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

CLÁUDIA LÁZARA POIET SAMPEDRO

ESTUDO SOBRE TÉCNICAS DE VISUALIZAÇÃO QUANTO AO USO DE RÓTULOS EM REPOSITÓRIOS DE SOFTWARE

MONOGRAFIA

CAMPO MOURÃO

2019

CLÁUDIA LÁZARA POIET SAMPEDRO

ESTUDO SOBRE TÉCNICAS DE VISUALIZAÇÃO QUANTO AO USO DE RÓTULOS EM REPOSITÓRIOS 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: Profª. Drª. Aretha Barbosa Alencar

Coorientador: Prof. Dr. Marco Aurélio Graciotto Silva

CAMPO MOURÃO

2019

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO

Às **13:50** do dia **29 de novembro de 2019** foi realizada na sala **D103** da UTFPR-CM a sessão pública da defesa do Trabalho de Conclusão do Curso de Bacharelado em Ciência da Computação do(a) acadêmico(a) **Claudia Lazara Poiet Sampedro** com o título **Estudo sobre técnicas de visualização quanto ao uso de rótulos em repositórios de software**. Estavam presentes, além do(a) acadêmico(a), os membros da banca examinadora composta por: **Profa. Dra. Aretha Barbosa Alencar** (orientador(a)), **Prof. Dr. Andre Luiz Satoshi Kawamoto, Prof. Dr. Lucio Geronimo Valentin e Prof. Dr. Marco Aurélio Graciotto Silva**. 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 ________ (__________________). Esse resultado foi comunicado ao (à) acadêmico(a) e aos presentes na sessão pública. A banca examinadora também comunicou ao (à) 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, **29 de novembro de 2019**

Prof. Dr. Andre Luiz Satoshi Kawamoto Membro 1	**Prof. Dr. Lucio Geronimo Valentin** Membro 2
Prof. Dr. Marco Aurélio Graciotto Silva Membro 3	**Profa. Dra. Aretha Barbosa Alencar** Orientador

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

Resumo

Poiet Sampedro, Cláudia Lázara. Estudo sobre técnicas de visualização quanto ao uso de rótulos em repositórios de software. 2019. 53. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2019.

Contexto: Técnicas de visualização são úteis para análise de grandes quantidades de dados, pois estas ampliam a capacidade cognitiva humana no processo de exploração de dados, através da utilização de modelos gráficos e representações visuais. Outra área bastante explorada é a mineração de repositórios, a qual pode transformar dados coletados de repositórios de software em informações úteis para tomada de decisões dentro do gerenciamento de projetos de software. Correlacionando estas duas áreas é possível buscar por padrões não-identificados em projetos de software.
Objetivo: O objetivo deste é aplicar técnicas de visualização para analisar o uso de rótulos (*labels*) em tarefas presentes em projetos hospedados em plataformas sociais de desenvolvimento.
Método: O método empregado foi organizado em cinco etapas: conhecimento do domínio; coleta e pré-processamento de dados; extração e visualização de padrões, responsável pela geração da visualização com os dados pré-processados e análise visual desta; pós-processamento, que pode reiniciar o ciclo já empregado em busca de novos padrões pelo uso de outras técnicas e/ou ajuste de parâmetros; e, por fim, utilização do conhecimento.
Resultados: Analisado o domínio de projetos de software livre, plataformas sociais de desenvolvimento de software e mecanismos de colaboração centrados em rótulos, foram escolhidos rótulos e sua utilização em tarefas no contexto da plataforma GitHub para o repositório NextCloud. Quanto à coleta e pré-processamento, foi utilizada a API REST da plataforma GitHub e scripts, desenvolvidos em Python e JavaScript. Buscando caracterizar e analisar o uso de rótulos, foram utilizadas visualizações baseadas nas técnicas *box plot*, *streamgraph*, desenho de grafo e diagrama de *Sankey*. Utilizando o conhecimento obtido nas etapas anteriores, conclui-se que o projeto analisado utiliza do recurso de rótulos e este tende a aumentar o número de comentários nas tarefas, melhorando a comunicação entre desenvolvedores. Quanto ao tempo de fechamento das tarefas, estes se mostraram menores para tarefas sem rótulos, o que pode indicar que estas sejam bastante simples e por isso foram concluídas mais rapidamente. Observando o grafo de coocorrência de rótulos, fica evidente que, além de vários rótulos por tarefa, existe uma grande quantidade de associações de rótulos utilizados por tarefa. Também foi possível perceber que a

comunidade do projeto tende a usar mais de um rótulo por tarefa. Considerando o diagrama de Sankey, foi possível observar a relação entre os rótulos, quantidade de comentários, tempo de conclusão das tarefas e tipo de conteúdo tratado nos comentários, observando, por exemplo, que tarefas com poucos comentários são concluídas mais rapidamente.
Conclusões: O uso de técnicas de visualização facilitam a identificação de padrões e indícios referentes às questões estabelecidas nesta trabalho, quanto ao uso de rótulos em repositórios de software hospedados na plataforma GitHub, em especial, qual a contribuição dos rótulos na comunicação dos desenvolvedores, e qual o efeito global do uso destes no tempo de conclusão das tarefas, na comunicação, e na forma com que esta é concluída.

Palavras-chaves: Visualização de informação. Mineração de repositórios de software. Plataforma social de desenvolvimento. Rótulos.

Abstract

Poiet Sampedro, Cláudia Lázara. . 2019. 53. f. Monograph (Undergradute Program in Computer Science), Federal University of Technology – Paraná. Campo Mourão, PR, Brazil, 2019.

Context: Visualizations techniques are used to analyzes large amounts of data, because they enhance human cognitive ability in the process of data exploration through the use of graphical models and visual representations. Repository mining is another widely explored area, which can transform data collected from software repositories into useful information for decisiont. By correlating these two areas it is possible to look for unidentified patterns in software projects.
Objective: This study pourpose is use visualization techniques to analyze the use of labels in issues present in projects hosted on social development platforms.
Method: The method employed was organized in five steps: domain knowledge; data collection and preprocessing; extraction and visualization of patterns, responsible for generating the visualization with the preprocessed data, and visual analysis of it; postprocessing, which may restart the cycle already employed, searching for new patterns by using others techniques and / or parameter setting; and, finally, use of knowledge.
Results: Analyzing the domain of open source projects, social software development platforms and label-centric collaboration mechanisms, labels were chosen and their use in issues in the context of the GitHub platform for the NextCloud repository. As for collection and preprocessing, we used the GitHub platform REST API and scripts, developed in Python and JavaScript. In order to characterize and analyze the use of labels, we used visualizations based on box plot, streamgraph, graph drawing and Sankey diagram techniques. Using the knowledge obtained in the previous steps, it is concluded that the analyzed project uses the labels feature and this tends to increase the number of comments on issues, improving communication between developers. As for the issues lifetime, these were shorter for issues without labels, which may indicate that they are quite simple and therefore completed quickly. Looking at the label co-occurrence graph, it is evident that in addition to several labels per issue, there are a large number of label associations used per issue. It was also noted that the project community tends to use more than one label per issue. Considering the Sankey diagram, it was possible to observe the relationship between labels, number of comments, issue lifetime and the content handled in comments, noting, for example, that issues with few comments are finished faster.

Conclusions: Visualization techniques facilitates the identification of patterns and evidences, regarding the questions established in this study about the use of labels on github repositories, in particular, the contribution of labels on communication process with the developers, and what is the global effect on the issue's closing time, in the communication and in the issue conclusion.

Keywords: Data Visualization. Mining software reposirories. Social develpment platform. Labels

Lista de figuras

Lista de acrônimos

Sumário

Capítulo 1

Introdução

Com a evolução da computação, tornou-se possível obter e manipular grandes quantidades de dados, ocorrência a qual pode afetar a análise e entendimento das informações por parte dos usuários. A análise de dados pode ser complexa, podendo gerar perda de informações úteis se não feita com cuidado. Para contornar esta questão, surgiram as técnicas de visualização de dados, buscando ampliar a capacidade cognitiva humana em processo de exploração de dados através da criação de modelos gráficos e representações visuais (ALENCAR, 2013).

Paralelamente a isso, áreas relacionadas ao desenvolvimento de softwares estão cada vez mais presentes e, por consequência, é sempre desejado o aprimoramento de técnicas de engenharia de software, a fim de se obter softwares melhores e em menores períodos de tempo. Além disso, com a popularização de plataformas sociais para desenvolvimento colaborativo de softwares (como GitHub, GitLab e BitBucket), a área de mineração de repositórios se expandiu, permitindo que estes repositórios sejam fontes de dados e que possam ser analisados e manipulados a fim de agregar contribuições durante a tomada de decisões de um projeto de software (HASSAN, 2008).

Dentro das plataformas sociais para desenvolvimento colaborativo de software, comumente há alguma ferramenta para a rotulagem de tarefas, utilizada para marcar elementos referentes a uma tarefa, como erros, a fim de facilitar a identificação destes (TREUDE, 2012). Além disso, os rótulos podem ser utilizados como uma maneira informal do time de desenvolvimento tomar notas sobre tarefas durante o processo de desenvolvimento. Existem várias questões que permeiam o uso de rótulos em repositórios de software, sendo estas diversas e abrangentes em várias áreas, como organização de repositórios, comunicação entre desenvolvedores, redução do tempo de vida de uma tarefa, entre outras. As principais questões que buscamos responder com este estudo são:

- O projeto faz uso de rótulos?
- O tempo de vida de uma tarefa é influenciado pelo rótulo atribuído?

- O uso de rótulos melhora a comunicação entre desenvolvedores?
- Qual é o efeito global do uso de rótulos no tempo de conclusão das tarefas, na comunicação e na forma da conclusão das tarefas?

A primeira questão trata da premissa que o projeto a ser analisado precisa definir e utilizar rótulos em suas tarefas. A não satisfação desta questão, obviamente, torna desnecessário o estudo dos efeitos de rótulos no projeto. Vencida essa barreira, ainda é necessário compreender como os rótulos são utilizados. Por exemplo, as associações de rótulos a uma tarefa e a utilização de rótulos ao longo do tempo são indicadores da forma com que os rótulos são utilizados.

Entretanto, a simples utilização de rótulos não implica em efeitos nas atividades de engenharia de software. As questões seguintes tratam desse aspecto. Uma das razões da utilização de rótulos é externar as intenções de gerenciamento e fomentar as ações de desenvolvedores. Isso pode afetar o tempo necessário para concluir uma tarefa e a quantidade de atividades, representadas pelas mensagens trocadas na tarefa. Além disso, nem sempre a relação entre os rótulos e os possíveis efeitos em seu ciclo de vida e comunicação são tão diretamente perceptíveis. A última questão atenta-se justamente a isso.

Assim, técnicas de visualização de dados foram aplicadas para auxiliar nas buscas para as respostas dessas questões, pois estas auxiliam no processo de entendimento e reconhecimento de padrões previamente desconhecidos de grandes quantidades de dados. Neste estudo foi possível gerar visualizações de informação sobre dados coletados dos rótulos de um repositório de software hospedado na plataforma social de desenvolvimento colaborativo GitHub.

Portanto, desejamos detectar visualmente padrões para descobrir se o projeto de software analisado faz uso de rótulos, sejam estes individuais ou em conjunto, bem como qual a influência que o uso de rótulos pode ter no tempo de vida de uma tarefa. Outro ponto a ser visualizado, é a contribuição de os rótulos podem ter na comunicação entre os desenvolvedores, analisando o número de comentários que tarefas rotuladas e não rotuladas possuem associados. Por fim, busca-se gerar um gráfico que nos ajude a descobrir qual é o efeito global do uso de rótulos no tempo de conclusão das tarefas, e na comunicação e na forma de conclusão das tarefas.

Serão encontrados no decorrer do texto outros três capítulos. No Capítulo 2 está contida toda a revisão bibliográfica e embasamentos teóricos necessários para a elaboração do estudo. No Capítulo 3 apresenta-se o método empregado durante o desenvolvimento do estudo, bem como as visualizações obtidas como resultados. No Capítulo 4 são apresentadas as contribuições obtidas com esse estudo e trabalhos futuros possíveis.

Capítulo 2

Referencial Teórico

Este capítulo possui a finalidade de fundamentar princípios e técnicas que foram utilizadas para a elaboração deste trabalho. Além disso, através de artigos relacionados ao tema proposto, foram feitas análises mais completas sobre a temática utilizada e, assim, notados alguns pontos de atenção quanto ao uso de Visualização de Dados para visualizar o uso de rótulos em projetos de software livre.

Neste capítulo, na Seção 2.1 é abordado o uso de plataformas sociais para desenvolvimento de projetos, como o GitHub. A Seção 2.2 relata o contexto do uso de rótulos em projetos de software. O contexto da área de mineração de repositórios de software é discutido na Seção 2.3 que consiste na aquisição de dados sobre repositórios de software com o objetivo de analisar padrões e comportamentos presentes no repositório e utilizá-los para aprimorar o projeto. A Seção 2.4 apresenta a área de pesquisa de Visualização de Informação e algumas técnicas dessa área utilizadas neste trabalho. Por fim, a Seção 2.5 apresenta trabalhos relacionados que buscam usar essa área de pesquisa para visualizar repositórios de software.

2.1. Plataformas Sociais para Engenharia de Software

Plataformas sociais para desenvolvimento colaborativo são utilizadas para hospedagem de repositórios e para o desenvolvimento colaborativo de software entre membros de uma equipe, colaboradores de uma ferramenta. Estas costumam utilizar a ferramenta *git*[1] para realizar o controle de versão e o gerenciamento de alterações. Dentre as plataformas mais famosas estão o GitHub, GitLab e BitBucket, os quais possuem a finalidade de registrar todas as mudanças realizadas em arquivos de código presentes em um **repositório**, registrar defeitos e tarefas, além do emprego de mecanismos para controle de alteração de código integrado com revisão de código.

[1] <https://git-scm.com/>

Os repositórios de software consistem basicamente em um conjunto de diretórios e arquivos, onde os arquivos estão dispostos, dispondo de funções extras, como criar ***branches***, as quais se tornam uma versão paralela ao repositório e suas mudanças não afetam a versão principal. As mudanças implementadas em um *branch* podem ser aplicadas a versão principal, utilizando-se o comando ***merge***. Mudanças podem ser submetidas para um *branch* através de ***commits***. Esses *commits* costumam possuir mensagens de texto para explicitar as alterações realizadas.

Além das operações supracitadas, são possíveis também as operações de ***clone*** e ***fork***, as quais permitem um usuário copiar o repositório para seu computador e para seu perfil do GitHub, respectivamente. Os *commits* são isolados para a cópia realizada, seja através de *clone* ou *fork*, e a única forma destes atingirem o projeto de "origem"ou qualquer outro com história em comum, é por operações de *push* ou *pull request* entre os repositórios.

Dentro dos repositórios é possível a criação de tarefas (***issues***), as quais simbolizam relatos de erros ou tarefas a serem realizadas dentro do projeto, como novas funcionalidades, etc. Nessas podem ser adicionados rótulos (***labels***), com a finalidade de rotular o conteúdo de uma tarefa (*issue*), facilitando a compreensão e gerenciamento da tarefa a ser realizada. Um colaborador do projeto geralmente soluciona uma dada tarefa pela submissão de suas mudanças através de um ***pull request***, a qual cabe aos administradores revisarem e decidirem pela aceitação ou rejeição das mudanças propostas.

Neste trabalho, a plataforma social para engenharia de software selecionada para a obtenção dos dados foi o GitHub, por questões de popularidade (KALLIAMVAKOU et al., 2014) e disponibilidade de API para a obtenção dos dados. No entanto, destaca-se que, com as devidas adaptações para coleta de dados, as técnicas de visualização selecionadas nesse estudo poderiam ser aplicadas a outras plataformas sociais.

O GitHub é uma plataforma que utiliza a ferramenta de controle de versão *git*, focada na hospedagem de repositórios de códigos fonte. Este fornece uma interface *web* para repositórios, acompanhamento de tarefas, dentre outras funcionalidades. Sua popularidade aumentou nos últimos anos graças à possibilidade da utilização deste por pessoas ou grandes empresas para hospedagem de repositórios de software (KIKAS et al., 2015).

No contexto do GitHub, há uma maneira de sinalizar tarefas em aberto para implementação. Estas são geralmente representadas por um texto livre, o qual descreve a tarefa pendente. Estas estão relacionadas a um repositório e podem ser criadas por qualquer pessoa. Além disso, há a possibilidade de outros usuários adicionarem comentários em forma de texto às mesmas.

As tarefas possuem um sistema de categorização, as quais as rotulam como abertas ou fechadas. Tarefas abertas significam que ainda não foi apresentada uma solução para a tarefa proposta, enquanto as fechadas simbolizam tarefas que já foram concluídas ou canceladas. Importante ressaltar que as tarefas podem transitar por entre estas categorias livremente (KIKAS et al., 2015).

Além do texto presente na descrição de uma tarefa, o GitHub dispõe de um sistema de rotulação, a qual permite a adição de rótulos, para classificar a tarefa de alguma forma, categorizando-a como, por exemplo, um erro ou uma nova funcionalidade relacionada a uma versão do software.

2.2. Uso de marcações ou rótulos em projetos de software

Como mencionado anteriormente, várias ferramentas utilizadas para controle de versão e gerenciamento de softwares, como GitHub, GitLab, Jazz, entre outros, utilizam algum sistema para rotular as atividades a serem feitas, empregando características relevantes a estas. Os rótulos são palavras ou termos escolhidos livremente, a fim de atribuí-los a parte de uma informação. No contexto de desenvolvimento de software, rótulos são utilizados para marcar elementos referentes a uma tarefa, como necessidade de testes ou erros, a fim de ajudar no processo de identificação destes (TREUDE, 2012).

Rótulos estão sendo utilizados há décadas para rotular artefatos de software existentes durante o período de desenvolvimento de um software, e ainda ajudar na documentação, gerenciamento, correção de erros, entre outros. Além disso, estes podem ser usadas pelo time de desenvolvimento como uma maneira informal de tomar notas sobre tarefas durante o desenvolvimento de software.

Dentre as maiores vantagens do uso de rótulos no desenvolvimento de software está contido o fato destes serem flexíveis, simples e surgirem naturalmente, minimizando a necessidade de mudar a classificação inicial de um artefato de software. Em complemento, rótulos podem ser utilizado para organizar, gerir e categorizar artefatos de software (TREUDE, 2012).

Em plataformas de desenvolvimento com aspectos sociais, e comumente relacionadas ao desenvolvimento de software livre, como GitHub e GitLab, o uso de rótulos é possível e promovido como maneira de categorizar mudanças necessárias e informalmente relacioná-las a um componente ou versão (NAYEBI et al., 2017). Estes ainda, podem ser utilizados como atrativo para colaboradores, por supostamente explicitar a qual categoria pertence uma tarefa aberta, as tornando mais convidativas para desenvolvedores que se identificam com as características destas.

2.3. Mineração de repositórios de software (MSR)

Repositórios de software são comumente utilizados para armazenar informações e raramente utilizados na tomada de decisões do projeto. A área de pesquisa de Mineração de Repositórios de Software (MSR) vem para suprir esse problema, a fim de descobrir novos padrões e informações capazes de ajudar na tomada de decisões sobre o projeto. De maneira geral, os aspectos principais da mineração de software são: criação de técnicas automatizadas para melhorar a extração de informações e a descoberta de novas técnicas para minerar padrões previamente desconhecidos de um repositório (HASSAN, 2008).

O ciclo da MSR é composto por 4 etapas: coleta de dados, modelagem dos dados, síntese dos dados e análise de resultados. Na primeira etapa os dados são coletados, selecionados e organizados de forma mais interessante na segunda etapa. A terceira etapa é responsável por sumarizar os dados, os quais são analisados na última etapa (HEMMATI et al., 2013).

Os dados minerados podem ser bastante heterogêneos, variando desde nomes, descrições, comentários, rótulos, usuários, tarefas, entre tantos outros. Além disso, há diversas fontes para a obtenção dos dados, dentre essas bases de código estático, histórico de versões do software, rastros de execução de programas, relatórios de erros, listas de discussão e *logs* de implantação de sistemas.

2.4. Visualização de Informação

Com a evolução da computação tornou-se possível obter e manipular montantes de dados volumosos, fenômeno que pode prejudicar a compreensão dessas informações. Já que se não analisarmos esses dados com a mesma velocidade que os obtemos, podemos estar perdendo a informação possivelmente útil contida neles e coletamos os dados desnecessariamente.

A Visualização de Dados (CARD et al., 1999; CHEN, 2006; SPENCE, 2007) é uma área de pesquisa que surgiu para minimizar esta questão, buscando através da criação de modelos gráficos e representações visuais, ampliar a capacidade cognitiva do ser humano em processo de exploração de dados (ALENCAR, 2013). Essa área de pesquisa teve sua utilização acelerada nas últimas décadas a partir da disponibilidade de interfaces gráficas mais eficientes com recursos de interação adequados. A Visualização de Dados tem como finalidade criar modelos gráficos e representações visuais, com o objetivo de explorar a sofisticada capacidade visual do ser humano para facilitar a exploração e aquisição de informações úteis contidas nos dados (OLIVEIRA; LEVKOWITZ, 2003). A Visualização tem o seu potencial maximizado quando acoplada a estratégias analíticas, oriundas da Mineração de Dados (FAYYAD et al., 1996), voltada ao processo de extração de conhecimento útil e previamente desconhecido em dados, por meio da aplicação de algoritmos que extraem modelos e padrões representativos. A integração de algoritmos de mineração e visualização permite associar a capacidade computacional com o conhecimento e capacidade de perceber padrões do ser humano. Já a disciplina de Mineração Visual (THOMAS; COOK, 2006) evoluiu a partir da Visualização e da Mineração de Dados, combinando essas duas áreas e reforçando a interação humana no processo analítico.

Existem inúmeras técnicas de Visualização de Dados, cada qual se mostra eficiente para diferentes tipos de dados (temporal, multidimensional, hierárquico, textual, geoespacial, etc.) e/ou diferentes padrões que se busca descobrir ou confirmar. Nas seções seguintes são apresentadas algumas técnicas de visualização de informação utilizadas nesse trabalho.

2.4.1. Box Plot

A técnica de visualização *box plot*, apesar de simples, é bastante poderosa quando utilizada para representar graficamente informações estatísticas dos dados (BENJAMINI, 1988).

Esta visualização representa a distribuição estatística dos dados e seus valores discrepantes (*outliers*), fornecendo assim um meio complementar para desenvolver uma perspectiva sobre o caráter dos dados. Um *box plot* é composto geralmente de limite inferior/valor mínimo, *lower fence*, primeiro quartil, segundo quartil ou mediana, terceiro quartil, *upper fence* e limite superior/valor máximo.

Um quartil é qualquer uma das três medidas de posição, que divide um conjunto ordenado de dados em quatro partes iguais, e assim cada parte representa 1/4 da amostra ou população. Por exemplo, o primeiro quartil ($q1$) é o valor limite/máximo do primeiro quarto, o qual engloba 25% dos valores menores ou iguais ao valor do primeiro quartil. A Figura 2.1 representa um *box plot* gerado com valores arbitrário de 0 a 50, e com seus pontos principais destacados por etiquetas.

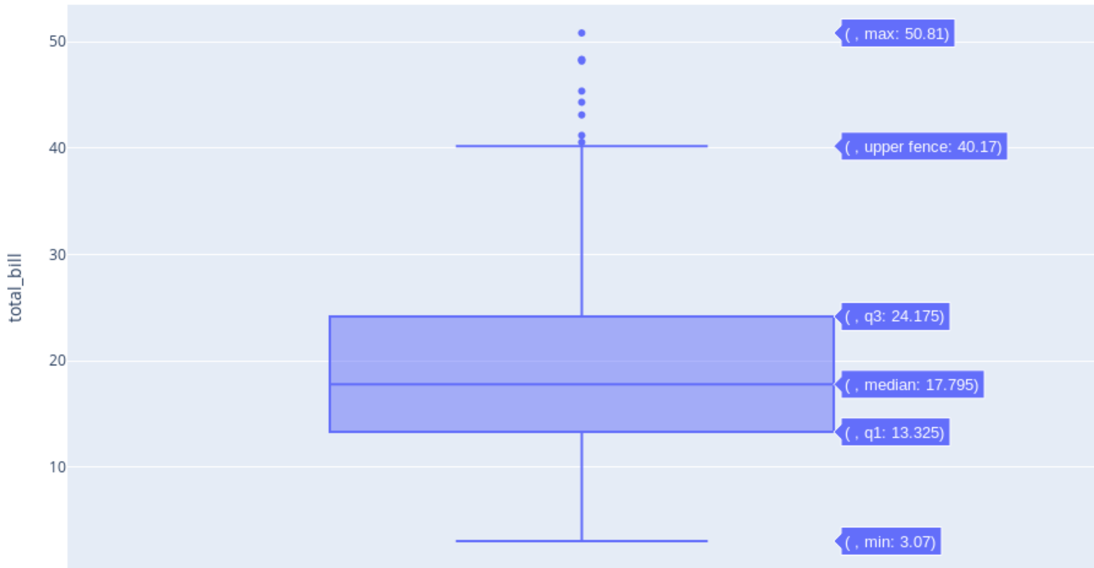


Figura 2.1. Exemplo da técnica box plot para um montante de dados arbitrários dentro intervalo numérico de 0 a 50.
Fonte: Página da biblioteca Plotly[2].

A etiqueta de nome *min* representa o limite inferior, ou seja, o menor valor apresentado na distribuição de dados. O primeiro quartil ($q1$) representa o início da amplitude interquartílica, que é a faixa de dados que demonstra 50% do total de valores analisado. O segundo quartil, *median*, têm a função de apresentar a mediana dos dados – valor que divide um conjunto de valores ordenados em partes iguais. O terceiro quartil ($q3$) demonstra o limite superior da amplitude interquartílica. O último quartil, *upper fence*, delimita o maior valor encontrado dentro da maior parte dos dados e os valores discrepantes, ou seja, os valores acima do último quartil são considerados *outliers*. O *upper fence* é geralmente definido pela fórmula Q3 + (1.5 * IQR), onde Q3 representa o terceiro quartil, e o IQR, intervalo interquartil, representa os valores contidos entre o primeiro e o terceiro quartil.

2.4.2. StreamGraph

A técnica de visualização *streamgraph* surgiu a partir do protótipo chamado *ThemeRiver* (HAVRE et al., 2000), o qual visualiza variações temáticas no decorrer do tempo em grandes coleções de documentos. O *ThemeRiver* recebe este nome porque seu formato pode ser associado a um rio, que vai da esquerda para a direita com o passar do tempo, alterando temporariamente a largura das faixas associadas a algum dado.

A técnica *streamgraph* é uma variação de um gráfico de área empilhado, mas em vez de plotar valores em relação a um eixo fixo inferior, um *streamgraph* possui valores deslocados em torno de uma linha central. Esse tipo de gráfico exibe as alterações nos dados ao longo do tempo de diferentes categorias por meio do uso de formas fluidas e orgânicas que se assemelham ao fluxo de um rio. Isso torna os gráficos mais agradáveis esteticamente. O tamanho de cada fluxo individual é proporcional aos valores de cada categoria. O eixo ao qual um gráfico flui paralelo é usado para a escala de tempo. A cor pode ser usada para distinguir cada categoria / fluxo. Essa técnica de visualização é eficaz para descobrir tendências e padrões ao longo do tempo.

A Figura 2.2 apresenta a técnica *ThemeRiver* aplicada a notícias do período da crise cubana-americana ocorrida em 1960. Cada faixa representa palavras-chaves (termos mais frequentes) e suas respectivas frequências em notícias publicadas entre dezembro de 1959 até junho de 1961. A visualização foi anotada manualmente pelos autores com eventos externos que podem ter gerado mudanças na frequência de cada palavra-chave. Por exemplo, o termo “kennedy” apresenta uma maior frequência (largura da faixa) em abril de 1961, o que está diretamente ligado ao evento da invasão da Baía dos Porcos (“*Bay of Pigs*”)

2.4.3. Grafos

Um grafo é dado por um conjunto de vértices e um conjunto de arestas, onde cada aresta está associada a dois vértices (FEOFILOFF Y. KOHAYAKAWA, 2011). Este conceito pode ser visualmente representado pela Figura 2.3, nela os vértices são representados pela letra “Vi” e as arestas pela letra “ai”, sendo “i” o índice atribuído a estes. Em outras palavras, a função das arestas é estabelecer uma relação entre os vértices. É importante ressaltar que essas arestas podem ter peso e sentido, formando um grafo valorado e um dígrafo, respectivamente.

A aplicação de grafos na área de Visualização de Informação pode ser usada para a descoberta de padrões associados às relações estabelecidas entre os vértices e arestas. Existem várias técnicas de desenho de grafos (*layout*) que buscam posicionar os nós de um grafo em um espaço geralmente bidimensional para facilitar a visualização de padrões nos grafos. A maior parte dessas técnicas de desenho de grafos buscam seguir algumas regras estéticas como: minimizar

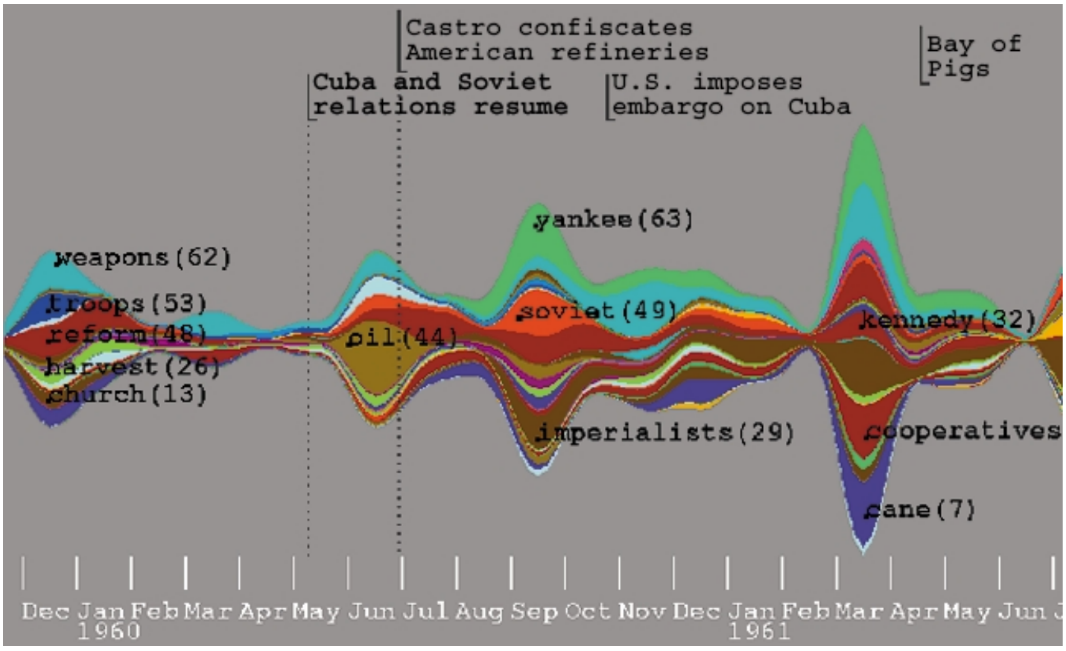


Figura 2.2. *ThemeRiver* usando dados de notícias jornalísticas. Esta mostra as palavras-chaves mais encontradas em notícias no período da crise cubana-americana ocorrida em 1960.
Fonte: Havre et al. (2000).

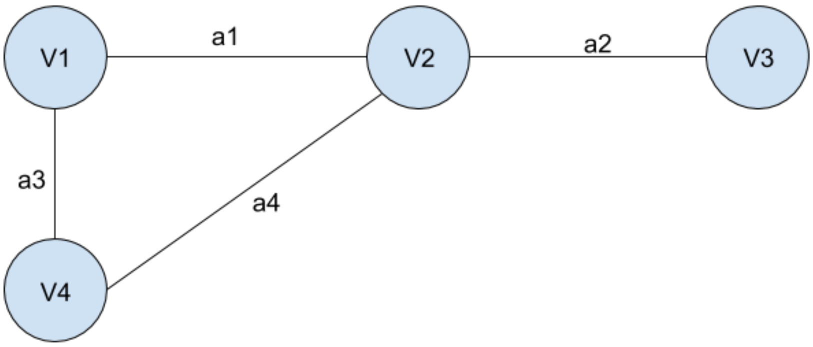
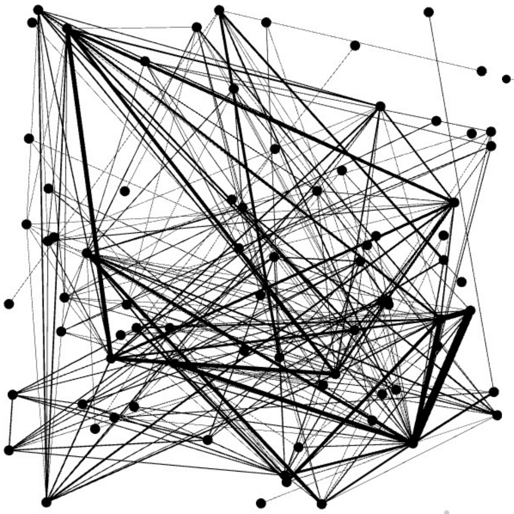


Figura 2.3. Exemplo de grafo não direcionado e sem peso nas arestas.
Fonte: Autoria própria (2018).

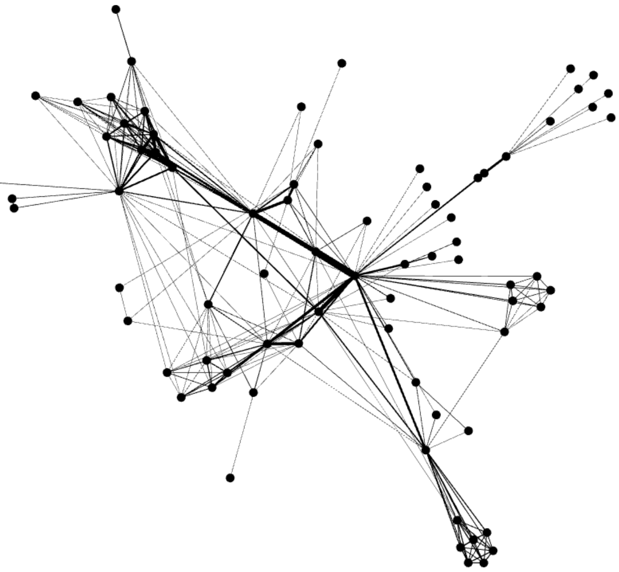
cruzamento de arestas, diminuir oclusão visual, minimizar tamanho total das arestas, manter razão de aspecto agradável, minimizar a área total de desenho, etc.

A algoritmo para aplicação de *layout* em grafos *ForceAtlas*, implementado pela ferramenta de manipulação de grafos Gephi[4], simula um sistema físico magnético para posicionar os nós de um grafo: nós aplicam repulsão entre si, e arestas atraem seus conectados. Este sistema movimenta iterativamente o grafo de forma a convergir para um estado de balanço entre os nós e as arestas, melhorando a compreensão dos dados dispostos (JACOMY SEBASTIEN HEYMANN, 2012).

[4] <https://gephi.org>



(a) Desenho de grafo sem a aplicação de técnica *layout*



(b) Desenho de grafo com a aplicação de técnica de *layout*

Figura 2.4. Exemplo do desenho de um mesmo grafo com e sem a aplicação de algoritmo de *layout*, utilizando a base de dados do filme *Les Miserables*.
Fonte: Página da ferramenta Gephi[3].

A Figura 2.4 mostra o mesmo grafo, com e sem aplicação de uma técnica de layout de grafos, respectivamente. Os dados do grafo apresentado são do filme *Les Miserables*, onde cada nó representa um personagem e as arestas as cenas em comum entre estes. A Figura 2.4a mostra o grafo sendo desenhado com seus nós sendo posicionados aleatoriamente. Já a Figura 2.4b mostra o mesmo grafo após a aplicação do algoritmo *ForceAtlas2*. Pode-se perceber que após aplicação do *layout* as informações estão melhor distribuídas e com menor grau de oclusão visual, além das arestas se apresentaram mais visíveis.

2.4.4. Diagrama de Sankey

Um diagrama de sankey é uma técnica de visualização usada para representar um fluxo de um conjunto de valores para outro. Os itens sendo conectados são chamadas nós e as conexões são chamadas *links*, onde a largura desses links é proporcional a taxa de fluxo existente entre os nós conectados. Diagramas de sankey ilustram informações quantitativas sobre fluxos, seus relacionamentos e transformações, sendo tradicionalmente utilizado para representar o fluxo de energia ou materiais de diversas redes ou processos (RIEHMANN et al., 2005).

Esse tipo de diagrama é mais usado quando você deseja mostrar um mapeamento muitos para muitos entre dois domínios. Por exemplo, Figura 2.5 mostra um diagrama sankey para visualizar um cenário projetado para a produção e o consumo de energia do Reino Unido em 2050. O suprimento de energia está à esquerda e as demandas à direita. Os nós intermediários agrupam formas de produção relacionadas e mostram como a energia é convertida e transmitida antes de ser consumida (ou até perdida). A espessura de cada conexão codifica a quantidade de fluxo da origem ao destino. Nota-se que fontes nucleares serão muito utilizadas na produção de energia termal neste cenário. Parte da energia termal é transferida para a rede elétrica de distribuição, porém parte dessa energia é perdida durante o processo de transferência.

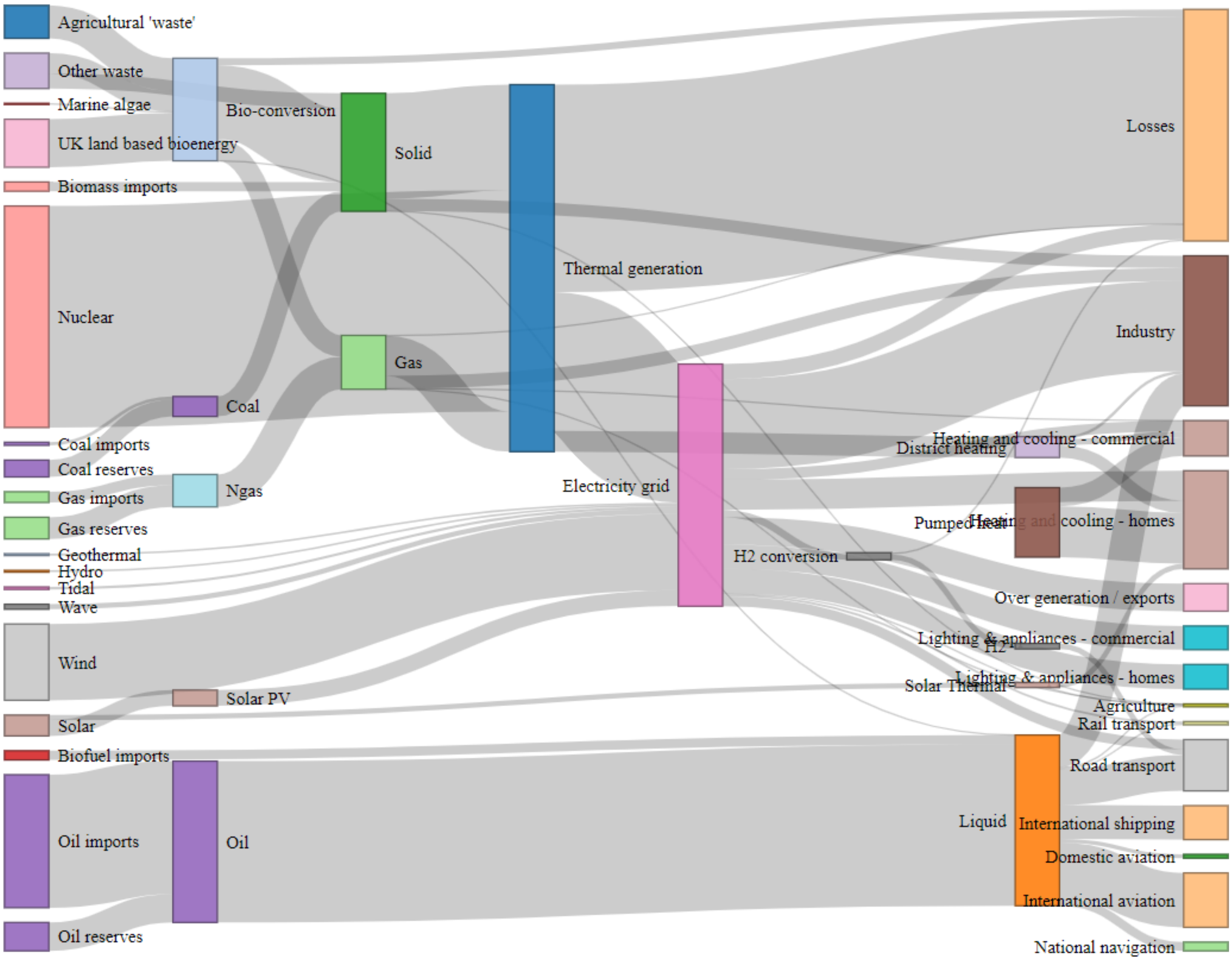


Figura 2.5. Exemplo de um diagrama de Sankey mostrando um cenário projetado para a produção e o consumo de energia do Reino Unido em 2050.
Fonte: Departamento de Energia e Mudanças Climáticas, Tom Counsell.

Uma grande vantagem do diagrama de Sankey é que ele gera menos oclusão visual quando comparado a outras técnicas similares com as coordenadas paralelas (ZHOU et al., 2008). No entanto, essa técnica apresenta limitações quando queremos relacionar o fluxo ao longo de diferentes eixos.

2.5. Visualização de Informação e aplicações em MSR

Com o objetivo de entender melhor o uso de técnicas de visualização de dados para repositórios de software do GitHub, em específico rótulos atribuídos a tarefas, foram selecionados alguns trabalhos relacionados ao tema proposto. Estes não são homogêneos, tendo em vista que mesmo abordando temas bastante próximos possuem abordagens e metodologias diferentes.

O artigo *GiLA: GitHub Label Analyser* (IZQUIERDO et al., 2015) propõe a criação de uma ferramenta *web* capaz de gerar três visualizações distintas, onde cada uma explora o uso dos rótulos utilizados nas tarefas do projeto de maneira diversa.

De maneira geral, a primeira visualização, ilustrada na Figura 2.6, tem o objetivo de identificar quais os rótulos mais usados e quais são mais comumente utilizados em conjunto. Essa visualização mostra um grafo de coocorrência de rótulos. Os vértices são os rótulos e o tamanho de cada vértice é proporcional ao número de vezes que cada rótulo foi utilizado individualmente em diferentes tarefas. As arestas conectam um par de rótulos e possuem pesos que indicam quantas vezes os rótulos conectados foram usados em conjunto em diferentes tarefas. Esse peso é representado visualmente pela largura da aresta.

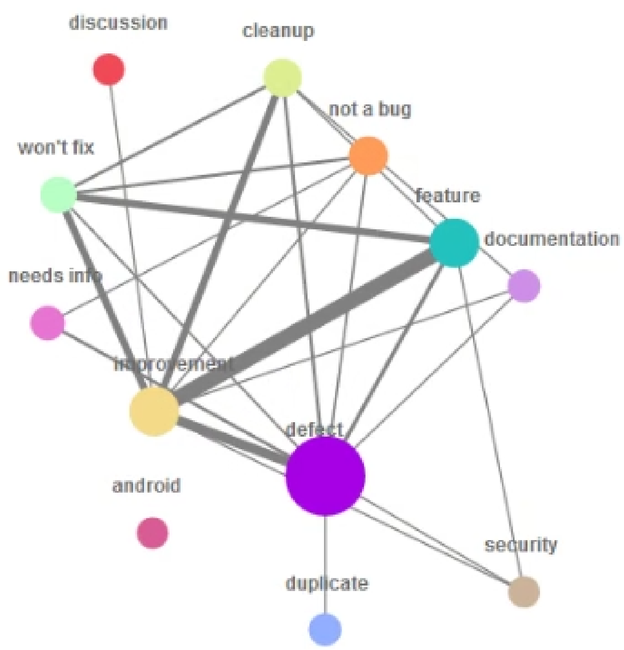


Figura 2.6. Primeira visualização proposta pelo artigo GiLA, mostrando um grafo de coocorrência entre rótulos.
Fonte: Izquierdo et al. (2015).

A segunda visualização tem o objetivo de relacionar os usuários, que colaboraram abrindo, fechando e comentando tarefas que utilizam de um dado rótulo. A Figura 2.7 exemplifica uma aplicação desta. O vértice circular central representa um rótulo, escolhido pelo usuário, e os vértices quadrados em volta representam usuários que contribuíram com tarefas que utilizavam deste rótulo. A espessura das arestas demonstra o número de comentários que o usuário fez em tarefas que utilizam deste rótulo. A largura e altura do vértice do usuário é proporcional ao número de tarefas criadas e fechadas, respectivamente, com este rótulo. Há uma diferença nas cores dos nós de usuários para diferenciar usuários comuns de administradores, que estão em roxo e laranja, respectivamente.

Um ponto de atenção sobre a segunda visualização é que esta se prende a apenas um rótulo, não sendo tão interessante para análises mais completas e que envolvam mais de um rótulo de um mesmo repositório. Colocar mais de um rótulo poderia gerar oclusão visual comprometendo a compreensão da visualização, mas isso poderia ser melhorado com técnicas mais avançadas de *layout* de grafos (e.g., *edge bundle*) e agrupamento visual de comunidades (detecção de comunidades).

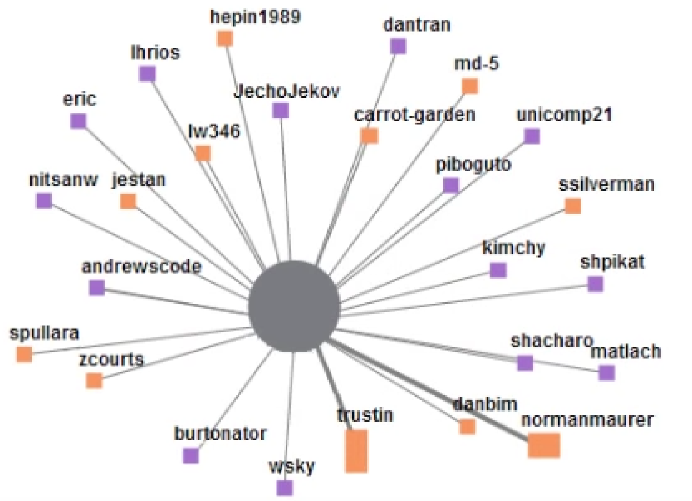


Figura 2.7. Segunda visualização proposta pelo artigo GiLA, relacionando a participação de usuários em tarefas com uma dado rótulo.
Fonte: Izquierdo et al. (2015).

A terceira e última visualização, demonstrada pela Figura 2.8, tem o objetivo de relatar o ciclo de vida das tarefas de determinado rótulo por meio de uma visualização baseada em árvore. Os autores determinaram dois marcos importantes no ramo central: tempo médio para primeiro comentário nas tarefas que contém tal rótulo e tempo médio para o primeiro comentário de um colaborador do projeto. O ramo central se ramifica em três ramificações: a porcentagem de tarefas fechadas, não fechadas e mescladas. Também é mostrada a média, em dias, para cada marco citado anteriormente. Esta terceira visualização tem a mesma característica de se focar em apenas um rótulo no repositório por vez.

O artigo possui conteúdo bastante interessante quando tratado o uso de rótulos dentro de *issues* do GitHub, em especial, a utilização de técnicas de visualização para analisar estas. A ferramenta para a obtenção dos dados que este utiliza é o GHTorrent [5], a qual pega os dados do GitHub e transforma em um banco de dados SQL. Esta tende ser de fácil utilização, mas apresenta inconsistência nos dados, podendo apresentá-los de forma obsoleta.

[5] http://ghtorrent.org/

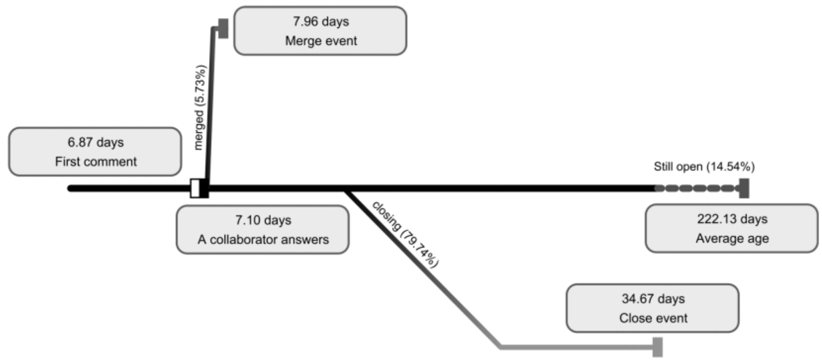


Figura 2.8. Terceira visualização proposta pelo artigo GiLA, demonstrando o ciclo de vida das tarefas associadas a um determinado rótulo.
Fonte: Izquierdo et al. (2015).

Já o estudo apresentado por Liao et al. (2018) mostra uso de técnicas de inteligência artificial para analisar repositórios de software, apresentando discussões interessantes do ponto de vista da engenharia de software e da visualização de dados. O foco principal deste artigo é responder três perguntas: (1) Quais comportamentos de usuários são importantes no GitHub? Quais comportamentos do usuário baseados em tarefas são importantes? Tarefas são importantes para projetos no GitHub?; (2) É efetivo aplicar rótulos a tarefas para o gerenciamento de tarefas?; e (3) Há um padrão temporal nas mensagens de *commit* em projetos populares? Se sim, que relaciona os padrões encontrados com as fases de um projeto?. Para responder tais questões os autores selecionaram alguns repositórios que consideraram relevantes, coletando seus dados através da API REST do GitHub.

A Figura 2.9 é responsável pela resposta da primeira parte da primeira questão proposta, "Quais comportamentos do usuário são importantes?", pois mostra todos os comportamentos de usuários e os distribui em um *heatmap*. Em um *heatmap*, os valores individuais contidos em uma matriz são representados graficamente como cores em uma escala de cor. No geral, quanto mais escura a cor, maior o valor associado. Nas linhas do *heatmap* são apresentados sete projetos considerados populares pelos autores e nas colunas são apresentados os comportados dos usuários. Cada evento de comportamento representa uma série de operações no objeto correspondente. Por exemplo, o *IssuesEvent* é acionado quando uma tarefa é atribuída, desatribuída, rotulada, tem seus rótulos removidos, aberta, editada, associada a *milestone*, desassociada de *milestones*, fechada ou reaberta. Todas as essas operações pontuariam no comportamento de usuário *IssuesEvent*. Cálculos semelhantes são feitos para todos os outros comportamentos de usuários. A Figura 2.9 ainda ilustra que a contribuição dos comportamentos de usuários geralmente tem alto grau de importância nestes sete projetos. As importâncias do *IssueComment* e *PullRequest* são relativamente altas em todos os repositórios selecionados.

Ainda sobre a primeira questão, a Figura 2.10 mostra uma visualização do tipo *treemap*. Este tipo de visualização é capaz de representar visualmente dados hierárquicos, na qual um retângulo é recursivamente dividido em pedaços, alternando cortes horizontais e verticais, com base nas populações das sub-árvores em um dado nível. Cada retângulo apresenta área de acordo

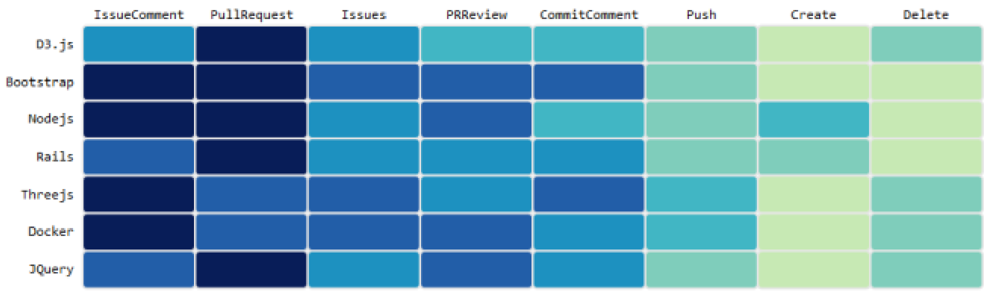


Figura 2.9. *HeatMap* da importância de eventos de comportamento de usuários em sete projetos populares.
Fonte: Liao et al. (2018).

com a quantidade de dados que este representa. Na Figura 2.10, os retângulos de dados representam comportamentos de usuários e estão agrupados de acordo com suas semelhanças, formando grupos relacionados a eventos de *pull requests*, tarefas, *commits*, *follow* e *watch*, e *fork* e *push*. Estes agrupamentos foram definidos manualmente pelos autores do artigo. É possível notar que o evento de *IssueCommentEvent* é o de maior importância do grupo de tarefas na *treemap* e o evento de *PullRequestEvent* é o de maior importância no grupo que é relacionado a *Pull Request*.

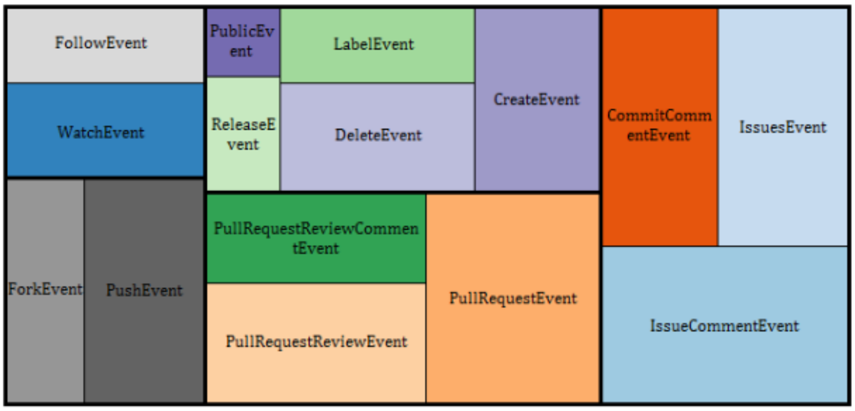


Figura 2.10. Agrupamentos de eventos baseados nas características e importância dos comportamentos. A área dos retângulos representa quão importante é tal comportamento.
Fonte: Liao et al. (2018).

Para segunda pergunta proposta, “É efetivo aplicar rótulos a tarefas para o gerenciamento de tarefas?”, usa-se uma visualização do tipo radar em sua resposta, representada pela Figura 2.11. A técnica de visualização utilizada permite comparar diferentes instâncias para diferentes variáveis numéricas. Em visualizações do tipo radar, cada eixo partindo do ponto central representa uma das variáveis numéricas e cada polilinha representa uma dada instância, intersectando esses eixos no valor correspondente.

No contexto do artigo, a visualização do tipo radar consiste em coletar tarefas com e sem rótulos para os projetos relacionado, e relacionar o tempo que estas levaram para serem fechadas em cada um deste projeto. A polilinha em azul representa tarefas com rótulos e a polilinha roxa tarefas sem rótulos. Quanto mais próxima da borda a polilinha está, maior o número de tarefas fechadas. O primeiro gráfico de radar considera o intervalo de tempo de 3 dias e o segundo gráfico

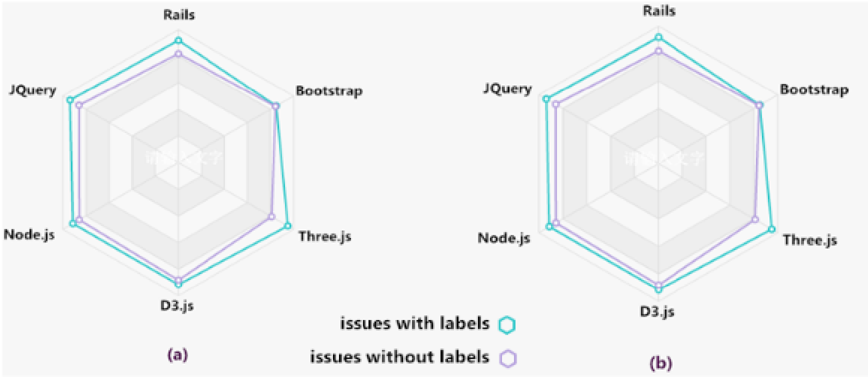


Figura 2.11. Representação em forma de radar para analisar a influência dos rótulos no tempo de fechamento de uma tarefa. Os gráficos (a) e (b) consideram o intervalo de tempo de 3 e 365 dias, respectivamente. Fonte: Liao et al. (2018).

um intervalo de tempo de 365 dias. Apesar dos intervalos de tempo diferentes, os resultados foram basicamente iguais, concluindo que tarefas com rótulos são fechadas mais rapidamente independente do intervalo de tempo utilizado na coletados dados.

A terceira e última questão levantada no artigo, "Há um padrão temporal nas mensagens de *commit* em projetos populares? Se sim, que relaciona os padrões encontrados com as fases de um projeto", busca alguma relação entre mensagens de *commits* e fases comuns no desenvolvimento de um *software*. Para tal, os *commits* dos repositórios *Three.js* e *JQuery* foram agrupados em três períodos de tempo. Em seguida foram selecionadas palavras chaves (mais frequentes) das mensagens de cada *commit* em cada período. Essa foram representadas por uma visualização do tipo *word cloud*, a qual dispõe a frequência de palavras, onde o tamanho de cada é diretamente proporcional a frequência das mesmas.

A Figura 2.12 mostra a aplicação da técnica *word cloud* para as palavras chaves de *commits*, organizadas por períodos de tempo. A primeira linha de *word clouds* corresponde ao repositório *Three.js*, enquanto a segunda linha corresponde ao repositório *JQuery*. Os autores definiram que o primeiro período de tempo é relacionado a coleta de funcionalidades, coincidindo com os rótulos mais presente nesta etapa, os termos *Feature* e *Requests*. Já no segundo período segundo os autores estão contidas atividades relacionadas a desenvolvimento e *design*, e os termos mais aparentes nos *commits* são: *docs*, relacionada a documentação; e *js*, relacionada a linguagem de programação utilizada no desenvolvimento do projeto *Three.js*. O último período teoricamente é relacionado a manutenção do software, e os rótulos mais presentes são: *css*, voltado para *design*; e *js*, relacionada a desenvolvimento do segundo projeto. Os autores deduziram então que as palavras que mais aparecem em cada etapa estão relacionadas com as etapas de desenvolvimento de um software. Entretanto, essa afirmação não condiz totalmente com os termos mais frequentes encontrados durante essas três fases.

As questões propostas são respondidas, mas muitas partem de pressuposições vagas e sem embasamento teórico. Por exemplo, na terceira inferência o autor agrupa tópicos que ele considera

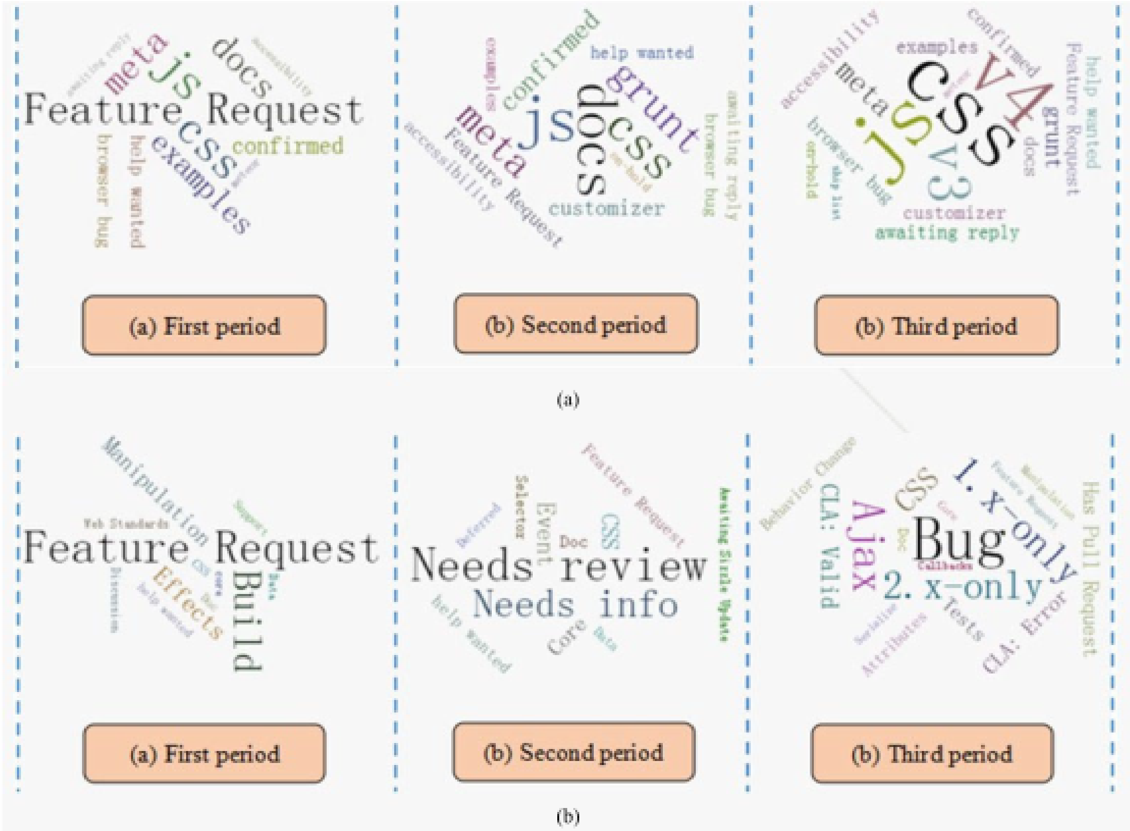


Figura 2.12. Representação da frequência de palavras chaves de *commits* nos repositórios *Three.js* e *JQuery*, agrupadas por períodos de tempo.
Fonte: Liao et al. (2018).

fazerem parte de determinada fase de um projeto, sem uma referência para este agrupamento, feito totalmente de forma manual de acordo com a concepção dos autores.

2.6. Considerações Finais

O embasamento teórico deste trabalho se deu basicamente por quatro temáticas distintas, mas correlacionadas de alguma forma, as quais são, plataformas sociais para engenharia de software, uso de marcações ou rótulos em projetos de software, mineração de repositórios de software (MSR) e visualização de informação e aplicações em MSR.

A primeira temática explorada, plataformas sociais para engenharia de software, consiste no entendimento sobre o funcionamento destas plataformas, bem como suas terminologias e ferramentas específicas que estas fornecem. Além disso, foi explicado mais especificamente sobre a plataforma GitHub, a qual foi utilizada como fonte para os dados utilizados para a produção dos resultados preliminares.

O uso de marcações ou rótulos em projetos de software foi explorado para ampliar a compreensão sobre a aplicação destes rótulos em projetos de software hospedados em plataformas sociais, bem como entender a finalidade e vantagens destes.

A temática seguinte, mineração de repositórios de software, tem a finalidade de elucidar os motivos para o uso da aplicação de mineração em repositórios de software. Estes por sua vez, consistem na descoberta de novos padrões e informações que podem ser úteis na tomada de decisão de um projeto. Também é explicado sobre o ciclo que envolve a mineração, o qual começa na coleta de dados e vai até a análise final e obtenção de resultados.

Interligando as temáticas supracitadas, têm-se a ideia da aplicação de técnicas de visualização de informação para dados coletados de repositórios de software, além da aplicação de técnicas de MSR durante o processo. Para tal, foram analisados artigos que utilizam dessa ideia para a obtenção de seus resultados, com foco na utilização de dados relacionados ao uso de rótulos em repositórios de software.

No próximo capítulo serão apresentadas as metas e questões de pesquisa deste trabalho. Também serão apresentados resultados preliminares obtidos e o cronograma de atividades a ser seguido.

Capítulo 3

Método e Resultados

3.1. Metas e Questões de Pesquisa

Existem várias questões que permeiam o uso de rótulos em repositórios de software, sendo estas diversas e abrangentes em várias áreas, como organização de repositórios, comunicação entre desenvolvedores, redução do tempo de vida de uma tarefa, entre outras. As principais questões que buscamos responder com este estudo são:

- O projeto faz uso de rótulos?
- O tempo de vida de uma tarefa é influenciado pelo rótulo atribuído?
- O uso de rótulos melhora a comunicação entre desenvolvedores?
- Qual é o efeito global do uso de rótulos no tempo de conclusão das tarefas, na comunicação e na forma da conclusão das tarefas?

A primeira questão trata da premissa que o projeto a ser analisado precisa definir e utilizar rótulos em suas tarefas. A não satisfação desta questão, obviamente, torna desnecessário o estudo dos efeitos de rótulos no projeto. Vencida essa barreira, ainda é necessário compreender como os rótulos são utilizados. Por exemplo, as associações de rótulos a uma tarefa e a utilização de rótulos ao longo do tempo são indicadores da forma com que os rótulos são utilizados.

Entretanto, a simples utilização de rótulos não implica em efeitos nas atividades de engenharia de software. As questões seguintes tratam desse aspecto. Uma das razões da utilização de rótulos é externar as intenções de gerenciamento e fomentar as ações de desenvolvedores. Isso pode afetar o tempo necessário para concluir uma tarefa e a quantidade de atividades, representadas pelas mensagens trocadas na tarefa. Além disso, nem sempre a relação entre os rótulos e os possíveis efeitos em seu ciclo de vida e comunicação são tão diretamente perceptíveis. A última questão atenta-se justamente a isso.

Assim, técnicas de visualização de dados foram aplicadas para auxiliar nas buscas para as respostas dessas questões, pois estas auxiliam no processo de entendimento e reconhecimento de padrões previamente desconhecidos de grandes quantidades de dados.

Para que estas questões de estudo fossem respondidas, foi necessário a aplicação de um método completo, que percorra com eficiência as etapas de coleta, pré-processamento e aplicação de técnicas de mineração visual de dados. Todo o processo discutido nesse trabalho pode ser aplicado para qualquer repositório de software, gerando resultados consistentes e satisfatórios.

Dentre as diversas técnicas de visualização de informação existentes, foram escolhidas quatro que evidenciam diferentes aspectos sobre bases de dados. Estas foram: *streamgraph*, a qual se mostra eficiente para evidenciar o aspecto temporal em base de dados, mostrando variações nas frequências dos dados ao longo do tempo; *box plot*, utilizado para representar a distribuição estatística dos dados e seus valores discrepantes; técnicas de visualização de grafos, a fim de se observar a relação entre as instâncias de dados; e o diagrama de Sankey, que ilustra informações quantitativas sobre fluxos de dados, seus relacionamentos e transformações.

3.2. Método de Pesquisa – Mineração Visual de Dados

As etapas necessárias para um processo de mineração visual de dados aplicado ao contexto de repositórios de software são agrupadas em: coleta e pré-processamento; extração e visualização de padrões; e pós-processamento. Essa divisão também inclui uma fase anterior ao processo, composta pela aquisição do conhecimento do domínio, e uma fase posterior, composta pela utilização do conhecimento. Podem ocorrer múltiplos ciclos envolvendo as três etapas principais, justamente o que confere o caráter iterativo ao processo. Para cada questão, uma ou mais visualizações foram concebidas considerando tais etapas. Na Seção 3.3, os resultados de cada visualização em relação às questões de pesquisa são apresentados segundo esse processo.

3.2.1. Conhecimento do domínio

Na etapa de conhecimento de domínio é fundamental a identificação do problema, que requer um estudo do domínio – como o apresentado no capítulo anterior. Esse estudo de domínio é imprescindível a fim de adquirir um conhecimento inicial e subsidiar as etapas posteriores do processo.

Nesta etapa, também são definidos os objetivos e metas a serem alcançadas no domínio. No contexto dessa proposta, tais objetivos e metas foram apresentados na Seção 3.1.

3.2.2. Coleta e Pré-processamento dos Dados

Para a utilização de técnicas de visualização de dados, frequentemente é necessária a extração de dados e pré-processamento destes, a fim de torná-los compatíveis com o propósito desejado.

Esta extração pode vir por diversos meios (API de repositórios de software, uso de bases de dados, raspagem de dados, entre outros) e possui a finalidade de obter os dados que futuramente serão a fonte de alimentação das técnicas de visualização selecionadas.

Geralmente, ao realizar a extração de algum dado qualquer, este não está na forma desejada, podendo conter informações desnecessárias ou ainda não estar na estrutura de dados adequada para a situação. Assim, é realizado o pré-processamento destes, processo o qual adaptará os dados obtidos para o formato desejado. Os *scripts* utilizados em tal processo devem levar em consideração o formato dos dados obtidos e do formato aceito pela técnica de visualização selecionada, gerando uma estrutura de dados compatível com a entrada desejada pela biblioteca que implementa a técnica de visualização de informação.

3.2.3. Extração e Visualização de Padrões

Esta é a principal etapa do processo e é direcionada ao cumprimento dos objetivos definidos na Seção 3.1. Existem diversas técnicas tanto no contexto de mineração de dados quanto no contexto de visualização de dados, e cada uma destas possui uma função distinta que deve ser considerada durante a escolha de uma das técnicas de ambas as áreas.

Tratando-se especificamente do contexto de visualização de informação, não existe uma única técnica de visualização que irá prover todas as respostas desejadas, dado que cada técnica de visualização tende a focar em um aspecto em específico – temporal, hierárquico, relacional, geoespacial, multidimensional, etc. O maior ganho se dá na utilização de diversas técnicas de visualização para a mesma base de dados para capturar diferentes aspectos da base e correlacioná-los. Para algumas das questões proposta por este estudo, foi necessária a aplicação de mais de uma técnica de visualização de informação, para garantir maior completude e embasamento à questão proposta.

Nos resultados apresentados na Seção 3.3, foram utilizadas técnicas com diferentes focos, sendo estes em aspectos temporais, estatísticos e relacionais. Também foi dada prioridade a técnicas de visualização que tenham mecanismos de interações por apoiarem a análise visual e percepção por parte do usuário. Assim, a análise se torna dinâmica, o que pode facilitar a compreensão e dedução por parte do usuário em diversos aspectos.

3.2.4. Pós-processamento

Nesta etapa o usuário pode decidir extrair outros padrões, executando novamente o ciclo principal do processo, voltando a etapa de pré-processamento e eventualmente mudando a escolha de algoritmos e parâmetros. A aplicação desta etapa pode envolver a escolha de novos repositórios para análise, bem como pela mudança dos algoritmos utilizados, a fim de aprimorá-los para a coleta de maiores quantidades de dados.

3.2.5. Utilização de Conhecimento

Na etapa de utilização do conhecimento todo o conhecimento adquirido no decorrer das etapas anteriores é aplicado. Esta aplicação pode ser feita através do uso de Sistemas Inteligentes ou diretamente pelo usuário final em algum processo de tomada de decisão, sendo este último o cenário típico em engenharia de software.

3.3. Resultados

Nessa seção são apresentados os resultados obtidos para visualizar aspectos gerais sobre o uso de rótulos no contexto de um repositório de software. Para a geração de todas as visualizações foi aplicado o processo apresentado na Capítulo 3, adaptando os *scripts* de pré-processamento de acordo com o necessário. Todos os *scripts* e resultados foram realizados com base nas tarefas e rótulos empregados no repositório do software NextCloud[1], que consiste em uma plataforma *open source* de compartilhamento de arquivos e colaboração auto-hospedada na web. O projeto NextCloud está hospedado na plataforma social de desenvolvimento GitHub no endereço <https://github.com/nextcloud>.

Em relação ao método de pesquisa empregado, apresentado na Seção 3.2, a etapa de conhecimento do domínio e grande parte da etapa coleta e pré-processamento dos dados são comuns a todos os resultados apresentados. Os resultados referentes ao conhecimento de domínio foram apresentados no Capítulo 2 e aqueles referentes à coleta estão na Seção 3.4. As demais etapas de extração, visualização de padrões e pós-processamento são particulares a cada visualização, sendo apresentados individualmente nas respectivas subseções a seguir. Todos os *scripts* desenvolvidos e arquivos de dados coletados para execução deste estudo encontram-se disponíveis em <https://github.com/claudiaps/TCC>.

3.4. Coleta de dados

Considerando o volume de dados a serem tratados, foram desenvolvidos pequenos programas (*scripts*) para a coleta e pré-processamento. Neste estudo foram utilizadas as linguagens Python e JavaScript para esta função, pois estas possuem várias bibliotecas e funcionalidades que facilitam a manipulação dos dados, além da familiaridade da autora com tais linguagens.

Para coletar os dados necessários para esse estudo, foi utilizada a API REST V3 do GitHub[2]. Esta dispõe de diversos métodos para recuperar e coletar dados relacionados aos repositórios, usuários, tarefas e rótulos, além de ser gratuita[3]. A API REST V3 do GitHub foi escolhida pois essa apresenta consistência desejada nos dados obtidos, além da disponibilização destes em tempo real.

[1] <https://nextcloud.com>
[2] <https://developer.github.com/v3>
[3] No modo gratuito, aplicam-se restrições quanto à quantidade de requisições por período, atualmente de 5 mil requisições por hora.

O *script* utilizado para realizar as requisições foi feito na linguagem de programação Python, com o auxílio do módulo PyGitHub[4], o qual abstrai parte da complexidade da API do GitHub. Após a coleta dos dados, estes foram armazenados em um arquivo JSON, que foi base para a etapa de pré-processamento de todas as visualizações de dados geradas.

Os *scripts* de pré-processamento foram pensados e implementados de acordo com o formato de dados esperado pelas técnicas de visualização de dados selecionadas e de acordo com os resultados esperados. Para algumas das técnicas escolhidas, *box plot* e *streamgraph*, bastou o rearranjo dos dados para o formato esperado. Para as outras técnicas, grafo e diagrama de Sankey, foi necessária a implementação de algoritmos mais robustos, que criassem alguma relação entre os dados, seja esta quantitativa ou qualitativa, além da necessidade da utilização de funções para manipulação de valores temporais e textuais. Aspectos de coleta e pré-processamento específicos a cada visualização serão tratados em suas respectivas seções.

3.5. Análise da quantidade de rótulos por tarefa

Para a análise da quantidade de rótulos que são aplicados às tarefas no repositório NextCloud, foi utilizada a técnica de visualização *box plot*, pois esta além de representar de forma eficiente a distribuição estatística de dados numéricos, exibe valores considerados *outliers*. O objetivo neste caso é analisar, por exemplo, as menores e maiores quantidade de rótulos que são associados a uma tarefa, bem como obter informações estatísticas sobre estas frequências de atribuições, como medianas e valores discrepantes.

Considerando os dados previamente coletados, eles foram submetidos a um novo *script* de pré-processamento para realizar a contagem do número de rótulos por tarefa. Esse resultado é disposto em um único vetor, o qual é submetido a ferramenta de *box plot* da biblioteca Plotly. O pré-processamento foi novamente implementado na linguagem de programação JavaScript e a visualização em Python.

A Figura 3.1 mostra o resultado obtido. O eixo y representa o número de rótulos atribuídos às tarefas e o *box plot* denota a distribuição das tarefas com base no eixo y. A versão dinâmica dessas visualizações está localizada em <http://aretha.pro.br/tags/boxplot-rotulo.html>.

Analisando o resultado, nota-se que o a quantidade de rótulos atribuídos a uma tarefa varia entre 0 e 6, mas que algumas tarefas podem conter quantidade maiores de atribuição, configurando *outliers* nos valores 6 e 8. O primeiro quartil apresenta valor 1, o que significa que 75% das tarefas tem pelo menos um rótulo associado. Já a mediana é dada pelo valor 2. O resultado apresentada pelo *box plot* da Figura 3.1 é um bom indicativo do gerenciamento desse repositório, pois denota uma preocupação da comunidade que mantém o NextCloud em atribuir rótulos as tarefas.

[4] <https://github.com/PyGithub/PyGithub>

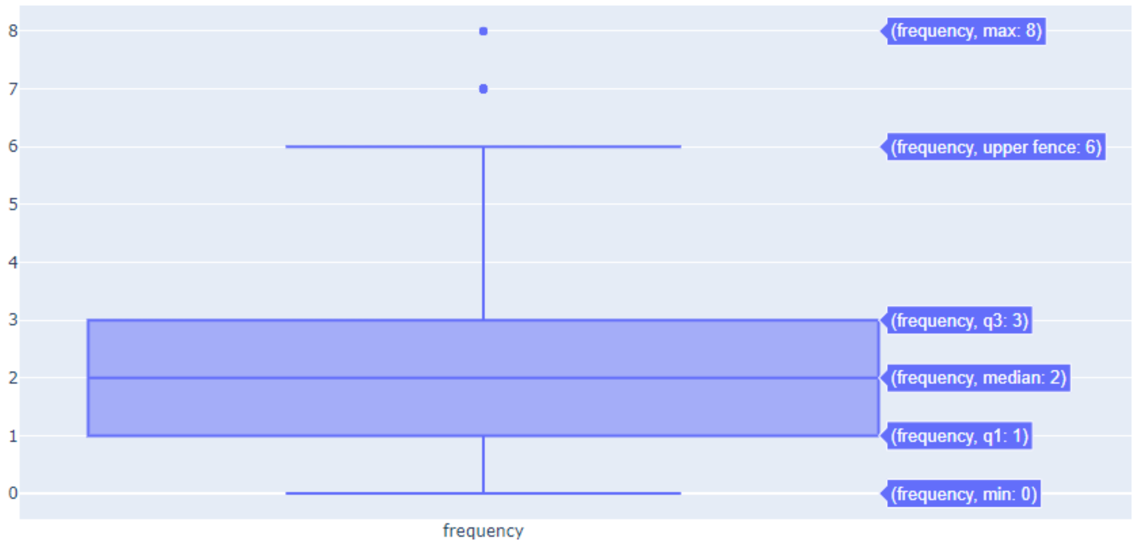


Figura 3.1. *Box plot* gerado a partir dos do número de rótulos atribuídos às tarefas do repositório de software da ferramenta NextCloud.
Fonte: Autoria própria (2019).

3.6. Análise do uso de rótulos ao longo do tempo

A primeira análise realizada usou a técnica *streamgraph* para visualizar a variação temporal no uso de rótulos no repositório NextCloud. Para coletar os dados relacionados aos rótulos (nome e frequência destes no decorrer dos meses durante os anos de vida do repositório) foi necessário coletar todas as tarefas do repositório, e então armazenar o resultado da busca em um arquivo JSON.

O arquivo JSON com os dados coletados foi utilizado para a geração de um arquivo intermediário JSON, o qual contém o nome do rótulo, data (ano e mês) e frequência de cada um nesta data, considerando também datas com frequência 0 para tal rótulo.

Entretanto, o arquivo JSON é um formato incompatível com o necessário para a geração da visualização *streamgraph* na linguagem de programação R. Logo, os dados ainda foram submetidos a um terceiro *script*, o qual modela os dados do segundo arquivo JSON (nome do rótulo, data e frequência) para um arquivo em formato CSV, o qual contém as mesmas informações do arquivo JSON de entrada.

Os dados do arquivo CSV foram utilizados para alimentar a biblioteca *streamgraph* na linguagem de programação R. A Figura 3.2 mostra o resultado obtido após a geração da *streamgraph*, utilizando como dados os rótulos presentes no decorrer do tempo nos repositórios da ferramenta NextCloud. A versão dinâmica dessas visualizações está localizada em <http://aretha.pro.br/tags/streamgraph.html>.

Quando analisados os resultados obtidos minerando os dados dos rótulos do repositório NextCloud, é possível observar que o uso de rótulos é quantitativamente representativo desde o final de 2016. A quantidade total de rótulos é relativamente alta, contando com 77 rótulos, onde os 5 com maior frequência total são: ‘*bug*’, ‘*enhancement*’, ‘*1. to develop*’, ‘*to review*’ e ‘*design*’. O

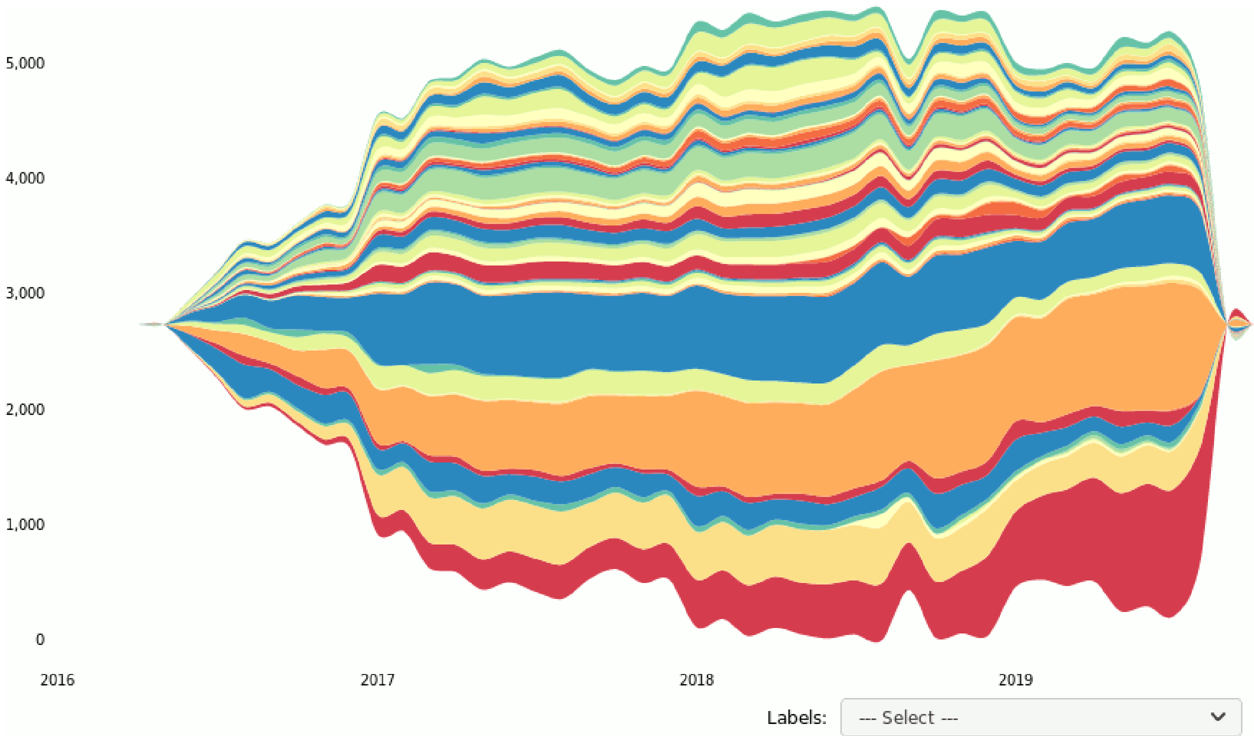
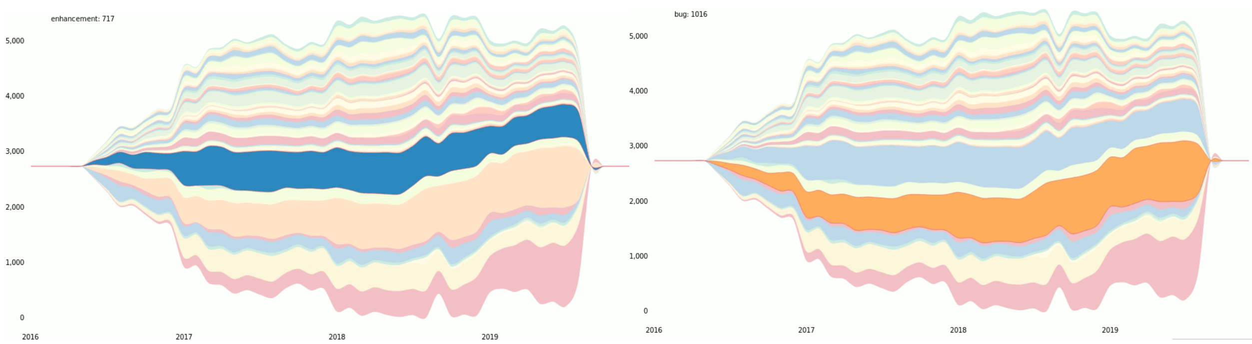


Figura 3.2. *Streamgraph* gerada para os dados dos rótulos coletados do repositório NextCloud.
Fonte: Autoria própria (2019).



(a) Rótulo '*enhancement*' destacado em azul, próximo ao eixo central.

(b) Rótulo '*bug*' destacado em laranja, abaixo do eixo central.

Figura 3.3. *Streamgraph* para o repositório NextCloud, com destaque para as faixas que representam os rótulos mais utilizadas.
Fonte: Autoria própria (2019).

rótulo '*needs triage*' teve aumento considerável nos últimos meses, mas sua distribuição não é uniforme no decorrer do tempo de vida do projeto.

A Figura 3.3 aponta com destaque as faixas da *streamgraph* que representam as frequências dos rótulos mais frequente, onde a imagem *a* representa o rótulo *'enhancement'* e a imagem *b* o rótulo *'bug'*. O rótulo *'bug'* é o mais frequente ao longo do tempo, com aproximadamente 1000 aplicações deste rótulo à tarefas por mês. O segundo rótulo mais frequente deste repositório é o rótulo *'enhancement'*, com frequência máxima de 755, no primeiro semestre de 2018.

Também foi gerada uma faixa para as tarefas sem rótulos, representada pela Figura 3.4. É perceptível que o número de tarefas sem rótulos associados é bem baixo, totalizando máximo de 208 em meados de 2018. Isso denota que a comunidade que mantém o projeto NextCloud possui uma preocupação para com o uso de rótulos em suas tarefas.

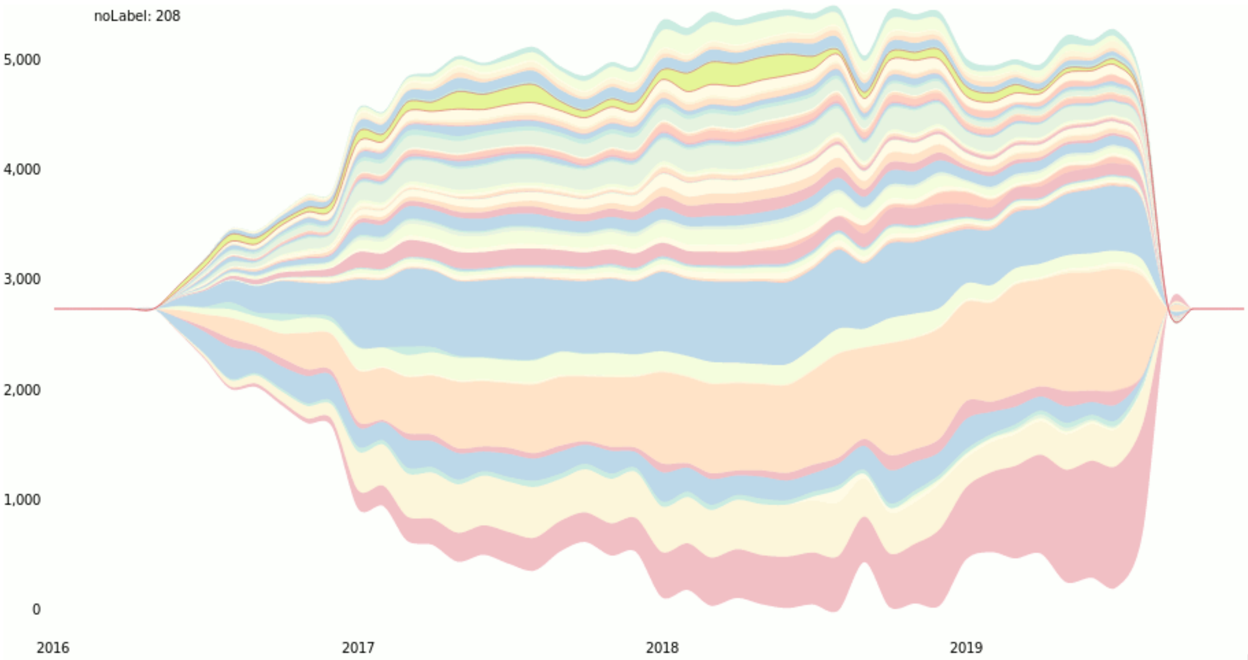


Figura 3.4. *Streamgraph* para o repositório Nextcloud, com destaque para a faixa para tarefas sem rótulos (sétima faixa no topo da figura, em verde limão).
Fonte: Autoria própria (2019).

3.7. Análise dos tempos de conclusão das tarefas

O tempo de conclusão de uma tarefa consiste no período entre a criação e a conclusão desta, conclusão a qual é dada pelo fechamento desta na plataforma social de desenvolvimento analisada. Salienta-se que isso não necessariamente envolve a implementação da solução para o problema proposto, mas é um forte indicativo.

Para analisar a contribuição do uso de rótulos neste aspecto, foi utilizada a técnica de visualização *box plot*, pois esta além de representar de forma eficiente a distribuição estatística de dados numéricos, exibe valores considerados *outliers*.

Após a aquisição dos dados através da API V3 da plataforma do GitHub, estes foram submetidos a um *script* para pré-processar os dados, implementado na linguagem de programação JavaScript. Este possui a finalidade de calcular o tempo de conclusão de tarefas associadas a cada rótulo individualmente, com base na data de criação e fechamento de cada tarefa. O algoritmo gera um arquivo no formato JSON contendo o nome do rótulo, e o tempo de conclusão de todas as tarefas as quais possuem este associado, calculado em dias. Importante ressaltar que também foi gerado um *box plot* para tarefas que não possuem rótulos atribuídos, analisando os mesmos aspectos.

Para a geração do *box plot*, foi utilizada a linguagem de programação Python, juntamente com a biblioteca Plotly[5]. Essa biblioteca fornece implementações de diversas técnicas de visualização de dados para as linguagens de programação Python, JavaScript e R. Esta foi escolhida por ser livre, de fácil utilização e gerar visualizações gráficas interativas.

A biblioteca aceita como entrada dados numéricos em forma de vetor e é possível ter mais de um vetor, configurando assim diversos gráficos *box plot* em uma mesma visualização. No caso deste estudo, temos um vetor de dados numéricos (tempos de conclusão) para cada rótulo. Dessa forma, cada um dos rótulos configurou um *box plot* e os valores atribuídos a estes foram os tempos de conclusão de tarefas que utilizam deste rótulo, ou que não possuem rótulos.

A Figura 3.5 denota os resultados obtidos após a aplicação do processo supracitado. Abaixo de cada *box plot* individual são exibidos os nomes dos rótulos. A escala utilizada para representar o tempo de conclusão no eixo y está em dias. Além disso, há uma legenda no lado direito da visualização, mostrando os nomes dos rótulos e as cores utilizadas para representar estes. Também são mostrados através de pontos desconexos os *outliers* – observações que são numericamente distantes do resto dos dados – de cada *box plot* individual. A versão dinâmica dessa visualização está localizada em <http://aretha.pro.br/tags/boxplot-tempo-conclusao.html>.

Analisando de forma empírica os resultados obtidos, é possível notar que alguns rótulos estão associados a tarefas que possuem tempo de conclusão bastante variado. Por exemplo, o rótulo *bug* tem a maior porção de suas tarefas concluídas entre 0 e 60 dias, com a mediana em 2. Entretanto, quando analisados os *outliers*, estes vão de 61 até 1054 dias. Dessa forma, há uma tarefa com o rótulo *bug* que demorou 1054 dias para ser concluída. Com valores distintos, mas exibindo o mesmo padrão, têm-se os rótulos '*0. Need triage*', '*enhancement*', '*3. to review*', '*design*', '*needs info*', '*1. to develop*', '*feature: dev*', '*feature: sharing*' e "*high*'.

Quando analisado o *box plot* para as tarefas não rotuladas, vemos que primeiro quartil e a mediana possuem o mesmo valor, 0, e o *upper fence* possui valor 22. Seus valores discrepantes vão de 23 até 550 dias. Pode-se concluir então, que tarefas sem rótulos no projeto analisado são fechadas mais rapidamente. Isso pode ser explicado pelo fato dessas tarefas serem tão simples, de baixa complexidade técnica ou de fácil gerenciamento (esta pode estar duplicada e já ter sido resolvida), que não houve uma preocupação em atribuir rótulos as mesmas.

Tratando os rótulos que possuem os maiores valores, pode-se observar o rótulo *stale* como o de maior duração, tendo tarefas que vão de 237 a 1054 dias para serem concluídas. Este é seguido pelos rótulos *standardisation* e *feature: logging* com tarefas fechadas em no mínimo de 0 para ambos, e máximo de 873 e 739 dias respectivamente. Sobre os que possuem os menores tempos de fechamento de tarefas, pode-se destacar *feature: dependecies*, *javascript* e *php*, todos com valores mínimos de 0 dias e valores máximos de 5, 7, e 11 dias, respectivamente. Estes dados

[5] <https://plot.ly/python/>

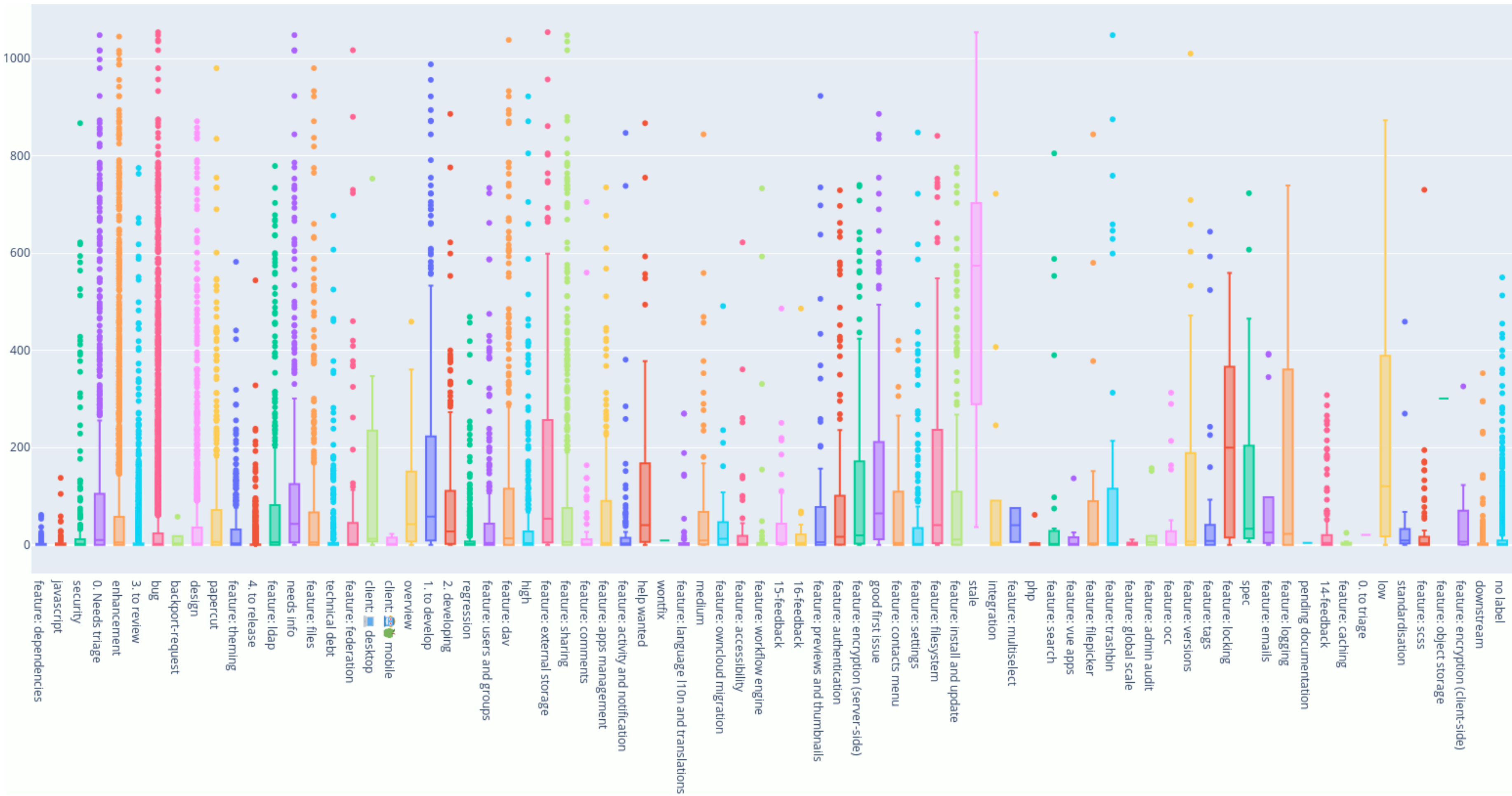


Figura 3.5. Gráfico *box plot* gerado a partir dos tempo de conclusão de tarefas com e sem rótulos. Cada *box plot* representa um rótulo, e o eixo *y* representa o tempo de conclusão de tarefas que têm este rótulo atribuído.
Fonte: Autoria própria (2019).

foram analisados com base nos valores contidos nos quartis dos *box plots*, não estando sendo considerado os *outliers* nesta análise.

3.8. Análise do número comentários para tarefas rotuladas

Ainda reaproveitando a biblioteca Plotly, os dados já coletados e algumas partes dos *scripts* de pré-processamento já utilizados, foi gerado mais um *box plot*, o qual ilustra a quantidade de comentários das tarefas rotuladas. De maneira geral, os comentários têm a finalidade de abrir alguma discussão relacionada ao conteúdo da tarefa, seja este a solução de um erro, possíveis melhorias, questionamentos, dúvidas, etc.

Nesta nova visualização busca-se analisar a quantidade de comentários associados a cada tarefa, formando um vetor que contém a quantidade de comentários para cada tarefa. Este vetor foi então utilizado como entrada de dados na ferramenta de *box plot* da biblioteca Plotly. O pré-processamento e a geração da visualização foram implementados respectivamente nas linguagens JavaScript e Python.

Ao gerar o *box plot* percebeu-se um grande número de *outliers* presentes. Por isso, esta visualização passou por um processo de pós-processamento, onde foi optado pela remoção de parte dos *outliers*. Para a remoção destes, foi utilizada a biblioteca para a linguagem Python, Outlier [6]. Nesta, basta passar o vetor com os dados e o parâmetro *alpha* dejesado, neste caso, 0,2.

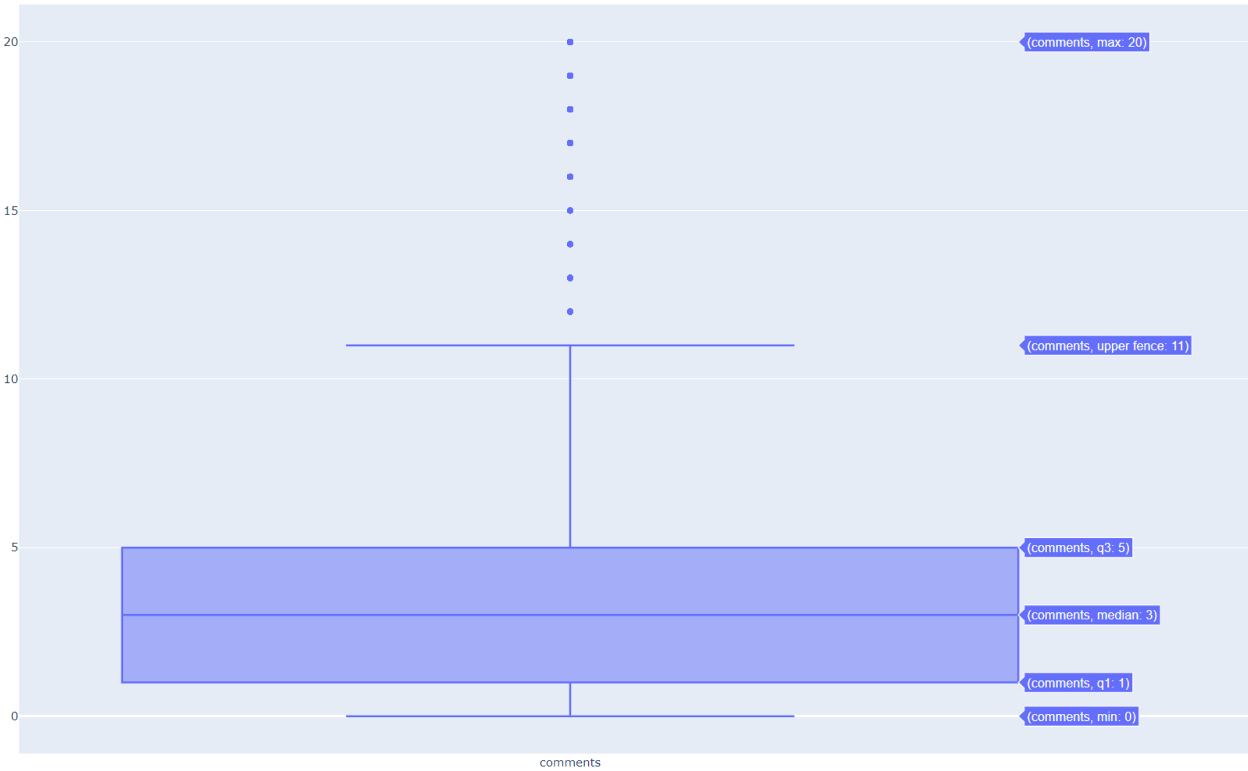


Figura 3.6. *Box plot* gerado a partir do número de comentários associados à tarefas rotuladas.
Fonte: Autoria própria (2019).

[6] <https://github.com/archongum/outlier>

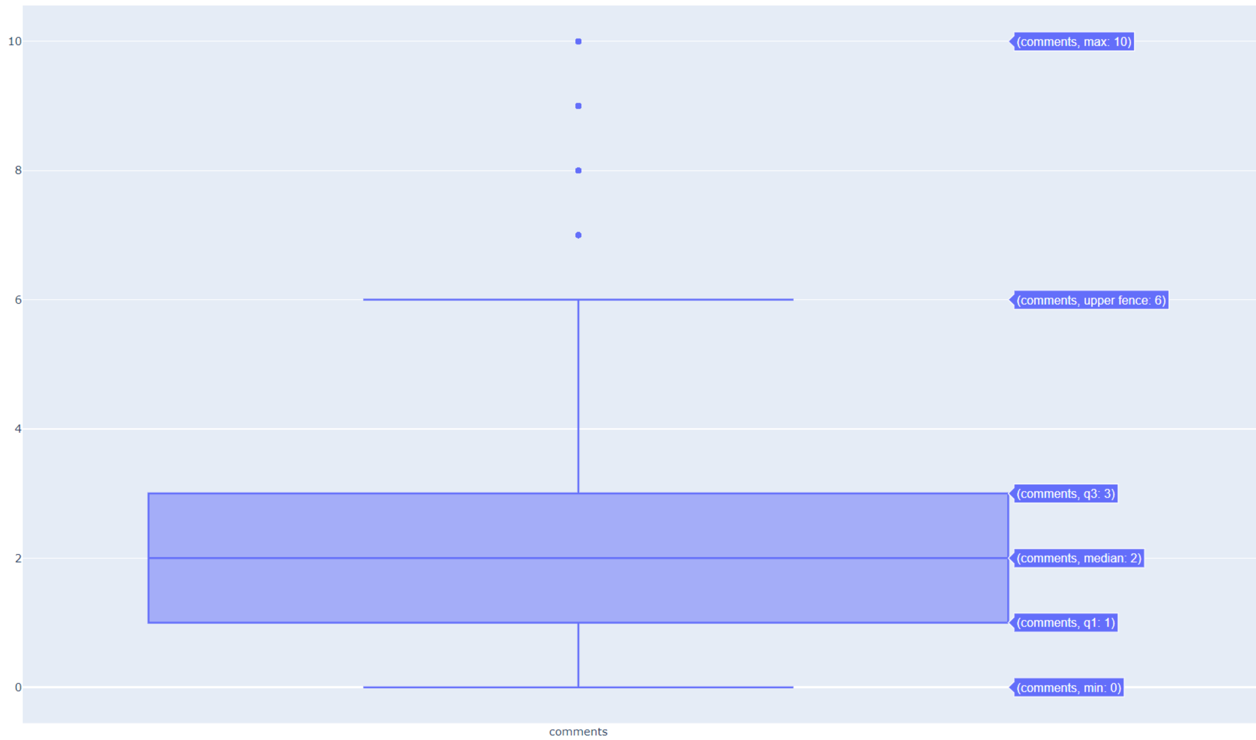


Figura 3.7. *Box plot* gerado a partir do número de comentários associados à tarefas que não possuem rótulos associados.
Fonte: Autoria própria (2019).

A Figura 3.6 denota o *box plot* para o número de comentários em tarefas rotuladas. É possível notar que 50% das tarefas rotuladas possuem entre 1 e 5 comentários, tendo como valor da mediana 3 comentários. Apenas 25% das tarefas possuem mais que 5 comentários, terceiro quartil. Já a Figura 3.7 aplica a mesma técnica supracitada, mas para as tarefas sem rótulos. Nota-se que 50% das tarefas não-rotuladas possuem entre 1 e 3 comentários, 25% não possuem comentários e 25% das tarefas não-rotuladas possuem mais que 3 comentários. A versão dinâmica dessas visualizações está localizada em <http://aretha.pro.br/tags/boxplot-comentarios-com-rotulos.html> e <http://aretha.pro.br/tags/boxplot-comentarios-sem-rotulos.html>.

Analisando a Figura 3.6 e a Figura 3.7, temos que a mediana e o terceiro quartil para tarefas sem rótulos é 2 e 3, respectivamente, enquanto para as rotuladas é 3 e 5, o que numericamente indica um maior número de comentários para tarefas rotuladas. Esses valores indicam que o uso de rótulos no repositório do software NextCloud pode contribuir na comunicação dos desenvolvedores.

3.9. Análise da coocorrência entre rótulos

A fim de analisar a coocorrência de rótulos em tarefas em um dado repositório, ou seja, rótulos utilizados em conjunto em uma única tarefa, foi implementado uma visualização baseada em grafos. Nesta, os rótulos são representados pelos nós do grafo e suas coocorrências pelas arestas. Uma aresta é adicionada entre dois nós (rótulos) se eles são associados em conjunto em pelo menos uma tarefa. O peso de cada aresta entre dois nós é igual ao número de tarefas nas quais os dois nós aparecem simultaneamente como rótulos. O peso dos nós é igual ao seu grau, ou seja, o número de arestas que este recebe. Dada essa formulação, este é um grafo não direcionado, que permite identificar quais rótulos são mais populares no repositório selecionado e quais são comumente utilizados em conjunto.

A implementação desta ferramenta foi realizada em três etapas: aquisição dos dados, pré-processamento e configuração de *layout*. A primeira etapa se deu pelo reaproveitamento dos dados previamente coletados, não sendo necessária nenhuma outra coleta. O ponto de interesse dos dados estava na obtenção dos rótulos de cada tarefa, para na etapa seguinte estes serem relacionados

Na etapa de pré-processamento, implementada na linguagem de programação JavaScript, a correlação por entre os rótulos foi estabelecida: primeiramente foram coletados todos os rótulos de uma tarefa, e realizada uma combinação de seus pares. Por exemplo, se forem consideradas os rótulos fictícios *rótulo 1*, *rótulo 2* e *rótulo 3*, estes seriam combinados de forma a gerar a saída *rótulo 1 - rótulo 2*, *rótulo 2 - rótulo 3* e *rótulo 1 - rótulo 3*. Sempre que uma combinação gerada era igual a outra já existente, o peso da combinação existente era incrementado.

Após as relações terem sido criadas e seus pesos devidamente atribuídos, foi necessário salvar os dados do grafo formado em um dos formatos de arquivo aceitos pela ferramenta Gephi[7]. Essa ferramenta foi utilizada para a exibição do grafo e configuração de seu *layout*. O formato do arquivo escolhido foi o GEXF, o qual foi desenvolvido especialmente para a ferramenta por seus desenvolvedores, e permite uma manipulação mais completa do grafo. O arquivo GEXF trata-se de um arquivo na linguagem de marcação XML, e pode ser bastante complexo para se configurar manualmente. A fim de otimizar o processo, foi utilizada uma biblioteca na linguagem JavaScript hospedada no GitHub, GEXF JavaScript Library [8], para converter dados armazenados em um objeto JavaScript para o GEXF.

O arquivo criado na etapa anterior foi importado na ferramenta Gephi, iniciando a etapa de configuração de *layout* da visualização. Para a obtenção do resultado exibido na Figura 3.8, foram aplicados os seguintes passos: aplicação de *layout* através da utilização do algoritmo *Force Atlas* com força de repulsão em 10.000; ocultação de arestas que possuem peso menor que 2; aplicação de uma escala de cor verde para os nós, quanto mais escuro o tom de verde do nó maior é o grau do nó; tamanho dos nós é proporcional ao seu respectivo grau; ajuste na espessura das arestas, tornando-as mais espessas de acordo com o seu peso (co-ocorrência dos rótulos interligados); exibição das *labels* dos nós, ou seja, dos nomes dos rótulos, sendo o tamanho da *label* dos nós proporcional ao grau do nó; e aplicação de opacidade das arestas em 40%. A opacidade de arestas e filtragem de arestas com peso menor que 2 foram aplicadas para reduzir a oclusão visual.

Apesar da informação do nome dos rótulos no grafo ser bastante interessante, foi percebido através da Figura 3.8 que estes podem gerar oclusão visual na visualização, principalmente

[7] <https://gephi.org/>
[8] <https://github.com/Yomguithereal/gexf>

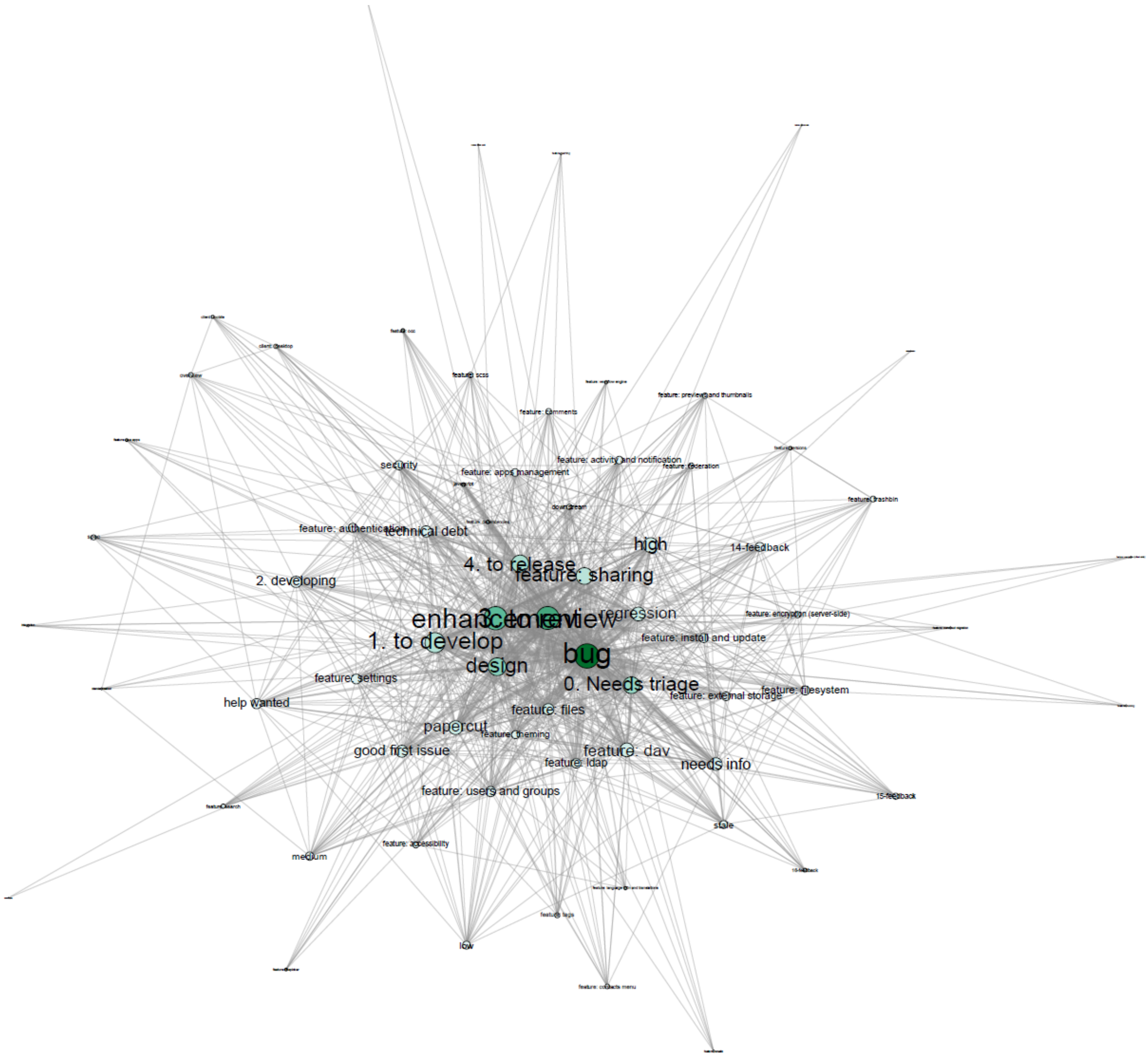


Figura 3.8. Grafo de coocorrência dos rótulos no repositório NextCloud com exibição do nome dos rótulos. Fonte: Autoria própria (2019)

em setores do grafo onde os nós estão muito conectados e suas arestas possuem pesos maiores. A região mais afetada foi a central, onde estão contidos os nós que representam os rótulos *bug*, *enhancement*, *1. to develop*, *3. to review*, entre outros. Uma forma de contornar este problema é analisar o mesmo grafo com e sem as *labels* dos rótulos simultaneamente, correlacionando as duas imagens. A Figura 3.9 demonstra o mesmo grafo sem os nomes dos rótulos. Salienta-se também que este tipo de visualização tem o seu completo potencial alcançado com uso de interações sobre o grafo, o que uma impressão em papel não permite, somente exibindo-o na ferramenta Gephi.

A fim de aprimorar a compreensão do grafo, a Figura 3.9 teve algumas etapas de configuração um pouco diferentes dos resultados apresentados pela Figura 3.8. As mudanças foram relativas a porcentagem da opacidade das arestas, de 40% para 100%, ajuste na escala de cores dos nós, tornando-os mais escuros, e aumento da espessura da borda dos nós. A arquivo do grafo que pode ser visualizado na ferramenta Gephi está localizado em <http://aretha.pro.br/tags/grafo.gephi>.

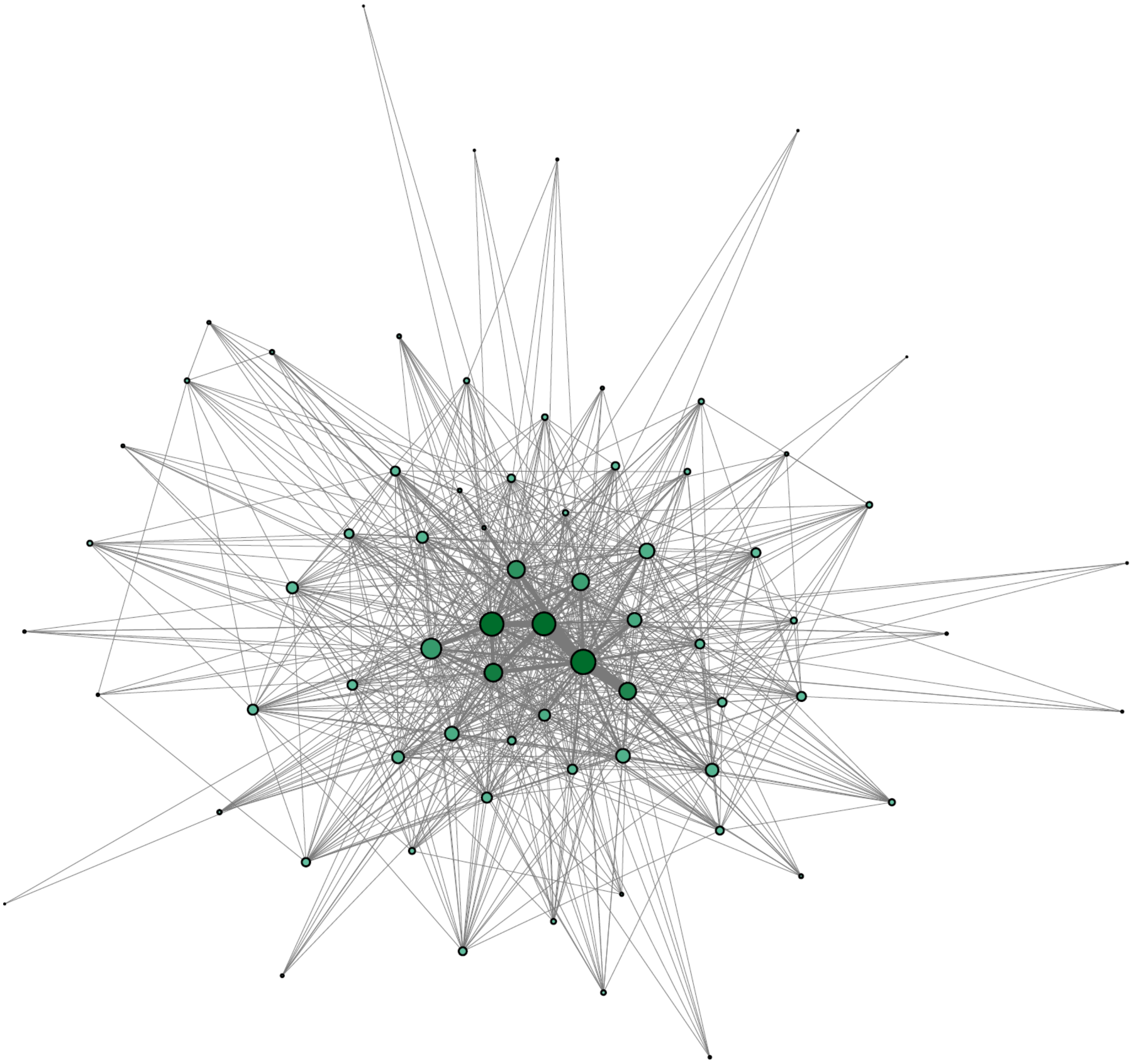


Figura 3.9. Grafo da correlação dos rótulos no repositório NextCloud, ocultando o nome dos rótulos, diminuindo a opacidade das arestas e realçando a borda dos nós.
Fonte: Autoria própria (2019).

Correlacionando as Figuras 3.8 e 3.9, pode-se perceber que o grafo é altamente conectado, o que denota uma tendência de na comunidade do NextCloud, de usar mais de um rótulo por tarefa. Olhando para a área central do grafo, é possível perceber que os rótulos *bug*, *enhancement*, *1. to develop*, *3. to review*, *design* e *4. to release* possuem alta correlação, e que o rótulo *bug* em especial representa o maior nó do grafo. Este fato pode estar relacionado ao fato que *bugs* em sistemas estão sempre relacionados a alguma funcionalidade, linguagem de programação, etapas de desenvolvimento, etc., denotando que o rótulo pode ser utilizado para sinalizar o problema em alguma atividade que é categorizada por outro rótulo. Outra possível contribuição que esta visualização pode fornecer é a sugestão rótulos secundários para uma tarefa, de acordo com um rótulo inicialmente associado pelo usuário. Isto poderia facilitar o gerenciamento do repositório e automatizar parte do processo de associação de rótulos.

3.10. Integrando as Visualizações

Após a coleta e extração de informações sobre a utilização de rótulos em tarefas, através do uso de técnicas de Visualização de Informação, como *Streamgraph*, *box plot* e grafo, uma última técnica foi aplicada, com o objetivo de interligar as informações já obtidas. Assim, os resultados anteriores além de expressarem características sobre tarefas rotuladas, serviram de base para a implementação do diagrama de Sankey apresentado nessa seção.

Este por sua vez, tem a finalidade de correlacionar rótulos e diversos aspectos sobre as tarefas e analisar características das tarefas de forma individual. Sua implementação foi realizada utilizando a ferramenta para diagrama de Sankey da biblioteca *Google Charts*[9], utilizando a linguagem de marcação HTML com JavaScript. O processo de implementação desta visualização foi mais complexo que das anteriores, pois muitos dos dados não haviam sido coletados ainda, além desta requerer uma entrada de dados bem específica e heterogênea.

De modo geral, esse diagrama Sankey será formado por quatro eixos: o primeiro eixo é formado por rótulos selecionados, o que faz que somente tarefas que contenham esses rótulos sejam mostradas nessa visualização; o segundo eixo é dado pelo número de comentários presentes em cada tarefa; o terceiro eixo é dado pelo tempo de conclusão para o fechamento das tarefas; e o quarto e último eixo descreve se a tarefa não tem código nos comentários, tem código nos comentários, tem *pull request* associado ou tem *pull request* aceito.

A etapa inicial, e que configurou toda a essência da visualização, se deu pela escolha dos rótulos a serem utilizados na composição do primeiro eixo do diagrama. Tal escolha foi pautada pela análise da *Streamgraph*, do *box plot* para tempo de conclusão de tarefas com base em seus rótulos e e do grafo para a correlação dos rótulos. A partir da *Streamgraph* foram selecionados rótulos mais utilizados no projeto, sendo estes *bug* e *enhancement*, e os que embora não muito volumosos, possuem constância dentro de todo o tempo de vida do projeto, sendo *3. to review* e *design*. Ainda analisando a *Streamgraph*, é perceptível que o rótulo *0. Needs triage* e feature: sharing estão sendo mais utilizados com o passar do tempo, tornando interessante a utilização destes no diagrama de Sankey. Analisando o *box plot* de tempo de fechamento de tarefas, temos que o rótulo *stale* é o que possui maior variação interquartílica sem a presença de *outliers*, característica a qual levou a seleção deste para compor o primeiro eixo do diagrama. O rótulo *14. feedback* foi escolhido pois este só apresenta associações em meados de 2018. Além destes rótulos, também foram analisadas as tarefas que não possuem nenhum rótulo, a fim de gerar uma comparação entre o uso e o não-uso de rótulos em projetos de software livre.

[9] <https://developers.google.com/chart/interactive/docs/gallery/sankey>

De modo geral, análises referentes à comunicação entre desenvolvedores leva em conta questões quantitativas, que estão atreladas, por exemplo, ao número de mensagens trocadas até a resolução do problema. Essa informação é representada pelo segundo eixo do diagrama que é composto pelo número de comentários em tarefas que possuem pelo menos um dos rótulos apresentados no primeiro eixo associados, ou não possuem rótulos. Estes foram divididos em faixas, as quais foram selecionadas com base no *box plot* de número de comentários nas tarefas rotuladas. As faixas *0 comments, 1 comments, 2-5 comments, 5-11 comments* e *12 + comments* são baseadas nos quartis do *box plot*.

O tempo de conclusão das tarefas configura o terceiro eixo, o qual seguia como base o *box plot* que exibe o tempo de conclusão das tarefas rotuladas ou sem rótulos. Inicialmente foram analisados os valores que definiam um valor próximo a média dos quartis dos *box plots*. Entretanto, isso gerou faixas com pesos desproporcionais. Assim, este eixo foi submetido a uma etapa de pós-processamento, a fim de redefinir os valores das faixas e tornar a visualização mais harmônica e consistente. O tempo de conclusão são medidos em dias, e variando de *0-10 days, 11-120 days, 201-400 days, 401-600 days, 601-800 days, 801-100 days, 1001+ days* e *not closed*, para denotar tarefas que ainda não foram concluídas.

Análises referentes à comunicação entre desenvolvedores também leva em conta questões qualitativas, que em geral estão atreladas ao conteúdo das mensagens trocadas entre desenvolvedores, como por exemplo, a presença de código nas mesmas. O quarto e último eixo busca mostrar as tarefas que tiveram *pull requests* submetidos e/ou aceitos e tarefas que possuem comentários e/ou código nos comentários. Estas informações não estavam contidas na coleta inicial de dados, sendo necessário requisitá-los através da API utilizada separadamente. Este eixo segue uma hierarquia quanto a seus nós: primeiramente foram avaliados os *pull requests* de cada tarefa e, caso esta os contenha, abertos ou fechados, seus comentários não foram analisados. Caso contrário, foi feita uma análise nos textos dos comentários, buscando a marcação utilizada para formatar código nos comentários das tarefas.

Um ponto de atenção sobre o diagrama de Sankey é que este não mantém a relação completa por entre os eixos dos dados. Ou seja, o eixo só está diretamente relacionado com os seus eixos vizinhos, e não com todos os existentes. Utilizando o resultado gerado como exemplo, é tida a relação entre rótulos e número de comentários entre tarefas, número de comentários entre tarefas e tempo de fechamento destas, e assim subsequentemente. Existem outras técnicas que são capazes de manter a relação ente os eixos, como a técnica Coordenadas Paralelas (INSELBERG, 2009). Essa técnica exibe uma polilinha que cruza os eixos em uma posição proporcional ao valor da dimensão exibida pelo eixo, porém geram uma grande oclusão visual pois exibi uma polilinha para cada instância dos dados. O diagrama Sankey foi escolhido pois evita a oclusão visual ao usar fluxos que agrupam as instâncias.

Todo o pré-processamento foi implementado em JavaScript, e os algoritmos que realizaram as requisições extras necessárias foram iplementados em Python, com a utilização do módulo

PyGithub previamente mencionado. A ferramenta utilizada para a geração da visualização espera como estrada um vetor com sub-vetores, onde em cada sub-vetor estão contidos os nós de origem e destino, bem como o peso desta relação.

A Figura 3.10 mostra o resultado obtido após a aplicação das etapas previamente citadas. O primeiro ponto a ser observado é quanto a ordenação dos nós dentro de um eixo, pois alguns não estão numericamente organizados. Isso se deve ao fato da própria ferramenta realizar esta organização, ordenando geralmente os nós de forma decrescente de acordo com o peso que estes recebem. Tal comportamento visa diminuir a oclusão visual que esta técnica pode gerar. A versão dinâmica dessa visualização está localizada em <http://aretha.pro.br/tags/sankey.html>.

Analisando os eixos de forma separada, e seguindo da esquerda para a direita (rótulo, número de comentários, tempo de conclusão, *pull requests* e código nos comentários), a primeira inferência obtida diz a respeito do uso de rótulos no repositório de software da ferramenta utilizada. É possível notar que o peso do fluxo que parte do nó que identifica tarefas não rotuladas é muito menor do que a soma dos outros nós, o que indica que as tarefas têm associado pelo menos um dos rótulos selecionados.

Olhando para o fluxo por entre os dois primeiro eixos, consegue-se perceber que, quando uma tarefa não é rotulada, esta tende a ter um menor número de comentários, já que os maiores fluxos, que conectam o nó (no label) no primeiro eixo com o segundo eixo de número de comentários, são os fluxo de 2 a 5 comentários ou 1 comentário. Analisando o nó 5-11 comentários no segundo eixo de número de comentários, percebe-se que seus maiores fluxos vem de tarefas rotuladas, por exemplo, *bug, 3. to review*, *enhancement* e *0. Needs triage*. Esse comportamento evidencia que o uso de rótulos gera mais discussão entre os programadores em busca da solução da tarefa, o que é algo positivo.

Seguindo a análise e olhando para o fluxo que permeia o segundo e o terceiro eixo, pode-se notar que muitas tarefas com baixo número de comentários (0 a 5) fecham em até 10 dias. Esse comportamento pode denotar que estas tarefas não abordam questões muito complexas, por isso são concluídas mais rapidamente.

Correlacionando o tempo de conclusão das tarefas com características referentes a *pull requests* e presença ou não de código em seus comentários, temos o fluxo presente entre os dois últimos eixos do diagrama da Sankey. Nota-se que a maioria das tarefas possuem *pull requests* fechados, ou seja, aceitos, e estes por sua vez, partem majoritariamente do nó que representa o tempo de conclusão de menos de 10 dias. O nó que representa tarefas sem código em seus comentários, está diretamente ligado a tarefas que não foram concluídas, ou que tiveram sua conclusão em menos de 200 dias. Quando analisando o nó que representa as tarefas que possuem *pull request* aberto, mas não aceito, é perceptível que todas as tarefas que entram nessa categoria ainda não foram concluídas, mostrando que neste projeto de software a conclusão de uma tarefa depende da implementação e da submissão deste código por parte de um desenvolvedor.

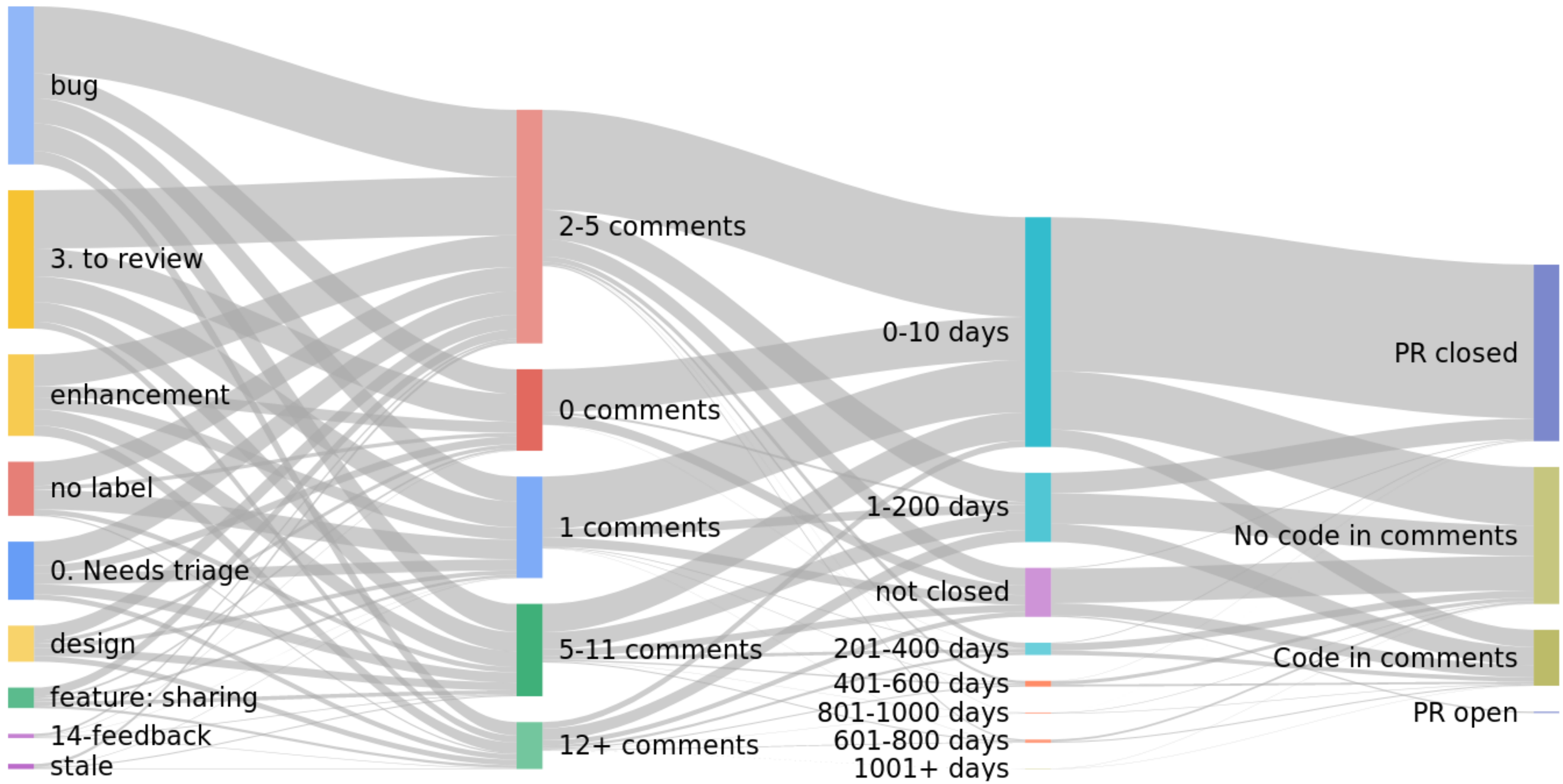


Figura 3.10. Diagrama de Sankey baseado no uso de rótulos no repositório de software NextCloud.O fluxo do diagrama é representado pelas tarefas e seus eixos por características destas, sendo primeiro alguns rótulos selecionados, o segundo e o terceiro respectivamente pelo número de comentários e tempo de fechamento das tarefas, e o quarto pela presença de *pull request* e código dos comentários das tarefas.
Fonte: Autoria própria (2019).

3.11. Considerações Finais

Todo o processo, desde a elaboração das questões de pesquisa até a obtenção dos resultados, se mostrou um bastante complexo. Sua complexidade se deu pela necessidade de conhecer o domínio, elaborar as questões de pesquisa, estudar sobre a aquisição dos dados e quais informações estes provêm. Também foi necessário analisar quais técnicas de visualização seriam interessantes utilizar, pensando em responder as questões propostas, mas levando em conta o formato dos dados obtidos, implementação dos *scripts* utilizados, e análise das visualizações obtidas. Além disso, na etapa de implementação, o processo se deu de forma bastante sequencial, mas heterogênea, pois o processo segue a ordem coleta dos dados, pré-processamento, aquisição da visualização e pós processamento, entretanto, para cada visualização os *scripts* eram diferentes de acordo com a necessidade.

Após a aplicação de diversas técnicas de Visualização de Informação, para conjuntos de dados heterogêneos e que demandaram diferentes abordagens na etapa de pré-processamento, pode-se perceber que muitas vezes a utilização de uma única técnica de Visualização de Dados pode não garantir inferências corretas e coerentes. Assim, é aconselhável utilizar mais de uma técnica para diferentes características da base de dados, e até mesmo para uma característica já analisada, a fim de se obter maior completude nos resultados.

Ao analisar questões unitárias sobre o uso de rótulos nas tarefas do repositório de software da ferramenta NextCloud, dentre estas o uso de rótulos ao longo do tempo (Seção 3.6), a quantidade de rótulos por tarefa (Seção 3.5), o tempo de conclusão das tarefas (Seção 3.7), o número de comentários em tarefas rotuladas (Seção 3.8) e a correlação entre os rótulos (Seção 3.9), foi percebida a necessidade de integrar essas informações e obter resultados mais completos. Assim, foi implementado o Diagrama de Sankey, Seção 3.10, que leva em conta todas as outras técnicas aplicadas nas tomadas de decisão sobre seus eixos e os valores de seus nós.

Capítulo 4

Conclusão e Trabalhos Futuros

Nesta seção será apresentado um resumo do estudo, focando nos resultados obtidos e os associando com as questões de pesquisa propostas. Também serão apresentados trabalhos futuros que podem contribuir com a completude deste.

O processo para a elaboração deste estudo se iniciou pela aquisição de conhecimento sobre o domínio utilizado, no caso, plataformas sociais para engenharia de software, uso de rótulos em projetos de software, mineração de repositórios de software, e Visualização de Informação. Após a conclusão da etapa de estudo, se iniciou a etapa de definição da metodologia e das questões de pesquisa, as quais consistem em:

- O projeto faz uso de rótulos?
- O tempo de vida de uma tarefa é influenciado pelo rótulo atribuído?
- O uso de rótulos melhora a comunicação entre desenvolvedores?
- Qual é o efeito global do uso de rótulos no tempo de conclusão das tarefas, na comunicação e na forma da conclusão das tarefas?

Com as questões definidas, foi iniciado o processo de escolha das técnicas de Visualização de Dados e implementação destas. As técnicas escolhidas foram *streamgraph*, *box plot*, grafo e diagrama de Sankey, e todas passaram por um processo de aquisição de dados, pré-processamento e geração da visualização.

4.1. Contribuições

Após todo o processo de estudo e desenvolvimento deste trabalho, analisando o uso de rótulos em repositórios de software livre, especificamente no repositório da ferramenta NextCloud, hospedado na plataforma social de desenvolvimento GitHub, pode-se concluir que a comunidade mantenedora do projeto se preocupa com o uso de rótulos em suas tarefas, frequentemente associando mais de um rótulo a uma mesma tarefa. Esta conclusão foi baseada nos resultados obtidos nas Seções 3.6, 3.5 e 3.10, onde foram implementadas as técnicas de Visualização de Informação *streamgraph*, *box plot* da quantidade de rótulos por tarefa e diagrama de Sankey, respectivamente.

Analisando o tempo de conclusão de uma tarefa e o correlacionando ao uso de rótulos, pode-se perceber que tarefas não rotuladas tendem a ser concluídas mais rapidamente. Talvez este fato esteja diretamente ligado a complexidade técnica desta tarefa, sendo esta tão simples que não houve a necessidade do emprego de um rótulo à mesma. Este indício está diretamente ligada ao resultado apresentado na Seção 3.7, onde foi implementado um *box plot* com o tempo de conclusão das tarefas de acordo com cada rótulos, e para tarefas não rotuladas.

Quando a contribuição que os rótulos podem ter quanto à comunicação entre desenvolvedores, foi possível perceber que tarefas rotuladas possuem maior número de comentários do que não rotuladas. Estes resultados estão contidos na Seção 3.8, onde foram gerados dois *box plots*, para tarefas rotuladas e não rotuladas, com a entrada de dados referente ao número de comentários por tarefa.

A fim de analisar a coocorrência de rótulos em tarefas em um dado repositório, ou seja, rótulos utilizados em conjunto em uma única tarefa, foi implementado uma visualização baseada em grafos. No grafo apresentado na Seção 3.9, os rótulos são representados pelos nós do grafo e suas coocorrências pelas arestas. Percebe-se que o grafo apresentado é altamente conectado, o que denota uma tendência de na comunidade do NextCloud, de usar mais de um rótulo por tarefa. Olhando para a área central do grafo, é possível perceber que alguns dos rótulos mais frequentes que possuem alta correlação, como por exemplo os rótulos *bug*, *enhancement*, *3.to review* e *1. to develop*.

Correlacionando todos os resultados obtidos através da implementação do diagrama de Sankey, explicitado na Seção 3.10, foi possível analisar características de tarefas que possuem alguns rótulos específicos associados. Estes rótulos foram escolhidos através da análise das outras visualizações implementadas, levando em conta as maiores frequências de uso, consistência de uso no decorrer do projeto, tempo de conclusão e correlação entre estes. Além disso, foram incluídas tarefas que não possuem nenhum rótulo associado. Assim, foi possível confirmar que as tarefas não rotuladas possuem um menor número de comentários, bem como tarefas com poucos comentários são concluídas mais rapidamente. Isto pode denotar que tarefas sem rótulos associados são mais simples e geralmente requerem pouca comunicação entre os desenvolvedores, assim como tarefas com poucos comentários (entre 0 e 5), pois estas tendem a serem concluídas mais rapidamente.

Analisando a forma de conclusão das tarefas, foi notado que a maioria das tarefas possuem *pull request* aceitos, e estes são providos majoritariamente de tarefas concluídas em até 10 dias. Poucas tarefas tem *pull requests* não-concluídos, o que pode denotar preocupação por parte dos mantenedores do projeto de aceitar os *pull requests* rapidamente. O último ponto analisado, a presença de código em comentários, mostrou que grande parte das tarefas não possuem esta característica.

4.2. Trabalhos Futuros

Os passos para dar continuidade a esse trabalho estão relacionados às limitações identificadas e a investigação e incorporação de algumas ideias que surgiram no decorrer na pesquisa, mas que não foram incluídas nesse trabalho. A seguir, alguns desses passos são brevemente discutidos:

- Agrupar rótulos com funções semelhantes nas visualizações. Os seguintes rótulos, por exemplo, '*feature:search*', '*feature:emails*' e '*feature:logging*' poderiam ser agrupados em um único rótulo chamado '*feature*';
- Adição de peso para os nós do grafo, a fim de representar a frequência de uso dos rótulos;
- Uso de uma biblioteca que permitisse a visualização do grafo de coocorrência de rótulos por meio de um arquivo HTML para manter o mesmo formato de arquivo para todas as visualizações e sem o uso de uma ferramenta externa;
- Incluir função para permitir reordenação dos eixos no diagrama Sankey para melhor análise; e inclusão ou remoção de características selecionadas para serem mapeadas para eixos no diagrama Sankey.
- Replicar a metodologia e implementação para outros repositórios de software;
- Ferramenta para automatizar a geração e exibição das visualizações apenas pela seleção do repositório a ser analisado.

Referências

ALENCAR, Aretha Barbosa. *Visualização da evolução temporal de coleções de artigos científicos*. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação - ICMC-USP, São Carlos-SP, Brasil, 12 2013.

BENJAMINI, Yoav. Opening the box of a boxplot. *The American Statistician*, Taylor & Francis Group, v. 42, n. 4, p. 257–262, 1988.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. (Ed.). *Readings in information visualization: using vision to think*. San Francisco, CA, EUA: Morgan Kaufmann, 1999. 712 p. ISBN 1-55860-533-9.

CHEN, C. *Information Visualization: Beyond the Horizon*. Secaucus, NJ, EUA: Springer-Verlag, 2006. ISBN 184628340X.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI Magazine*, Association for the Advancement of Artificial Intelligence (AAAI), Palo Alto, CA, EUA, v. 17, n. 3, p. 37–54, set.–dez. 1996. ISSN 0738-4602.

FEOFILOFF Y. KOHAYAKAWA, Y. Wakabayashi P. Uma introdução sucinta à teoria dos grafos. IME USP, 2011.

HASSAN, Ahmed E. The road ahead for mining software repositories. *2008 IEEE Frontiers of Software Maintenance*, IEEE, v. 1, p. 48–57, 2008.

HAVRE, Susan; HETZLER, Beth; NOWELL, Lucy. Themeriver: Visualizing theme changes over time. In: *Proceedings of the IEEE Symposium on Information Vizualization*. Washington, DC, EUA: IEEE Computer Society, 2000. p. 115–. ISBN 0-7695-0804-9.

HEMMATI, Hadi; NADI, Sarah; BAYSAL, Olga; KONONENKO, Oleksii; WANG, Wei; HOLMES, Reid; GODFREY, Michael W.; CHERITON, David R. The msr cookbook: Mining a decade of research. *2013 10th Working Conference on Mining Software Repositories (MSR)*, IEEE, v. 10, p. 343–352, 2013.

INSELBERG, Alfred. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN 0387215077.

IZQUIERDO, Javier Luis Cánovas; COSENTINO, Valerio; ROLANDI, Belén; BERGEL, Alexandre; CABOT, Jordi. Gila: Github label analyzer. *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, IEEE, v. 22, p. 479–483, 2015.

JACOMY SEBASTIEN HEYMANN, Tommaso Venturini Mathieu Bastian Mathieu. Forceatlas2, a continuous graph layoutalgorithm for handy network visualization. Universite Pierre et Marie Curie, 2012.

KALLIAMVAKOU, Eirini; SINGER, Leif; GOUSIOS, Georgios; GERMAN, Daniel M.; BLINCOE, Kelly; DAMIAN, Daniela. The promises and perils of mining github. *MSR 2014 Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR, v. 11, p. 92–101, 2014.

KIKAS, Riivo; DUMAS, Marlon; PFAHL, Dietmar. Issue dynamics in github projects. *2015 Product-Focused Software Process Improvement*, Springer, v. 9459, p. 295–310, 2015.

LIAO, Zhifang; HE, Dayu; CHEN, Zhijie; FAN, Xiaoping; ZHANG, Yan; LIU, Shengzong. Exploring the characteristics of issue-related behaviors in GitHub using visualization techniques. *IEEE Access*, IEEE, v. 6, p. 24003–24015, 2018.

NAYEBI, Maleknaz; KABEER, Shaikh Jeeshan; RUHE, Guenther; CARLSON, Chris; CHEW, Francis. Hybrid labels are the new measure. *2018 35nd IEEE Software*, IEEE, v. 35, p. 54–57, 2017.

OLIVEIRA, M. C. F.; LEVKOWITZ, H. From visual data exploration to visual data mining: a survey. *IEEE Transactions on Visualization and Computer Graphics*, IEEE Computer Society, Los Alamitos, CA, EUA, v. 9, n. 3, p. 378–394, jul.–set. 2003. ISSN 1077-2626.

RIEHMANN, Patrick; HANFLER, Manfred; FROEHLICH, Bernd. Interactive sankey diagrams. *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, p. 233–240, 2005.

SPENCE, R. *Information Visualization:Design for Interaction.* 2 ed.. ed. Harlow, Inglaterra: Prentice Hall, 2007. 304 p. ISBN 978-0132065504.

THOMAS, J. J.; COOK, K. A. A visual analytics agenda. *IEEE Computer Graphics and Applications*, IEEE Computer Society Press, Los Alamitos, CA, EUA, v. 26, n. 1, p. 10–13, jan. 2006. ISSN 0272-1716.

TREUDE, Margaret-Anne Storey Christoph. Work item tagging: Communicating concerns in collaborative software development. *2012 IEEE 38nd Transactions on Software Engineering*, IEEE, v. 38, p. 19–38, 2012.

ZHOU, Hong; YUAN, Xiaoru; QU, Huamin; CUI, Weiwei; CHEN, Baoquan. Visual clustering in parallel coordinates. *Computer Graphics Forum*, v. 27, p. 1047–1054, 05 2008.