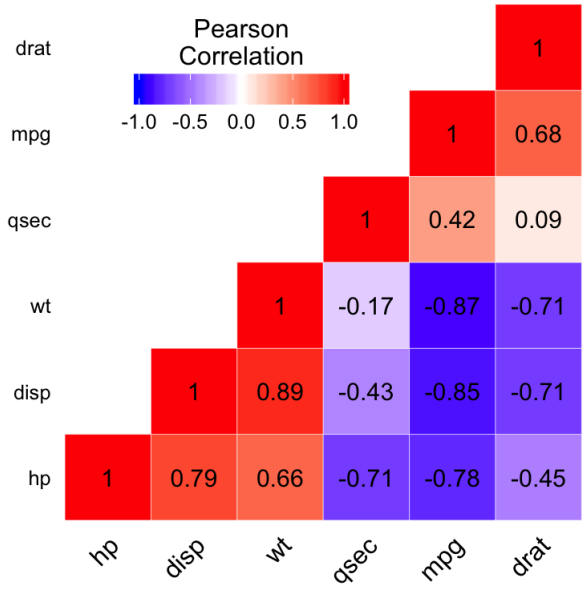

Figura 2.4. Exemplo de *Heat Map*

[5] http://www.sthda.com/english/wiki/ggplot2-quick-correlation-matrix-heatmap-r-software-and-data-visualization

Capítulo 3

Ferramenta: EntropyMetricsView

Nos estudos realizados por Hassan (2009) e Canfora *et al.* (2014) foram analisados a relação da entropia de mudança com a qualidade do projeto e algumas métricas de software, como por exemplo, número de contribuidores. No entanto, não há uma ferramenta que monitore a entropia e a sua relação com métricas sociais, de autoria e de processo.

Nesse trabalho foi implementado uma ferramenta, a *EntropyMetricsView*, para monitorar o valor da entropia de mudança e sua relação com as métricas sociais, de autoria e de processo, utilizando diferentes visualizações de dados para o monitoramento.

A visualização de dados tem como objetivo facilitar a compreensão e interpretação de grandes quantidades de dados. Na ferramenta as visualizações de dados tem como objetivo facilitar a análise das métricas de software de grandes projetos do Github em diferentes intervalos de tempo. Para visualizar os dados sobre os arquivos e métricas foram escolhidos 3 visualizações: *Treemap*, séries temporais e *Heatmap*.

Para o desenvolvimento da *EntropyMetricsView* foi utilizado a tecnologia Java Server Pages, biblioteca Javascript Google Charts[1] e o banco de dados MySql.

A página JSP chama um arquivo Java para coletar os dados com a API do Github em Java e persistir os dados em um banco de dados local. Foi desenvolvido um webservice em Java para permitir a comunicação com o banco de dados via JSON, pois a página JSP também utiliza código Javascript para gerar as visualizações.

Dessa forma, é feita uma requisição HTTP via Javascript para o webservice que retorna o JSON com os dados do projeto e assim permite que as visualizações sejam geradas pela biblioteca javascript. O código da ferramenta está disponível no Github[2].

[1] Documentação disponível em https://developers.google.com/chart/
[2] Disponível no link https://github.com/TiagoUmemura/EntropyMetricsView

A tecnologia Java Server Pages e o Webservice em java foram utilizados no desenvolvimento da ferramenta, devido a possibilidade para permitir o acesso a ferramenta em um *link* externo.

A figura 3.1 mostra o diagrama da arquitetura da ferramenta.

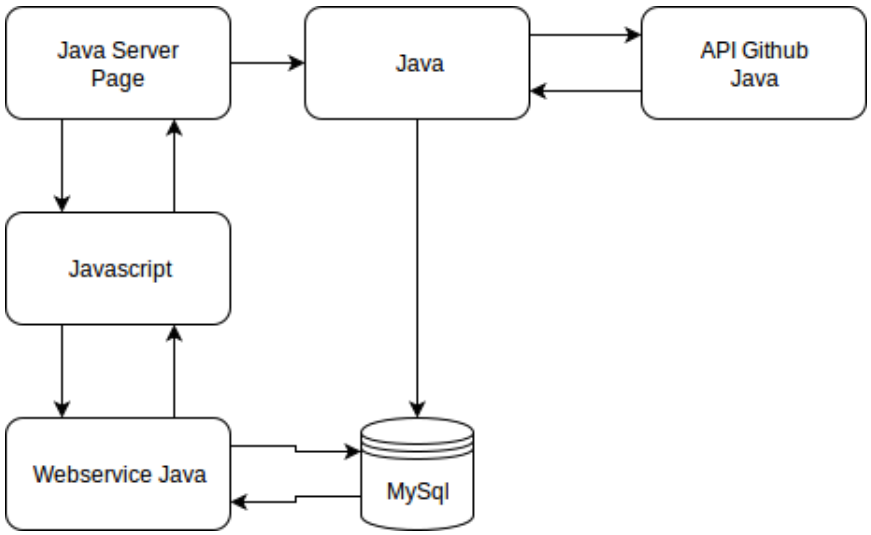

Figura 3.1. Arquitetura da ferramenta

A figura 3.2 mostra o modelo entidade relacionamento do banco de dados. Cada projeto possui vários *commits* e cada *commit* possui vários arquivos. Cada projeto também possui vários *Pull Requests* e cada *Pull Request* possui vários *commits* e vários arquivos.

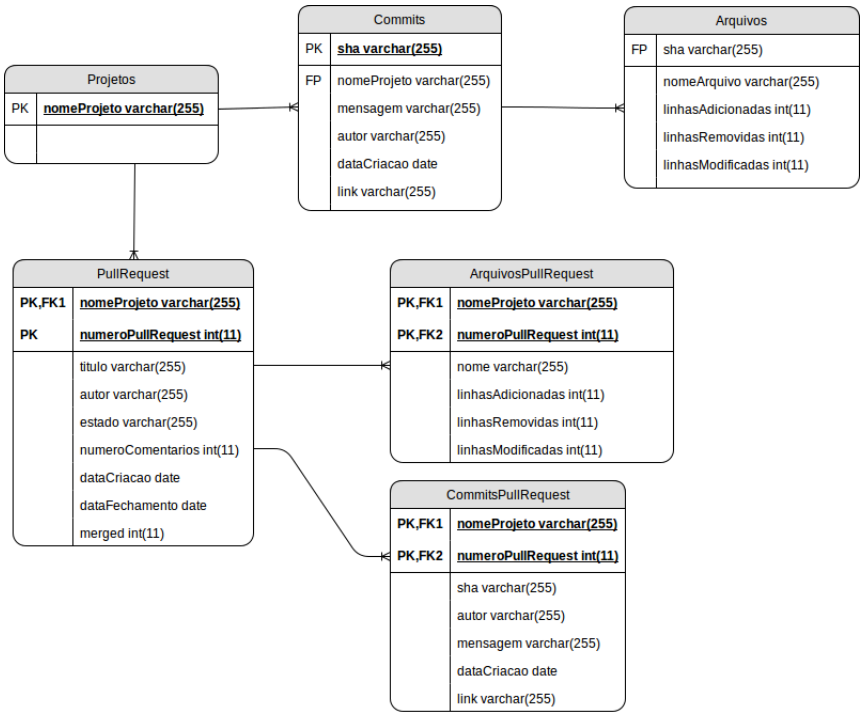

Figura 3.2. Modelo Entidade Relacionamento do banco de dados

A *EntropyMetricsView* possui seis módulos que são apresentados no fluxograma da Figura 3.3. Os módulos são apresentados nas próximas seções.

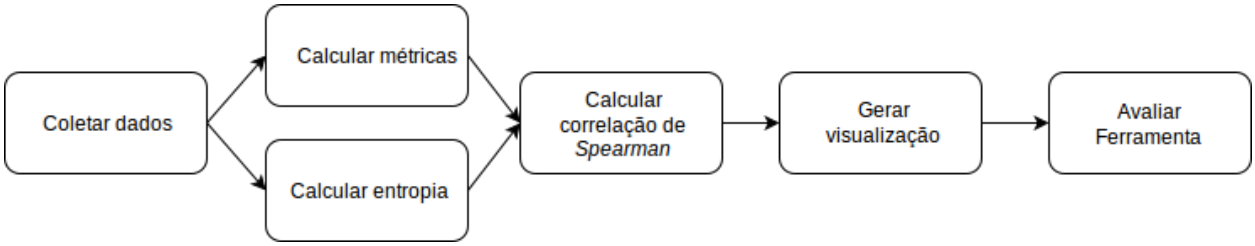

Figura 3.3. Fluxograma do funcionamento da ferramenta

As funcionalidades da ferramenta são apresentadas no diagrama de caso de uso na Figura 3.4.

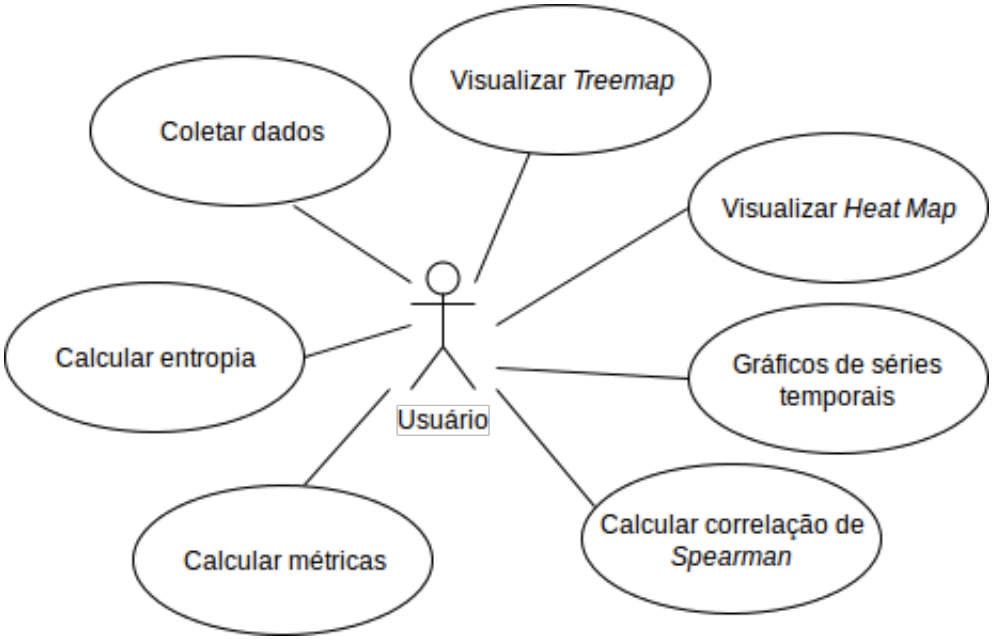

Figura 3.4. Diagrama de caso de uso da ferramenta

3.1. Coletar dados

O módulo de coleta de dados é responsável pela extração de dados dos projetos do Github e Git que são persistidos em um banco de dados local (MySql). Os dados são obtidos utilizando a API Github[3] em java.

Para extrair os dados de um projeto é necessário informar o nome do usuário, dono

[3] API Github permite coletar dados sobre os repositórios do Github, onde os dados são recebidos no formato JSON. A documentação da API pode ser encontrada no endereço: https://developer.github.com/v3/

do projeto, e o nome do projeto. Após finalizar a coleta de dados, o projeto passa a estar disponível na lista de projetos nas abas de projetos da ferramenta.

A Figura 3.5, mostra a aba de coleta de dados da ferramenta, onde usuário informa os campos usuário e nome do projeto e ao clicar no botão buscar, os dados são coletados e persistidos no banco de dados local.

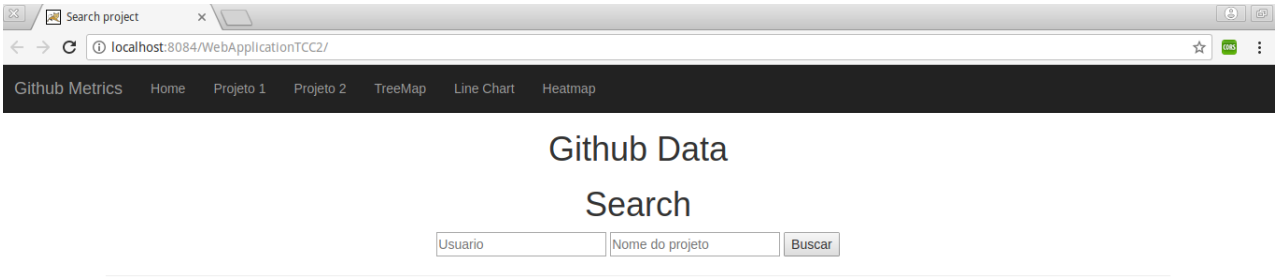

Figura 3.5. Aba de coleta de dados.

Para a avaliação da *EntropyMetricsView*, foram coletados os dados do projeto AngularJS, um *framework* Javascript que está na lista dos 100 maiores projetos do Github. Não foi utilizado outros projetos devido ao tempo de coleta de dados, por exemplo, o projeto *Ruby On Rails* possui 65.390 *commits*, 19.640 *pull requests* e a coleta dos dados dos arquivos de cada *commit* e *pull request* exigiria muito tempo, devido ao limite de 5000 requisições por hora da API do github.

O Angular foi usado como estudo de caso, pois, possui 7 anos de histórico para a coleta de dados, que compreendem 8.622 *commits* e 7.454 *pull requests*. Estes dados foram coletados nos dias 27 de outubro de 2017 e 28 de outubro de 2017.

3.2. Calcular entropia

Este módulo realiza o cálculo da entropia de cada arquivo do projeto no período que o usuário informar. O valor da entropia é determinado utilizando como medida o número de *commits* que o arquivo teve durante o período de tempo selecionado pelo usuário.

Para o cálculo da entropia é necessário fazer a contagem de quantos *commits* cada arquivo obteve no período e dividir pelo número de mudanças, onde cada *commit* em um arquivo é considerado uma mudança, por exemplo, se um *commit* tem três arquivos modificados então ocorreram três mudanças.

Na *EntropyMetricsView*, a entropia de mudança é calculada assim que o usuário selecionar o projeto a ser analisado e informar o intervalo de datas.

3.3. Calcular métricas

O módulo de cálculo das métricas, realiza o cálculo após a extração dos dados no módulo de coleta de dados.

O cálculo das métricas e o cálculo da entropia são feitos simultaneamente. Assim que o usuário informar o projeto e o intervalo, todas as métricas são calculadas pela ferramenta para o período inteiro e dividido em períodos menores, de 15 dias.

A forma como as métricas são calculadas estão descritas na Tabela 3.1.

Tabela 3.1. Tabela de cálculo das métricas

Métrica	**Descrição do cálculo**
Authorship	Quantidade de *commits* de cada contribuidor
Ownership	Maior valor da métrica de *authorship*
Experiência	Quantidade de linhas modificadas no arquivo de cada contribuidor
Pull Request Merged por arquivo	Contagem do número de *Pull Request Merged* relacionados a um determinado arquivo
Pull Request Unmerged por arquivo	Contagem do número de *Pull Request Unmerged* relacionados a um determinado arquivo
Comentários em *Pull Request Unmerged*	Contagem do número de comentários em todos os *Pull Request Unmerged* relacionados ao arquivo
Comentários em *Pull Request Merged*	Contagem do número de comentários em todos os *Pull Request Merged* relacionados ao arquivo
Quantidade de *commits*	Quantidade de total de *commits* do projeto e de cada arquivo no período informado pelo usuário
Quantidade de defeitos	Verificar se a mensagem do *commit* possui termos relacionados a defeitos
Quantidade de linhas removidas	Quantidade total de linhas removidas de cada arquivo no período informado pelo usuário
Quantidade de linhas adicionadas	Quantidade total de linhas adicionadas de cada arquivo no período informado pelo usuário
Code Churn	Quantidade total de linhas modificadas de cada arquivo no período informado pelo usuário

3.4. Calcular correlação entre as métricas

Após o cálculo das métricas para cada arquivo do projeto no período informado, é utilizado a correlação de *Spearman* para medir o nível de correlação entre todas as métricas.

Os coeficientes de correlação de *Spearman* podem variar -1 a +1. Quando uma variável aumenta e a outra variável aumenta de forma consistente então o coeficiente é um positivo, se uma variável aumenta e a outra variável diminui então o coeficiente é um negativo e quando a relação é aleatória então o coeficiente se aproxima de zero.

Os coeficientes obtidos da correlação entre as métricas, são utilizados no módulo de visualização para preencher a matriz da visualização de *Heat Map*.

3.5. Gerar Visualização

Nesse módulo são geradas as visualizações com os valores da entropia e das métricas de *software* calculadas nos módulos anteriores. Para visualização, foi utilizado as técnicas *Treemap*, *Heatmap* e séries temporais.

A *EntropyMetricsView* possui 5 abas para mostrar e comparar as visualizações. Na aba Projeto 1, o usuário seleciona um projeto e informa o intervalo de data que deseja calcular as métricas, ao clicar em calcular são geradas todas as visualizações. Na aba Projeto 2, o usuário pode gerar novas visualizações para intervalos diferentes, e assim, comparar com as visualizações da aba Projeto 1.

A Figura 3.6 mostra a aba Projeto 1, que apresenta 3 campos. O campo projeto para selecionar um projeto na *combo box*, campo para indicar a data inicial do intervalo de tempo e campo para indicar a data final do intervalo. Após preencher os campos e clicar no botão calcular, as métricas são calculadas e em seguida são geradas as visualizações

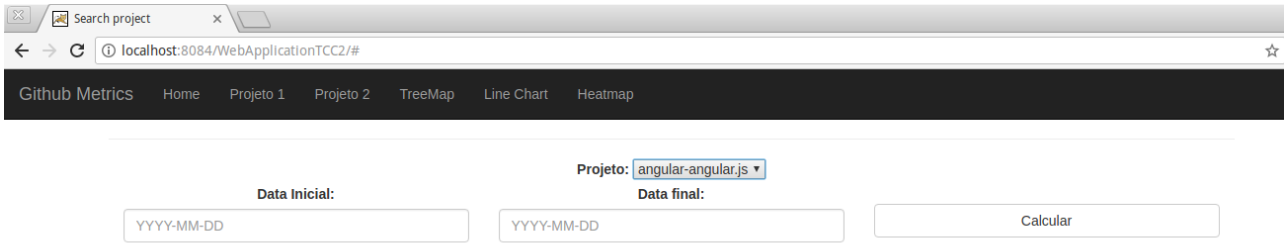

Figura 3.6. Aba projeto 1.

Após gerar as visualizações nas abas Projeto 1 e Projeto 2, ao clicar na aba *Treemap* apenas as *Treemaps* são mostradas, na aba *Heat Map* apenas as visualizações de *Heat Map* são mostradas e na aba *Line Chart* são mostradas apenas as séries temporais.

A *Treemap* é utilizada para visualizar os arquivos do projeto ou os pacotes do projeto, o tipo de visualização pode ser alterada ao clicar em um botão localizado embaixo da *Treemap*. Os arquivos com o valor de entropia mais alto, são representados com a cor vermelha e arquivos com valor de entropia mais baixo, são representados com a cor verde. O tamanho dos elementos da *Treemap* também representa o valor de entropia dos arquivos ou pacotes.

A Figura 3.7 apresenta a visualização por pacote, onde é possível ver todas as pastas e subpastas onde há arquivos que foram modificados no intervalo selecionado. Na visualização por pacote, o usuário pode explorar cada pasta apresentada ao clicar no *display* da pasta que se deseja ver mais detalhes. A cor das pastas são determinadas pelo nível médio de entropia dos arquivos que elas contêm.

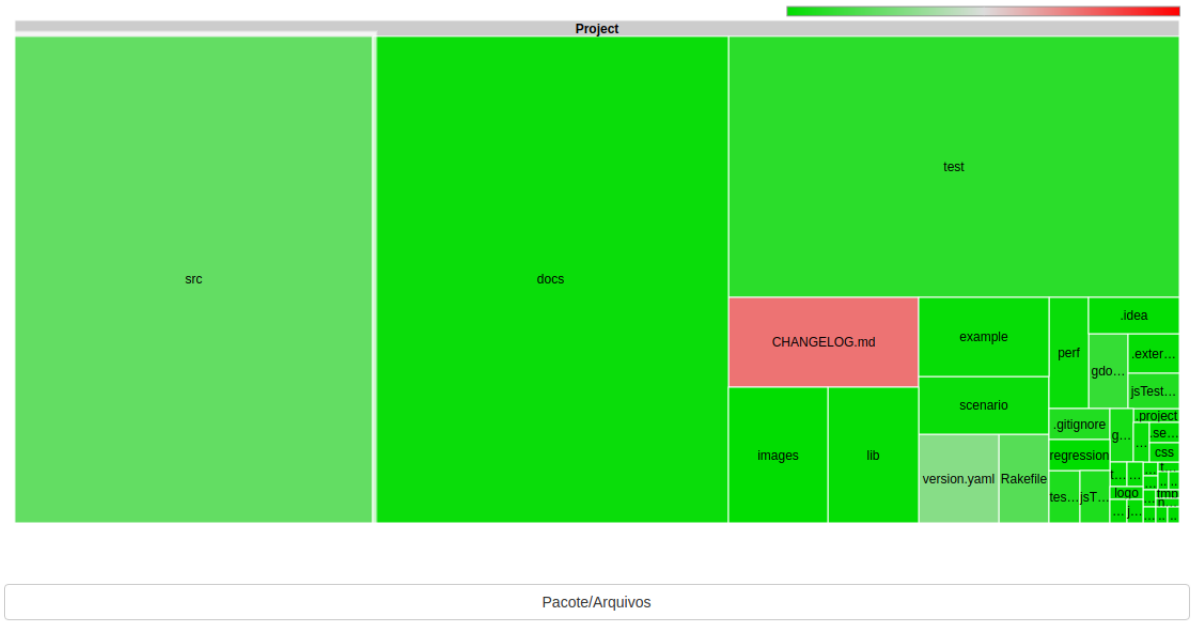

Figura 3.7. Exemplo de visualização *Treemap* por pacote.

A Figura 3.8 apresenta a visualização da *Treemap* por arquivo, onde os arquivos com cores verdes representam arquivos com baixa entropia e os arquivos com cores vermelhas representam arquivos com alta entropia.

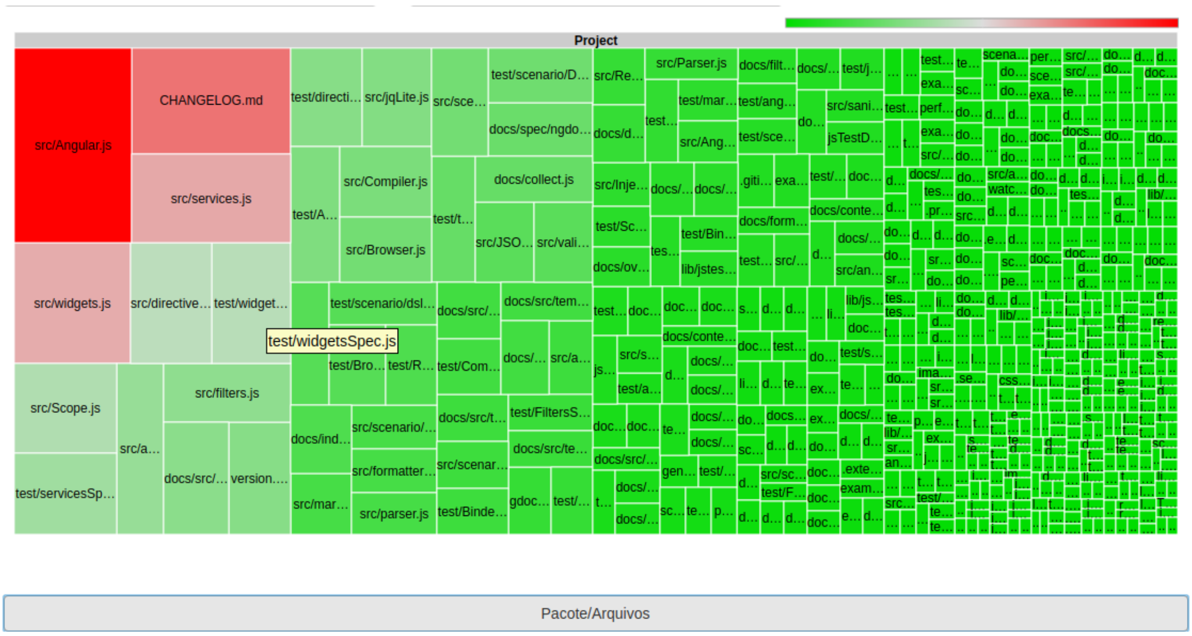

Figura 3.8. Exemplo de visualização *Treemap* por arquivo.

O gráfico de séries temporais é utilizado para mostrar a evolução das métricas ao longo do tempo. O eixo x do gráfico representa o tempo e o eixo y representa o valor das métricas, sendo que o tempo é dividido em intervalos de 15 dias.

Para evitar a oclusão visual nas séries temporais, foram criados três filtros: métricas relacionadas a *commits*, relacionadas a *pull request* e relacionadas a número de linhas modificadas nos arquivos do projeto.

As métricas relacionadas aos *commits* são: a quantidade de arquivos que mudaram a cada 15 dias, quantidade de *commits*, quantidade de *commits* do *Owner* e quantidade de *commits* do contribuidor com maior *experience*. As métricas relacionadas aos *pull requests* são: o número de *pull requests merged* e *unmerged*, e número de comentários em *pull requests merged* e *unmerged*. As métricas relacionadas ao número de linhas modificadas no arquivo são as quantidades de linhas adicionadas, removidas e modificadas em arquivos em intervalos de 15 dias.

A Figura 3.9 mostra um exemplo de série temporal, cada intervalo no eixo x representa um intervalo de 15 dias. Abaixo do gráfico estão disponíveis 3 botões, um para visualizar gráficos de métricas dos *commits*, um para visualizar as métricas das linhas modificadas dos arquivos e um para visualizar as métricas de *pull requests*.

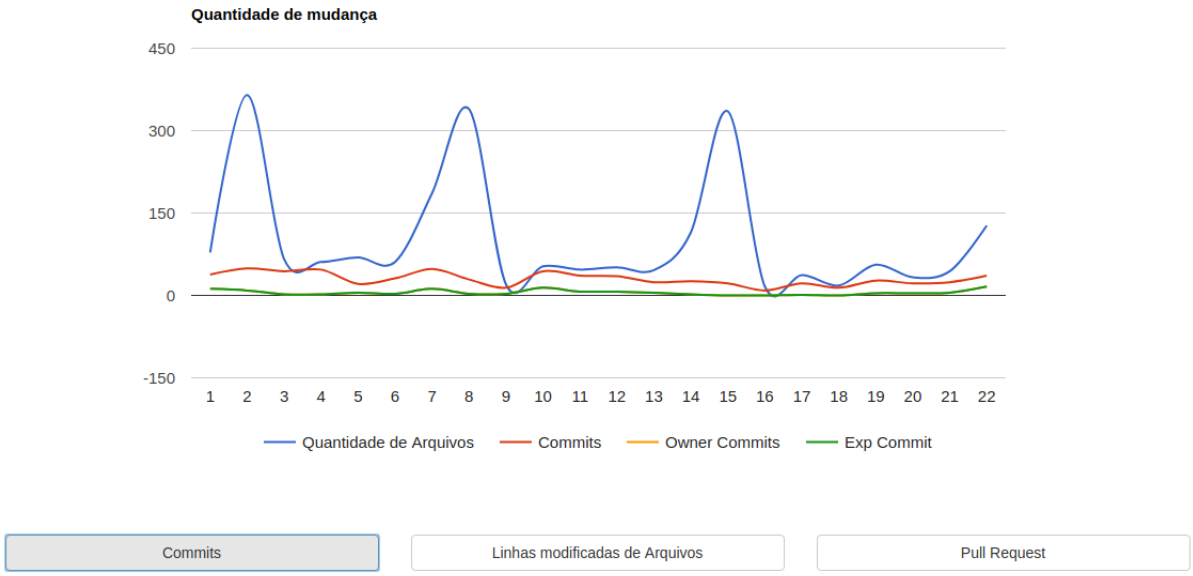

Figura 3.9. Exemplo de série temporal

Após gerar a *Treemap* e as séries temporais, é gerado uma tabela com todos os arquivos modificados no intervalo e os valores de todas as métricas de cada arquivo. A Figura 3.10 mostra a tabela com os valores das métricas. A primeira coluna apresenta os nomes do arquivos e as demais colunas apresentam os valores das métricas dos arquivos.

Search for names..

Arquivo	Commits	Add	Remove	Changed	Defects	P.R Merged	P.R Merged Comment	P.R Unmerged	P.R Unmerged Comment	Ow
src/ng/directive/ngModelOptions.js	2	11	6	17	1	0	0	3	2	
docs/content/misc/contribute.ngdoc	6	31	37	68	0	1	10	11	38	
docs/content/misc/faq.ngdoc	2	3	1	4	1	3	23	4	24	
Gruntfile.js	13	129	99	228	2	4	31	34	108	
lib/grunt/plugins.js	5	26	27	53	1	1	10	9	38	
package.json	25	70	68	138	4	8	26	33	113	
lib/grunt/utils.js	6	40	44	84	2	3	20	11	44	
yarn.lock	2	6803	17	6820	0	1	10	0	0	
scripts/check-size.sh	2	1	1	2	1	1	10	0	0	
scripts/travis/before_build.sh	2	4	4	8	0	1	10	7	12	
scripts/travis/build.sh	5	24	24	48	0	1	21	22	58	

Figura 3.10. Exemplo de tabela de métricas

A visualização de *Heat Map* é utilizada em forma de matriz, onde as linhas e colunas representam as métricas e as cores nas intersecções representam o nível de correlação entre as métricas e entre a entropia de mudança e as métricas. No *Heat Map* a cor vermelha é utilizada para correlações fortes e crescentes, a cor azul para correlações fortes e decrescentes

e a cor cinza para o nível de correlação baixo.

É gerado um *Heat Map* para cada intervalo informado pelo usuário. A Figura 3.11 apresenta um exemplo de *Heat Map* com as métricas de software do projeto Angular.

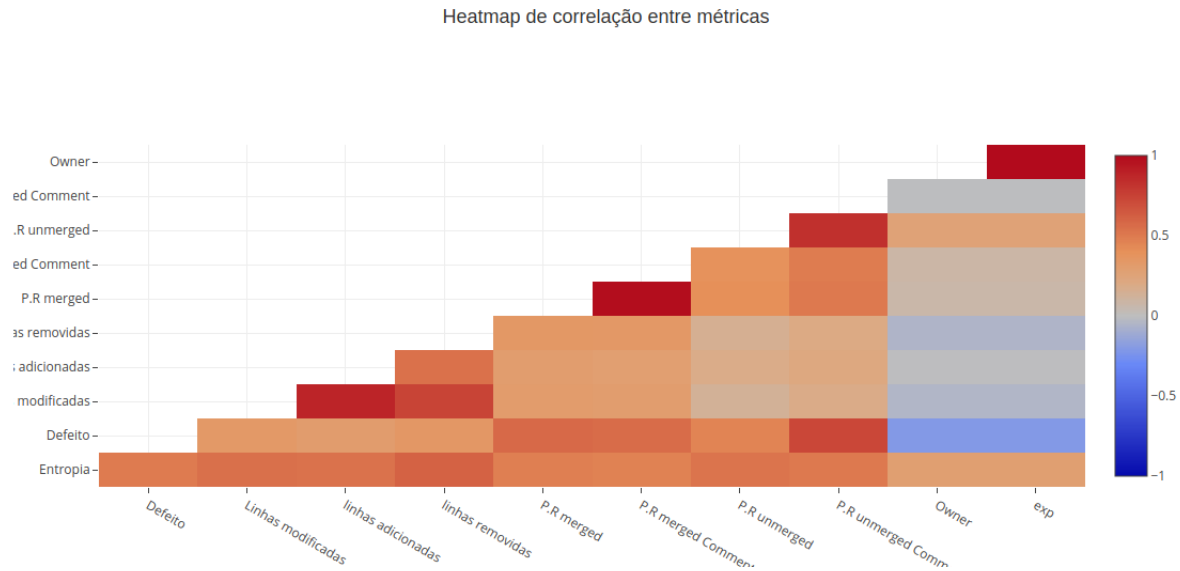

Figura 3.11. Exemplo de *Heat Map* com matriz

3.6. Avaliação da ferramenta

Para avaliação da *EntropyMetricsView* foram considerados cenários em que o usuário poderia utilizar a ferramenta para gerenciar e acompanhar as mudanças de um projeto qualquer. Assim o objetivo da avaliação foi simular possíveis cenários em que o usuário poderia acompanhar os valores de entropia de mudança e das métricas de software utilizando as visualizações da ferramenta.

Dois cenários foram utilizados, o comportamento da entropia de mudança ao longo do tempo e como as métricas se relacionam com a entropia.

3.6.1. Como a entropia se comporta ao longo do tempo?

Já foram feitas pesquisas que indicam que quanto maior a quantidade de mudanças em um arquivo, maiores as chances desse arquivo apresentar defeitos ao longo do tempo (Hassan, 2009). Portanto, para a análise do projeto Angular foi observado quais arquivos apresentavam maior quantidade de mudanças, podendo assim, apresentar maior influência na evolução e complexidade do projeto.

Para analisar o comportamento da entropia ao longo do tempo foi utilizado a visualização de *Treemap* com filtro por arquivos. Na análise da *Treemap* foi considerado os arquivos que apresentavam valor de entropia maior do que o valor médio de entropia dos arquivos.

Foram gerados visualizações para períodos de 2 anos, 4 anos e 7 anos. O período de 2 anos, entre 01/07/2015 e 01/07/2017, foi dividido em 8 intervalos de 3 meses. O período de 4 anos, entre 01/07/2013 e 01/07/2017, foi dividido em 8 intervalos de 6 meses. O período de 7 anos, entre 01/07/2010 e 01/07/2017, foi dividido em intervalos de 1 ano.

Para cada intervalo de cada período, foi gerado uma *treemap* e calculado o valor da entropia de cada arquivo modificado. Em seguida, foi calculado o valor máximo, valor mínimo, média, mediana e desvio padrão do valor de entropia dos arquivos nos 8 intervalos de cada período, no caso do período de 7 anos, os cálculos foram feitos considerando 7 intervalos. Além disso foi feito a contagem da quantidade de intervalos em que o arquivo apresenta valor de entropia maior do que a média.

3.6.2. Como as métricas se relacionam com a entropia?

Pesquisas indicam que métricas sociais, de processo (Bettenburg; Hassan, 2013) e de autoria (Rahman; Devanbu, 2011) estão relacionadas com a quantidade de defeitos que um projeto pode apresentar. Assim como a entropia de mudança, as métricas de software são indicadores que podem ser associados a existência de defeitos.

Portanto, a relação entre as métricas e a entropia podem indicar quais métricas de software tem maior correlação com a quantidade de defeitos, e assim o desenvolvedor tem mais indicadores para verificar quais arquivos que devem ter maior atenção ao serem modificados.

Para visualizar a correlação entre as métricas e a entropia foi utilizado o *Heat Map*, onde a cor na intersecção entre cada métrica correponde ao valor da correlação. O cálculo da correlação foi feito considerando os valores das métricas de todos os arquivos.

Foram gerados *Heat Maps* para períodos de 2 anos, 4 anos e 7 anos. O período de 2 anos, entre 01/07/2015 e 01/07/2017, foi dividido em 8 intervalos de 3 meses. O período de 4 anos, entre 01/07/2013 e 01/07/2017, foi dividido em 8 intervalos de 6 meses. O período de 7 anos, entre 01/07/2010 e 01/07/2017, foi dividido em intervalos de 1 ano.

Os níveis de correlação do *Heat Map* foram divididos em: muito forte, forte, moderada, fraca e desprezível[4]. Os intervalos de valores da correlação *Spearman* correpondente a cada nível de correlação são mostrados na Tabela 3.2.

[4] http://www.statisticssolutions.com/correlation-pearson-kendall-spearman/

Tabela 3.2. Níveis de correlação

Intervalo de valores	**Nível de correlação**
>0,9	Muito forte
0,7 a 0,9	Forte
0,5 a 0,7	Moderada
0,3 a 0,5	Fraca
0 a 0,3	Desprezível

Para a análise do *Heat Map*, em cada período foi feito a contagem da quantidade de vezes de como as correlações foram classificadas, por exemplo, a correlação entre a entropia e o número de defeitos foi classificada como forte em 1 dos 8 intervalos do período de 2 anos.

Capítulo 4

Resultados

Esta seção apresenta os resultados obtidos ao analisar o projeto AngularJS, com as visualizações geradas pela *EntropyMetricsView*. As visualizações foram geradas em intervalos de 3 meses, 6 meses e 1 ano.

4.1. Como a entropia se comporta ao longo do tempo?

A Tabela 4.1, apresenta os arquivos com o valor de entropia maior do que o valor médio de entropia do projeto durante o período de 2 anos, entre as datas de 01/07/2015 e 01/07/2017. Esse período foi dividido em 8 intervalos de 3 meses.

Nesse período, os arquivos CHANGELOG.MD e compile.js são os arquivos que apresentaram alto valor de entropia, em 6 dos 8 intervalos e são os arquivos que apresentaram as maiores médias do período, próximo a 0.020. O arquivo compileSpec.js apresenta alta entropia em 4 dos 8 intervalos e apresenta valor médio de entropia de 0.009, cerca de duas vezes mais baixa que dos arquivos CHANGELOG.md e compile.js. Os arquivos Angular.js, angular-mocks.js e AngularSpec.js apresentam o valor médio de entropia de 0.004, e tem alta entropia em 2 intervalos.

Os demais arquivos da Tabela 4.1, apresentam média dos valores de entropia próximo a 0.001, têm alta entropia em apenas 1 intervalo e desvio padrão é igual a 0.

Tabela 4.1. *Treemap* intervalo de 3 meses

Arquivos	**Intervalos**	**Max**	**Min**	**Média**	**Mediana**	**Desvio padrão**
Changelog.md	6	0,059	0,017	0,021	0,024	0,014
compile.js	6	0,040	0,014	0,019	0,026	0,007
compileSpec.js	4	0,022	0,013	0,009	0,018	0,002
Angular.js	2	0,019	0,018	0,004	0,018	0
angular-mocks.js	2	0,024	0,011	0,004	0,018	0,006
AngularSpec.js	2	0,019	0,014	0,004	0,018	0
package.json	1	0,034	0,034	0,002	0,034	0
animateCss.js	1	0,014	0,014	0,001	0,014	0
animateCssSpec.js	1	0,014	0,014	0,001	0,014	0
ngOptions.js	1	0,014	0,014	0,001	0,014	0
module.js	1	0,013	0,013	0,001	0,013	0
ngOptionsSpec.js	1	0,012	0,012	0,001	0,012	0
select.js	1	0,012	0,012	0,001	0,012	0
angularFiles.js	1	0,011	0,011	0,001	0,011	0

A Tabela 4.2, apresenta os arquivos com o valor de entropia maior do que o valor médio de entropia do projeto durante o período de 4 anos, entre as datas de 01/07/2013 e 01/07/2017. Esse período foi dividido em 8 intervalos de 6 meses.

Nesse período, o arquivo compile.js apresenta alta entropia em todos os intervalos, possui o maior valor médio de entropia entre os arquivos, maior desvio padrão e maior valor máximo. O arquivo CHANGELOG.md apresenta alta entropia em 7 intervalos e a segunda maior média do período. O arquivos angular.js possui apresenta entropia em 5 intervalos, a média é metade do valor da média de entropia do arquivo compile.js. Os arquivos package.json, input.js e compileSpec, possuem valores de média de entropia próximos a 0,007 e apresentam alta entropia em 3 intervalos, exceto o arquivo compileSpec.js, que apresenta alta entropia em 4 períodos.

Outros arquivos da Tabela 4.2, possuem valor médio de entropia abaixo de 0.004, apresentam alto valor de entropia em um ou dois períodos e tem valor de desvio padrão próximo ou igual a 0.

Tabela 4.2. *Treemap* intervalos de 6 meses

Arquivos	Intervalos	Max	Min	Média	Mediana	Desvio padrão
compile.js	8	0,035	0,010	0,019	0,017	0,007
changelog.md	7	0,026	0,013	0,016	0,018	0,004
angular.js	5	0,022	0,010	0,009	0,016	0,004
package.json	3	0,024	0,020	0,008	0,024	0,001
input.js	3	0,024	0,014	0,007	0,018	0,003
compileSpec.js	4	0,020	0,010	0,006	0,014	0,003
animate.js	2	0,025	0,013	0,004	0,019	0,005
angular-mocks.js	2	0,021	0,013	0,004	0,017	0,004
http.js	2	0,014	0,011	0,003	0,013	0,001
animateSpec.js	1	0,020	0,020	0,002	0,020	0
jqLite.js	1	0,014	0,014	0,001	0,014	0
select.js	1	0,011	0,011	0,001	0,011	0

A Tabela 4.3 apresenta os arquivos com o valor de entropia maior do que o valor médio de entropia do projeto durante o período de 7 anos, entre as datas de 01/07/2010 e 01/07/2017. Esse período foi dividido em 7 intervalos de 1 ano.

Nesse período, os arquivos angular.js, CHANGELOG.md e compile.js possuem as maiores médias de entropia, próximo a 0.012, também apresentam os maiores valores no desvio padrão e valor máximo, também apresentam alto valor de entropia em 5 dos 7 intervalos. Os arquivos angular-mocks.js, package.json, widgets.js e input.js possuem valor médio de entropia próximo a 0.005 e valor da mediana próximo a 0.015.

Outros arquivos da Tabela 4.3, possuem média próxima a 0.001, apresentam alta entropia em apenas 1 intervalo e têm valor de desvio padrão 0.

Tabela 4.3. *Treemap* intervalo de 1 ano

Arquivos	Intervalos	Max	Min	Média	Mediana	Desvio padrão
angular.js	5	0,040	0,012	0,012	0,016	0,010
changelog.md	5	0,029	0,012	0,012	0,020	0,005
compile.js	5	0,028	0,012	0,011	0,014	0,005
angular-mocks.js	4	0,015	0,009	0,006	0,013	0,002
package.json	3	0,022	0,008	0,005	0,016	0,005
widgets.js	2	0,024	0,012	0,004	0,018	0,006
input.js	2	0,018	0,016	0,004	0,017	0
animate.js	1	0,019	0,019	0,002	0,019	0
compileSpec.js	1	0,016	0,016	0,002	0,016	0
widgetsSpec.js	1	0,010	0,010	0,001	0,010	0
jqLite.js	1	0,010	0,010	0,001	0,0101442741	0
ngRepeat.js	1	0,009	0,009	0,001	0,009905894	0

Com base nas tabelas e visualizações geradas, foi observado que os arquivos CHANGELOG.md e compile.js apresentam valores altos de entropia na maioria dos intervalos de tempo considerados.

O arquivo CHANGELOG.md, um dos arquivos com maior valor de entropia ao longo do tempo, contém informações sobre as mudanças mais importantes para cada versão do projeto. Assim é necessário que esse arquivo sempre esteja atualizado sobre os novos requisitos e por consequência apresenta um grande número de mudanças. O arquivo CHANGELOG.md possui alto valor de entropia, entretando, não apresenta riscos para o projeto, já que a sua função é apenas registrar as mudanças ocorridas ao longo do tempo.

O arquivo compile.js[1] possui comentários alertando que mudanças feitas nesse arquivo podem levar falhas de segurança e que todo *commit* deve ser aprovado por dois membro que já modificaram esse arquivo, além disso, é um arquivo que apresenta o maior valor de entropia em vários dos intervalos analisados. Portanto esse pode ser um arquivo de grande relevância no Angular e contribuidores externos ao projeto podem ter dificuldades para modificá-lo.

Arquivos com nomes similares aos arquivos com grande número de mudanças, tendem a apresentar valor alto na entropia também, são os arquivos: compile.js, compileSpec.js, Angular.js, angular-mocks.js, animateCss.js, animeteCssSpec.js, animate.css, ngOptions e ngOptions.spec. Assim é provável que esses arquivos mudem juntos ao longo do tempo. Por exemplo, caso um desenvolvedor modifique o animateCss.js, pode ser necessário que ele também modifique o arquivo animateCssSpec.js.

4.1.1. Análise da entropia utilizando a ferramenta

Esta seção apresenta como a *EntropyMetricsView* foi utilizada para obter o valor de entropia dos arquivos com quantidades de mudanças maior do que a média do projeto.

Para obter o valor de entropia dos arquivos, foi necessário utilizar a visualização de *Treemap* por arquivo, em que os arquivos com valor de entropia maior do que a média do projeto aparecem pintados com a cor vermelha.

A Figura 4.1, mostra um exemplo em que os arquivos com os valores de entropia maior do que a média são apresentados com a cor vermelha. Na figura, os arquivos compile.js, CHANGELOG.md, compileSpec.js e angular-mocks.js apresentam valores maior do que a média. Ao colocar o cursor sobre a representação do arquivo, é possível ver na barra superior direita da *Treemap*, a relação entre o valor da entropia do arquivo e o valor máximo e minímo de entropia do projeto.

[1] O arquivo pode ser encontrado no endereço https://github.com/angular/angular.js/blob/master/src/ng/compile.js

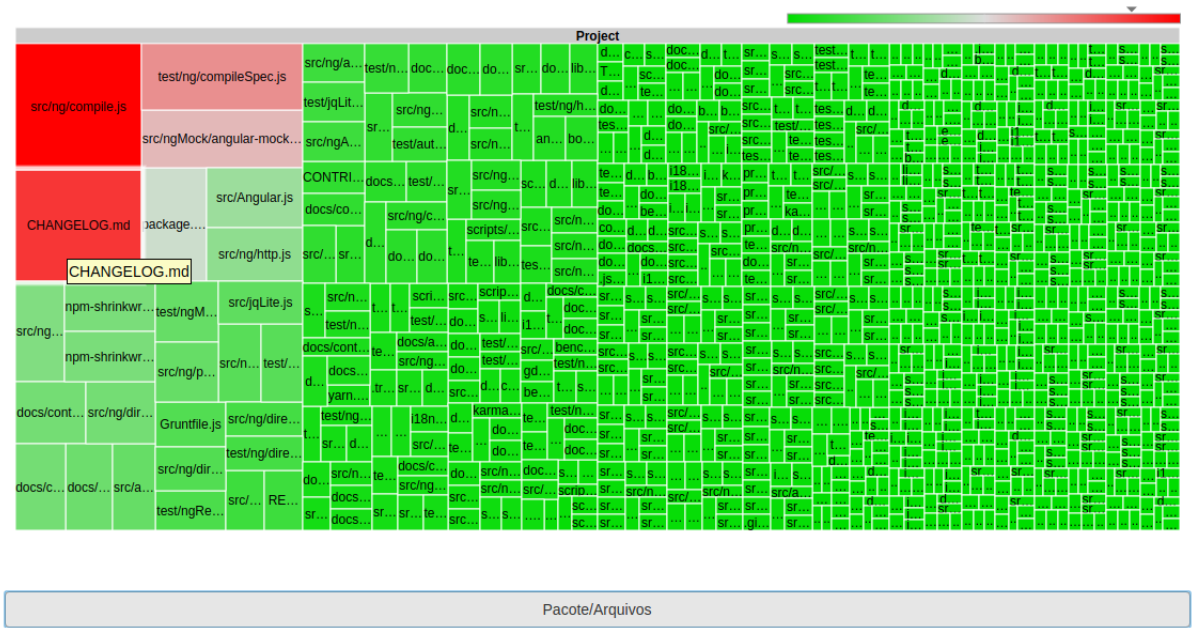

Figura 4.1. Exemplo de visualização *Treemap* por arquivo.

Para verificar o valor da entropia dos arquivos, é utilizado a tabela gerada pela ferramenta onde são apresentados todos os arquivos com os valores das métricas. A parte superior da tabela possui um campo de texto para filtrar a tabela pelo nome do arquivo. Assim, a tabela é filtrada com o nome dos arquivos desejados e, dessa forma, é obtido o valor de entropia dos arquivos representados na *Treemap*.

A figura 4.2 mostra um exemplo de tabela gerada pela ferramenta, utilizando como filtro a palavra *change*. Como resultado são mostrados apenas os arquivos que apresentam *change* no seu nome. Por exemplo, é possível localizar o valor de entropia do arquivo CHANGELOG.md.

change

Arquivo	Commits	Add	Remove	Changed	Defects	P.R Merged	P.R Merged Comment	P.R Unmerged	P.R Unmerged Comment	Ownership	Exp	Entropia
CHANGELOG.md	58	4458	460	4918	16	4	15	34	83	24	24	0.024421052631578948
changelog.js	3	2	212	214	1	2	11	15	72	1	1	0.0012631578947368421
changelog.spec.js	2	2	110	112	1	1	10	9	52	1	1	0.0008421052631578948

Figura 4.2. Exemplo de tabela com os valores das métricas por arquivo.

Para a análise da *EntropyMetricsView*, foi utilizado a tabela da ferramenta para verificar o valor de entropia dos arquivo em cada intervalo de cada período definido para a avaliação.

4.2. Como as métricas se relacionam com a entropia?

A Tabela 4.4 apresenta como as correlações entre a entropia e as métricas foram classificadas ao longo do período de dois anos, entre 01/07/2015 e 01/07/2017, dividido em 8 intervalos de 3 meses.

Considerando os intervalos de 3 meses, a maioria das correlações da entropia com as métricas de software foram de nível fraco. A correlação com número de defeitos e número de linhas adicionadas foi moderadas em 3 intervalos. A correlação com número de defeitos e linhas removidas foi forte em 1 intervalo. A correlação entre entropia e número de *commits* do *owner* foi fraca em 7 dos 8 intervalos nesse período.

Tabela 4.4. *Heat Map* intervalos de 3 meses

Entropia x Métricas	**Níveis de correlação**				
	Muito forte	**Forte**	**Moderada**	**Fraca**	**Desprezível**
Defeitos	0	1	3	3	1
Linhas modificadas	0	0	2	5	1
Linhas adicionadas	0	0	3	3	2
Linhas removidas	0	1	2	3	2
Pull Request Merged	0	0	1	6	1
Comentários em Pull Request Merged	0	0	1	5	2
Pull Request Unmerged	0	0	2	4	2
Comentários em Pull Request Unmerged	0	0	1	4	3
Commits Owner	0	0	1	7	0
Commits Experience	0	0	0	4	4

A Tabela 4.5, apresenta como as correlações entre a entropia e as métricas foram classificadas ao longo do período de quatro anos, entre 01/07/2013 e 01/07/2017, dividido em 8 intervalos de 6 meses.

Considerando os intervalos de 6 meses, a maioria dos níveis de correlação foram moderadas ou fracas. A correlação com a quantidade de linhas removidas foi moderada em 5 intervalos. A correlação da entropia com a quantidade de *pull request unmerged* foi moderada em 6 intervalos. A correlação com linhas adicionadas e comentários em *pull request merged* foi fraca em 5 intervalos. O nível de correlação foi forte apenas com o número de comentários em *pull request unmerged*, em um intervalo.

Tabela 4.5. *Heat Map* para intervalos de 6 meses

Entropia x métricas	**Níveis de correlação**				
	Muito forte	**Forte**	**Moderada**	**Fraca**	**Desprezível**
Defeitos	0	0	3	4	1
Linhas modificadas	0	0	4	4	0
Linhas adicionadas	0	0	3	5	0
Linhas removidas	0	0	5	3	0
Pull Request Merged	0	0	2	4	2
Comentários em Pull Request Merged	0	0	2	5	1
Pull Request Unmerged	0	0	6	2	0
Comentários em Pull Request Unmerged	0	1	4	3	0
Commits Owner	0	0	1	4	3
Commits Experience	0	0	0	3	5

A Tabela 4.6, apresenta como as correlações entre a entropia e as métricas foram classificadas ao longo do período de 7 anos, entre 01/07/2010 e 01/07/2017, dividido em 7 intervalos de 1 ano.

Considerando os intervalos de 1 ano, a maioria dos níveis de correlação foi moderada ou forte. A entropia apresentou forte correlação com número de defeitos e número de comentários em *pull request unmerged* em 4 dos 7 intervalos. O nível de correlação foi moderada com quantidade de linhas removidas, quantidade de *pull request merged* e quantidade de *pull request unmerged* em 4 intervalos. A correlação foi moderada com linhas modificadas e adiciondas em 5 intervalos.

Tabela 4.6. *Heat Map* para intervalos de 1 ano

Entropia x Métricas	**Níveis de correlação**				
	Muito forte	**Forte**	**Moderada**	**Fraca**	**Desprezível**
Defeitos	0	4	2	1	0
Linhas modificadas	0	1	5	1	0
Linhas adicionadas	0	2	5	0	0
Linhas removidas	0	3	4	0	0
Pull Request Merged	0	1	4	1	1
Comentários em Pull Request Merged	0	0	3	3	1
Pull Request Unmerged	0	2	4	1	0
Comentários em Pull Request Unmerged	0	4	2	1	0
Commits Owner	0	1	3	0	3
Commits Experience	0	1	3	0	3

Com base nas tabelas, as métricas apresentam os níveis de correlação mais baixo com a entropia são as quantidades de *commits* do *owner* e do contribuidor com maior *experience*. O número de defeitos, linhas modificadas, linhas adicionadas, linhas removidas e métricas relacionas aos *pull requests unmerged* apresentam nível de correlação moderada ou forte na maioria do intervalos. Assim, as métricas com nível de correlação moderada ou forte com a entropia, podem ser consideradas como indicadores ao analisar quais os arquivos merecem maior atenção no projeto.

Analisando as Tabelas 4.4, 4.5 e 4.6 é possível concluir que o nível de correlação aumenta quando se considera períodos maiores. Portanto, o desenvolvedor pode analisar as métricas de um projeto para intervalos de 6 meses ou um ano, e assim, ter mais métricas para utilizar como indicadores para localizar os arquivos mais críticos. Como o cálculo da correlação é feita considerando todos os arquivos, é provável que a correlação aumente, caso fosse considerado apenas os arquivos com valores de entropia maior do que a entropia média do projeto.

Capítulo 5

Conclusão

Este trabalho apresentou uma ferramenta para coletar dados de projetos do Github, calcular a entropia de mudança dos arquivos dos projetos, calcular métricas sociais, de autoria e de processo, e por fim gerar visualizações sobre a entropia e as métricas.

A *EntropyMetricsView* permite os desenvolvedores monitorarem os arquivos e pacotes do projeto com maior valor de entropia de mudança utilizando *Treemap*, assim, é possível visualizar facilmente quais arquivos e pacotes tem maior taxa de mudança, e portanto, maior chance de apresentar defeitos. As séries temporais geradas pela ferramenta permite visualizar o comportamento da métricas ao longo do tempo, tornando possível observar períodos em que há maior número de atividades no projeto. A visualização do *Heat Map* permite analisar os níveis de correlação entre as métricas e a entropia, fornecendo mais indicadores para auxiliar o desenvolvedor a encontrar os arquivos mais relevantes do projeto.

Por fim, foi feito uma análise do projeto Angular utilizando a *EntropyMetricsView*, em que foi gerado visualizações sobre a entropia e as métricas de *software* em diferentes períodos. Na análise, foi demonstrado que as visualizações permitem localizar os arquivos mais relevantes do Angular e que o nível de correlação entre a entropia e as métricas aumenta quando são considerados períodos maiores, assim a ferramenta é mais útil para projetos que tenham pelo menos 6 meses de histórico de mudanças.

5.1. Limitações e trabalhos futuros

Esta seção apresenta as limitações da avaliação da *EntropyMetricsView* e trabalhos futuros. A ferramenta não foi avaliada com outros desenvolvedores, devido ao tempo exigido para configurar um ambiente e analisar o uso da ferramenta com outras pessoas.

Não foram implementados visualizações de rede, pois a biblioteca utilizada não oferece suporte para visualizações de rede e assim seria necessário mudar a estrutura dos dados para utilizar outras bibliotecas. Devido ao tempo para implementar as métricas de software, não foram implementados métricas para medir a idade do arquivo, quantidade de refatorações e *ownership* por arquivo, que podem ser métricas que influenciam no valor da entropia. Como trabalhos futuros pretende-se utilizar bibliotecas que ofereçam suporte para visualizações de rede e implementar métricas para medir o tempo de existências dos arquivos, quantidade de refatorações e calcular *ownership* por arquivo.

Referências

BETTENBURG, Nicolas; HASSAN, Ahmed E. Studying the impact of social interactions on software quality. *Empirical Software Engineering*, v. 18, n. 2, p. 375–431, Apr 2013. ISSN 1573-7616. Disponível em: https://doi.org/10.1007/s10664-012-9205-0.

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.

FRY, Ben. *Visualizing Data*. 1st. ed. [S.l.: s.n.], 2008. ISBN 978-0-596-51455-6.

GREILER, Michaela; HERZIG, Kim. Code Ownership and Software Quality : A Replication Study. 2015.

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.

KOSCIANSKI, A.; SOARES, M. dos Santos. *Qualidade de Software - 2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. Novatec, 2007. ISBN 9788575221129. Disponível em: https://books.google.com.br/books?id=O9aWoUq6L88C.

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.