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

CLÁUDIA LÁZARA POIET SAMPEDRO

VISUALIZANDO O USO DE LABELS EM REPOSITÓRIOS DE SOFTWARE NO GITHUB

MONOGRAFIA

CAMPO MOURÃO
2019

CLÁUDIA LÁZARA POIET SAMPEDRO

VISUALIZANDO O USO DE LABELS EM REPOSITÓRIOS DE SOFTWARE NO GITHUB

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

Orientador: Profª. Drª. Aretha Barbosa Alencar

Coorientador: Prof. Dr. Marco Aurélio Graciotto Silva

CAMPO MOURÃO

2019

Resumo

Poiet Sampedro, Cláudia Lázara. Visualizando o uso de labels em repositórios de software no GitHub. 2019. 30. 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, pois esta pode transformar dados coletados de repositórios de software em informações úteis para tomada de decisões dentro do gerenciamento de projetos de software. Relacionando estas duas áreas, é possível analisar dados obtidos através da coleta e mineração de repositórios de software, e então, aplicar a estas técnicas de visualização de informações, a fim de detectar padrões previamente desconhecidos presentes nos repositórios de software. Neste estudo analisaremos os rótulos (*labels*) presentes em diversas plataformas sociais de desenvolvimento. Há várias questões permeando o uso de *labels* em repositórios de software, as quais podem ser exemplificadas por organização de repositórios, comunicação entre desenvolvedores e redução do tempo de vida de uma tarefa.
Objetivo: Este estudo busca analisar a aplicação dos rótulos em repositórios de software, em especial, se a utilização destes pode gerar melhorias no gerenciamento de repositórios de software. Também é desejado investigar as possíveis melhoras que a utilização de rótulos pode trazer para a comunicação entre o time de desenvolvimento, analisando questões quantitativas e qualitativas que permeiam a troca de mensagens entre os desenvolvedores. Outro ponto a ser explorado é a frequência de tarefas concluídas, que possuam alterações de código, as quais possuem *labels* associadas.
Método: A metodologia empregada para a aquisição de respostas para estas perguntas, pode ser dividida em três grandes etapas: coleta e pré-processamento de dados, a qual consiste no uso da API REST da plataforma GitHub para coletar os dados referentes as *labels* empregadas as tarefas de certo repositório, e no pré-processamento para adequar os dados coletados ao formato esperado; extração e visualização de padrões, etapa a qual é responsável pela geração da visualização com os dados pré-processados; e a etapa de pós-processamento, que pode reiniciar o ciclo já empregado em busca de resultados melhores.

Resultados esperados: Geração de visualizações de informação que facilitem as inferências referentes às questões relacionadas ao uso de *labels* em repositórios de software hospedados na plataforma GitHub, em especial, qual a contribuição das *labels* na comunicação dos desenvolvedores, no gerenciamento do projeto, e quais destas influenciam positivamente no contexto geral de engenharia de software.

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

Lista de figuras

Lista de acrônimos

MSR Mineração de Repositórios de Software 11

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 etapa de análise dos dados pode ser complexa, situação a qual pode gerar perda de informações úteis se não feita com cuidado, tornando a etapa de coleta dos dados desnecessária. Para contornar esta questão, surgiram as técnicas de visualização de dados, buscando através da criação de modelos gráficos e representações visuais, ampliar a capacidade cognitiva humana em processo de exploração de dados (ALENCAR, 2013).

Em paralelo 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 que possam ser analisado 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, a qual é utilizada para marcar elementos referentes a uma tarefa, como erros, a fim de facilitar a identificação destes (TREUDE, 2012). Além disso, os rótulos, normalmente chamados de *labels*, podem ser utilizados como uma maneira informal do time de desenvolvimento tomar notas sobre tarefas durante o processo de desenvolvimento.

Correlacionando estas áreas, será possível gerar visualizações de informação sobre dados coletados dos rótulos de repositórios de software hospedados nas plataformas sociais de desenvolvimento colaborativo GitHub. Deseja-se descobrir se o uso geral de rótulos ou de rótulos específicos contribuem com o gerenciamento de projetos de software, bem como quais os rótulos que influenciam positivamente no contexto geral de engenharia de software. Questões relacionadas

a possível melhora que o uso de rótulos pode trazer à comunicação entre desenvolvedores, analisando questões quantitativas e qualitativas, como a quantidade de mensagens trocadas até a conclusão da tarefa rotulada, e a qualidade dessas mensagens, também estão sendo consideradas neste estudo. Outro foco a ser explorado, é a frequência da atribuição de rótulos para tarefas concluídas que possuam alterações de código.

Serão encontrados no decorrer do texto outros dois capítulos. No Capítulo 2 está contida toda a revisão bibliográfica e embasamentos teórico necessários para a elaboração do estudo. No Capítulo 3, apresenta-se a proposta e método empregados durante o desenvolvimento do estudo, bem como os resultados preliminares obtidos e o cronograma para as atividades futuras.

Capítulo 2

Referencial Teórico

Este capítulo possui a finalidade de fundamentar princípios e técnicas que foram utilizadas para a elaboração desta proposta. 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 rótulos de projetos de software livre.

Neste capítulo, na Seção 2.1 é abordado o uso de plataformas sociais para gerenciamento de projetos, como o GitHub. A Seção 2.2 relata o contexto do uso de marcações ou rótulos em projetos de software. Já a Seção 2.3 discute o contexto da área de mineração de repositórios de software, a qual 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. Por fim, a Seção 2.4 apresenta a área de pesquisa de Visualização de Dados e 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, etc. Estas costumam utilizar a ferramenta *git*[1] para realizar o gerenciamento e controle de versão, e o registro e controle 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.

Os repositórios de software por sua vez consistem basicamente em um conjunto de diretórios e arquivos, onde os arquivos estão dispostos, dispondo de funções extras, como criar

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

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 qualquer *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 ***issues***, as quais simbolizam relatos de erros ou tarefas a serem realizadas dentro do projeto, como novas funcionalidades, etc. Nessas podem ser adicionadas ***labels***, com a finalidade de rotular o conteúdo de uma *issue*, facilitando a compreensão e gerenciamento da tarefa a ser realizada. Um colaborador do projeto geralmente soluciona uma dada *issue* 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.

2.1.1. GitHub

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 *issues*, dentre outras funcionalidades. Sua popularidade aumentou nos últimos anos graças a 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 (*issues*) 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 *issues* possuem um sistema de categorização, as quais as rotulam como abertas ou fechadas. *Issues* 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 *issues* podem transitar por entre estas categorias livremente (KIKAS et al., 2015).

Além do texto presente na descrição de uma *issue*, o GitHub dispõe de um sistema de rotulação, a qual permite a adição de rótulos, populares *labels*, para classificar a *issue* 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 *software*, como GitHub, GitLab, Jazz, entre outros, utilizam algum sistema para rotular as atividades a serem feitas, empregando características relevantes a estas. Estes rótulos, popularmente conhecidos por *labels*, 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).

Labels estão sendo utilizadas a 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, estas 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 teoricamente 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).

Dentro da mineração de software há um ciclo que começa na coleta de dados, até a análise final e obtenção de inferências. O ciclo todo 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 e aplicações em MSR

Com a evolução da computação tornou-se possível obter e manipular montantes de dados volumosos, fenômeno a qual 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 busca 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 disciplinas e reforçando a interação humana no processo analítico.

Existem inúmeras técnicas de Visualização de Dados, onde cada uma se mostra eficiente para diferentes tipos de dados (temporal, multidimensional, hierárquico, textual, geoespacial, etc.) e/ou diferentes padrões que se busca descobrir ou confirmar. Com o objetivo de entender melhor o uso de técnicas de visualização de dados para repositórios de *software* do GitHub, em específico

labels atribuídas à *issues*, 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 das *labels* utilizadas nas *issues* do projeto de maneira diversa.

De maneira geral, a primeira visualização, ilustrada na Figura 2.1, tem o objetivo de identificar quais as *labels* mais usadas e quais são mais comumente utilizadas em conjunto. Essa visualização é ilustrada na Figura 2.1, que mostra um grafo de coocorrência de *labels*. Os vértices são as *labels*, e as arestas possuem pesos que indicam quantas vezes duas *labels* foram usadas em conjunto em diferentes *issues*. Esse peso é representado visualmente pela largura da aresta.

Figura 2.1. Primeira visualização proposta pelo artigo GiLA, mostrando um grafo de coocorrência entre *labels*.

A segunda visualização tem o objetivo de relacionar os usuários, que colaboraram abrindo, fechando e comentando *issues* que utilizam de uma dada *label*. A Figura 2.2 exemplifica uma aplicação desta. O vértice circular central representa uma *label*, escolhida pelo usuário, e os vértices quadrados em volta, usuários que contribuíram com *issues* que utilizavam desta. A espessura das arestas demonstra o número de comentários que o usuário fez em *issues* que utilizam desta *label*. Já a largura e altura do vértice do usuário é proporcional ao número de *issues* criadas e fechadas, respectivamente, com esta *label*. 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 uma *label*, não sendo tão interessante para análises mais completas e que envolvam mais de uma *label* de um mesmo repositório. Colocar mais de uma *label* 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).

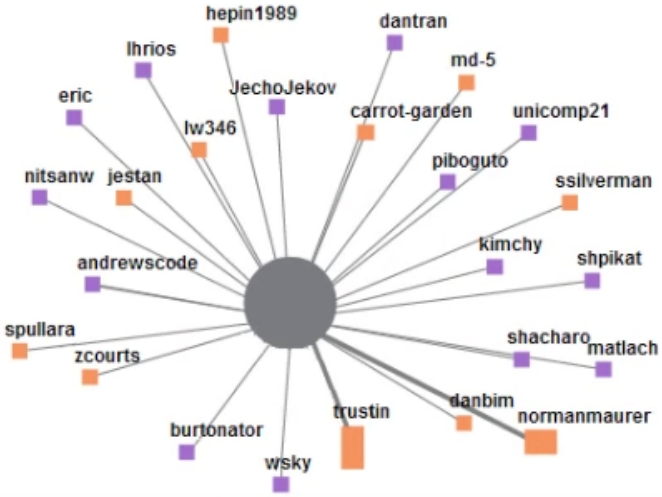


Figura 2.2. Segunda visualização proposta pelo artigo GiLA, relacionando a participação de usuários em *issues* com uma dada *label*.

A terceira e última visualização, demonstrada pela Figura 2.3, tem o objetivo de relatar o ciclo de vida das *issues* de determinada *label* por meio de uma visualização baseada em árvore. Os autores determinaram dois marcos importantes no ramo central: tempo médio para primeiro comentário nas *issues* que contém tal *label* e tempo médio para o primeiro comentário de um colaborador do projeto. O ramo central se ramifica em três ramificações: a porcentagem de *issues* 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 uma *label* no repositório por vez.

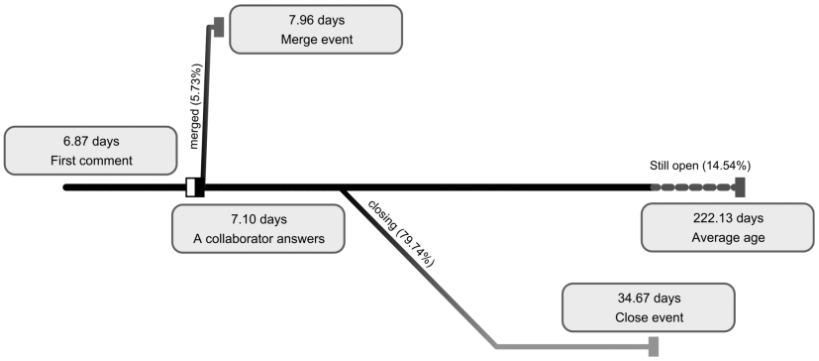


Figura 2.3. Terceira visualização proposta pelo artigo GiLA, demonstrando o ciclo de vida das *issues* associadas a uma determinada *label*

O artigo possui conteúdo bastante interessante quando tratado o uso de *labels* 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 [2], a qual pega os dados do GitHub e transforma em um banco de dados SQL. Esta pode tender ser de fácil utilização, mas apresenta inconsistência nos dados, podendo apresentá-los de forma obsoleta.

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 *issues* são importantes? *Issues* são importantes para projetos no GitHub?; (2) É efetivo aplicar *labels* a *issues* para o gerenciamento de *issues*?; 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 2.4 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 *issue* 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 2.4 ainda ilustra que a contribuição dos comportamentos de usuários geralmente tem alto grau de importância nestes sete projetos. As importâncias do *IssueCommentEvent* e *PullRequestEvent* são relativamente altos entre todos os repositórios selecionados.

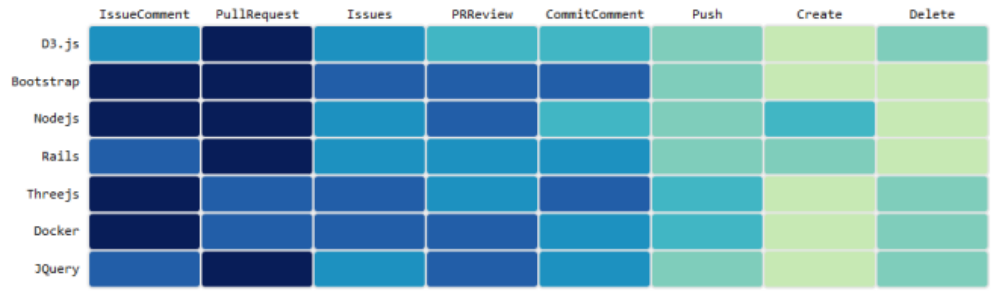


Figura 2.4. *HeatMap* da importância de eventos de comportamento de usuários em sete projetos populares.

[2] http://ghtorrent.org/

Ainda sobre a primeira questão, a Figura 2.5 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 com a quantidade de dados que este representa. Na Figura 2.5, os retângulos de dados representam comportamentos de usuários, e estão agrupados de acordo com suas semelhanças, formando grupos relacionados a eventos de *pull requests*, *issues*, *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 *issues* na *treemap* e o evento de *PullRequestEvent* é o de maior importância no grupo que é relacionado a *Pull Request*.

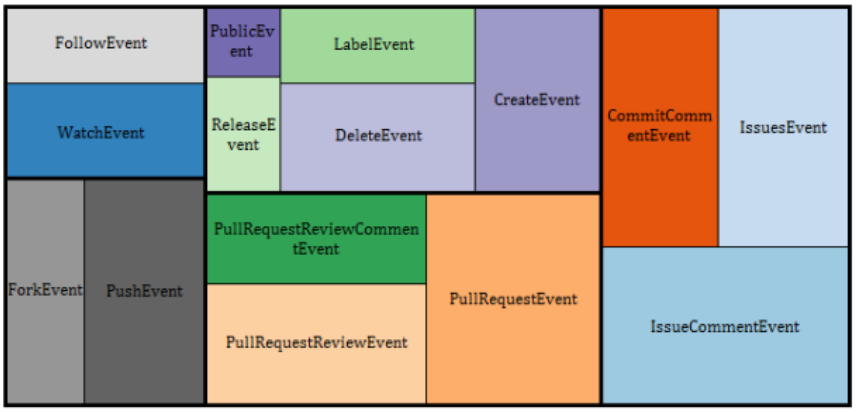


Figura 2.5. Agrupamentos de eventos baseados nas características e importância dos comportamentos. A área dos retângulos representa quão importante é tal comportamento.

A segunda pergunta proposta, "É efetivo aplicar *labels* a *issues* para o gerenciamento de *issues*?", usa uma visualização do tipo radar em sua resposta, representada pela Figura 2.6. 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, representando uma dada instância, intersecta esses eixos no valor correspondente.

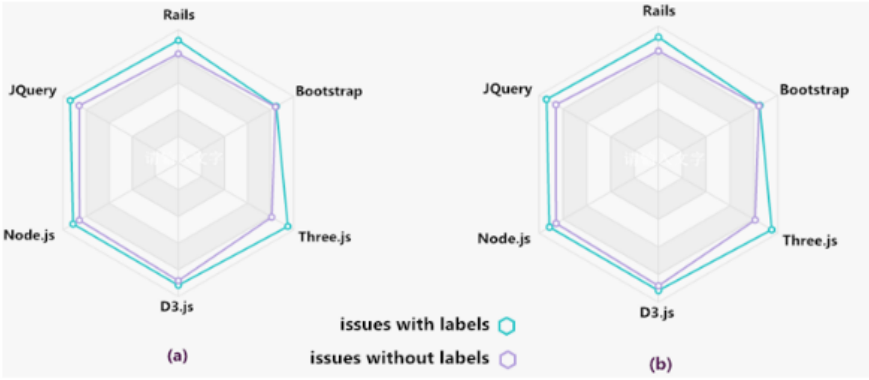


Figura 2.6. Representação em forma de radar para analisar a influência das *labels* no tempo de fechamento de uma *issue*. Os gráficos (a) e (b) consideram o intervalo de tempo de 3 e 365 dias, respectivamente..

No contexto do artigo, esta consiste em coletar *issues* com e sem *labels* para os projetos relacionados, e relacionar o tempo que estas levaram para serem fechadas em cada um deste projeto. A polilinha em azul representa *issues* com *labels* e a polilinha roxa *issues* sem *labels*. Quanto mais próxima da borda a polilinha está, maior o número de *issues* fechadas. O primeiro gráfico de radar considera o intervalo de tempo de 3 dias e o segundo gráfico o intervalo de tempo de 365 dias. Apesar dos intervalos de tempo diferentes, os resultados foram basicamente iguais, concluindo que *issues* com *labels* 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.

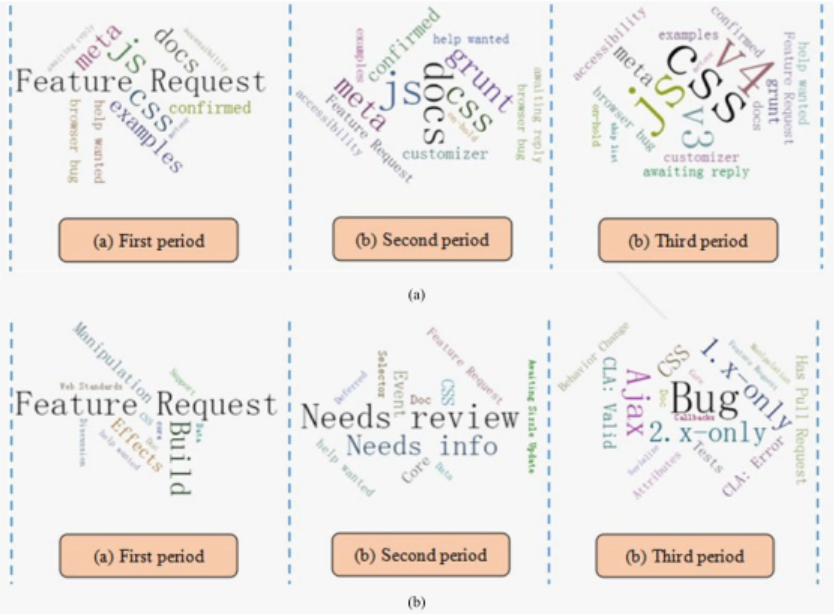


Figura 2.7. Representação da frequência de palavras chaves de *commits* nos repositórios *Three.js* e *JQuery*, agrupadas por períodos de tempo.

A Figura 2.7 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 as *labels* 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 as *labels* mais presentes são: *css*, voltado para *design*; e *js*, relacionada a desenvolvimento do segundo projeto. Os autores inferiram então, que as palavras que mais aparecem em cada etapa estão relacionas com as etapas de desenvolvimento de um software, afirmação a qual 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 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.

2.5. 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 *labels* em repositórios de software.

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

Capítulo 3

Método

3.1. Metas e Questões de Pesquisa

Existem várias questões que permeiam o uso de *labels* 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 a serem respondidas com este estudo são: o uso geral de *labels* ou de *labels* específicas contribuem com o gerenciamento de projetos? E quais *labels* influenciam positivamente no contexto geral de engenharia de software? Assim, técnicas de visualização de dados serão 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.

Outra questão é se o uso de *labels* realmente melhora a comunicação entre desenvolvedores. Essas análises referentes à comunicação entre desenvolvedores leva em conta questões quantitativas e qualitativas. As quantitativas estão atreladas ao número de mensagens trocadas até a resolução do problema, e as qualitativas ao conteúdo destas mensagens, podendo este ser investigado através do uso de mineração de texto e da análise de elementos do texto, como por exemplo, a presença de códigos. Pode-se observar que os temas citados estão contidos no escopo de gerenciamento das tarefas, o qual, consiste na análise e organização das tarefas pendentes dentro de um repositório de software.

Dentre as questões em aberto, pode-se elencar também qual a frequência da atribuição de *labels* para tarefas concluídas que contém alterações de código. Esta questão explora o quanto *labels* contribuem para que tarefas sejam concluídas em um período menor de tempo com contribuição efetiva.

Os primeiros resultados obtidos levaram em conta questões quantitativas relacionadas à frequência do uso de *labels* durante o período de vida dos projetos analisados, através da aplicação

da técnica de visualização de dados *streamgraph*, a qual é eficiente para evidenciar o aspecto temporal em base de dados, mostrando variações nas frequências dos dados ao longo do tempo.

Outras técnicas de visualização de dados serão aplicadas, a fim de fornecer visões complementares que destaquem outros padrões desconhecidos, buscando respostas para as questões propostas. Dado que o maior ganho no uso de técnicas de visualização se dá na utilização de diversas técnicas para a mesma base de dados para capturar diferentes aspectos da base e correlacioná-los.

Para que estas perguntas sejam respondidas, é 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. É desejado que este processo possa ser aplicado para qualquer repositório de software, gerando resultados consistentes e satisfatórios.

3.2. Método de Pesquisa – Mineração Visual de Dados

Nessa seção, serão apresentadas as etapas necessárias para um processo de mineração visual de dados aplicado ao contexto de repositórios de software. Esse processo foi dividido em três grandes etapas: 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.

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 (mineração de repositórios, 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 dependem diretamente do formato dos dados obtidos e do formato aceito pela técnica de visualização, geralmente gerando um arquivo final nesta etapa, que será utilizado na próxima etapa.

Para coletar os dados necessários para gerar visualizações de dados e então fazer a análise destas, a API REST V3 do GitHub[1] foi utilizada. Esta dispõe de diversos métodos para recuperar e coletar dados relacionados aos repositórios, usuários e *issues*, além de ser gratuita. A API REST V3 do GitHub foi escolhida pois essa apresenta consistência desejada nos dados obtidos, além da disponibilização destes em tempo real. Entretanto há uma limitação quanto ao número de requisições que se é possível realizar por hora, sendo este de 5 mil.

O *script* utilizado para realizar as requisições foi feito na linguagem de programação Python, com o auxílio do módulo de nome PyGitHub[2], o qual abstrai parte da complexidade da API do GitHub. A escolha da API REST do GitHub se deu pela consistência que os dados fornecem, tendo em vista que estes são atualizados em tempo real.

Além das etapas de pré-processamento previamente citadas, é possível que outras técnicas de pré-processamento se tornem necessárias durante a execução da proposta e possam ser empregadas, a fim de aprimorar os resultados, como, por exemplo, a implementação de *scripts* que possibilitem o agrupamento de *labels* de acordo com sua proximidade semântica.

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. Provavelmente, cada uma das metas (questões de pesquisa) apresentadas anteriormente irá gerar uma visualização distinta para o mesmo repositório de software selecionado. Existem diversas técnicas tanto no contexto de mineração de dados quanto no contexto de visualização de dados, onde cada uma destas possui uma função distinta que seve ser considerada durante a escolha de uma das técnicas de ambas as áreas.

Se tratando 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. Por exemplo, nos resultados preliminares, que serão apresentados na Seção 3.3, foi utilizada uma técnica cujo foco está no aspecto temporal dos dados.

[1] https://developer.github.com/v3/
[2] https://github.com/PyGithub/PyGithub

Também será 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.

3.3. Resultados Preliminares

Nessa seção serão apresentados alguns resultados preliminares obtidos para visualizar a frequência do uso de *labels* ao longo do tempo em dois diferentes repositórios de software, Shiny[3] e NextCloud[4]. O Shiny é um pacote para a linguagem de programação R, que facilita a construção de aplicativos interativas *web*. Já o NextCloud consiste em uma plataforma *open source* de compartilhamento de arquivos e colaboração auto-hospedada na *web*.

Inicialmente o repositório escolhido foi o Shiny, o qual não conta com grandes quantidades de dados quanto ao uso de *labels*. Dessa forma, também foi selecionado o repositório da plataforma NextCloud pois este apresenta uma maior quantidade de *labels* atribuídas.

Durante o processo de coleta dos dados, foi necessário principalmente contornar o limite de requisições por hora que a API REST da plataforma GitHub oferece, e realizar alguns ajustes para que o algoritmo de coleta não apresentasse falhas de consistência e integridade durante a execução.

Nesta análise inicial dos dados será usada a técnica de visualização *streamgraph*, que 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

[3] <https://github.com/rstudio/shiny>
[4] <https://github.com/nextcloud/server>

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. Por exemplo, a Figura 3.1 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 os anos de 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 certos temas.

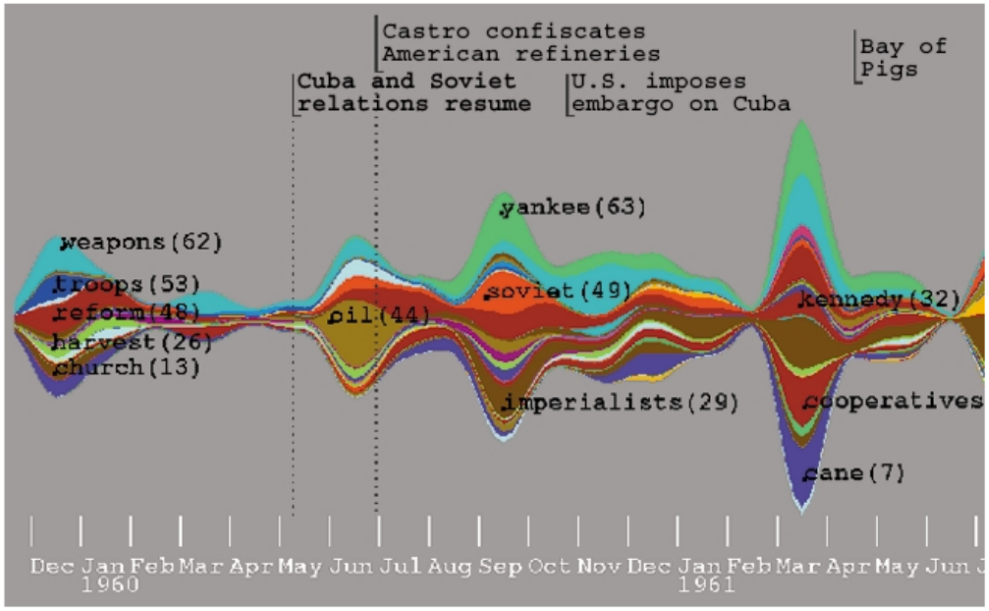


Figura 3.1. *ThemeRiver* usando dados de notícias jornalísticas. Esta mostra as palavras-chaves mais encontradas em notícias no período da crise cubana-americana ocorrida em 1960.
Fonte: Havre et al. (2000).

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 um pouco a um fluxo semelhante a 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. Para a geração da visualização de dados foi utilizada a linguagem de programação R em conjunto com a biblioteca *streamgraph*.

No contexto dessa proposta, a técnica *streamgraph* foi usada para visualizar a variação temporal no uso de *labels* nos repositórios Shiny e NextCloud no GitHub. Para coletar os dados relacionados às *labels* (nome e frequência destas no decorrer dos meses durante os anos de vida do repositório) foi necessário coletar todas as *issues* do repositório, e então armazenar o resultado

da busca em um arquivo JSON, mas que ainda é um formato incompatível com o necessário para a geração da visualização *streamgraph* na linguagem de programação R.

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

Os dados ainda foram submetidos a um terceiro *script*, o qual modela os dados do segundo arquivo JSON (nome da *label*, 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 então utilizados para alimentar a biblioteca. As Figuras 3.2 e 3.3 mostram os resultados obtidos após a geração da *streamgraph*, utilizando como dados as *labels* presentes no decorrer do tempo nos repositórios das ferramentas Shiny e NextCloud respectivamente. Os códigos utilizados para a obtenção destes resultados estão hospedados na plataforma GitHub[5], e a versão dinâmica das visualizações em um servidor de arquivos[6].

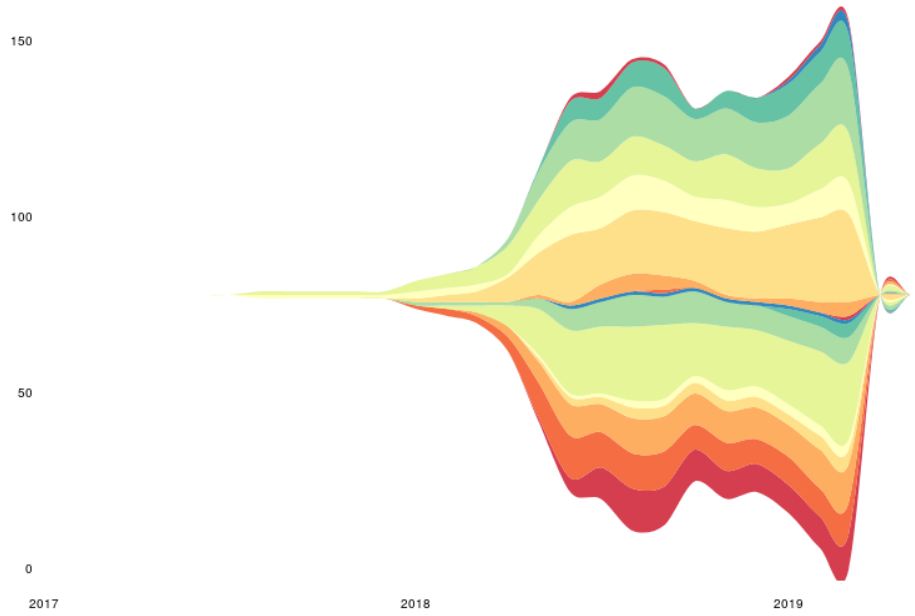


Figura 3.2. *Streamgraph* gerada para os dados das *labels* coletados do repositório Shiny.

Observando as Figuras 3.2 e 3.3 é possível notar uma divergência quanto ao uso de *labels* nos dois repositórios, que parece ser mais consolidada no repositório NextCloud.

No resultado para o repositório Shiny, é possível perceber que o uso de *labels* teve relevância quantitativa a partir de 2018, apesar do repositório ter sido lançado consideravelmente antes. As *labels* coletadas para esse repositório foram: *'difficulty: advanced', 'difficulty: intermediate', 'difficulty: novice', 'docs', 'effort: high', 'effort: low', 'effort: medium', 'feature: bookmarking', 'help wanted', 'invalid', need info', 'priority: high', 'priority: low', 'priority: medium', 'type: bug', 'type:*

[5] <https://github.com/claudiaps/TCC-1>
[6] <http://aretha.pro.br/tags>

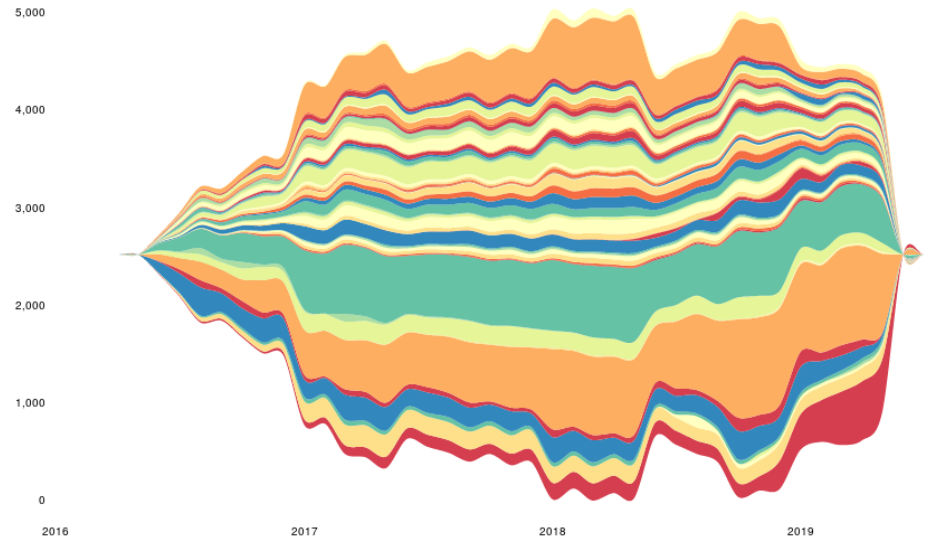
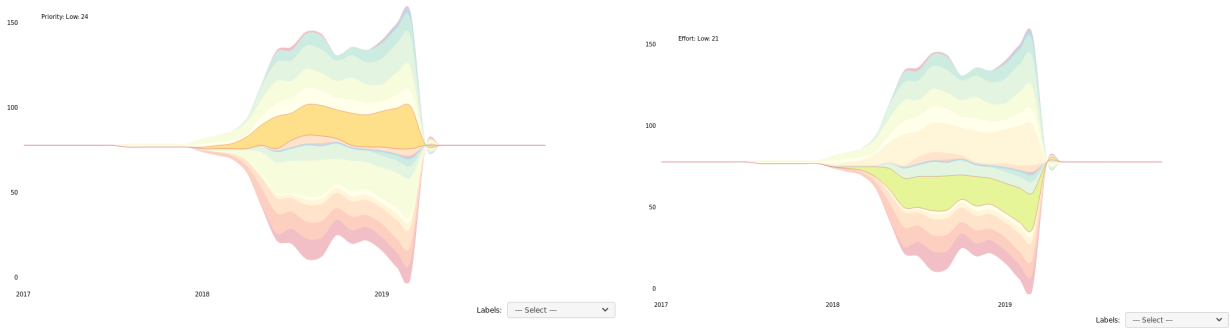


Figura 3.3. *Streamgraph* gerada para os dados das *labels* coletados do repositório NextCloud.

enhancement', 'type: question', 'type: regression', 'type: shiny server'. As *labels* mais presentes foram '*priority: low*', e '*effort: low*', com frequência máxima de 25 aplicações por mês. Estas estão destacadas pela Figura 3.4, onde a imagem *a* representa a *label 'priority: low'* e a imagem *b* a *label 'effort: low*'. Os dados foram coletados para análise em meados de abril de 2019, onde neste ano é notável um aumento considerável no número de *labels* aplicadas a tarefas neste repositório.



(a) *Label* '*prioritty: low*' destacada **(b)** *Label* '*effort: low*' destacada

Figura 3.4. Faixas que representam as *labels* mais presentes do repositório Shiny em destaque.

Quando analisados os resultados obtidos minerando os dados das *labels* do repositório NextCloud, é possível observar que o uso das *labels* é quantitativamente representativo desde o final de 2016. A *label 'bug'*, é mais frequente ao longo do tempo com aproximadamente 1000 aplicações desta *label* à *issues* por mês. A segunda *label* mais frequente deste repositório é a *label 'enhancement'*, com frequência máxima de 755, no primeiro semestre de 2018. A Figura 3.5 aponta com destaque as faixas da *streamgraph* que representam as frequências destas *labels*, onde a imagem *a* representa a *label 'enhancement'* e a imagem *b* a *label 'bug'*. A quantidade total de *labels* é bem maior que do outro repositório analisado, somando a quantidade de 77 *labels*, contra

19 presentes no repositório Shiny, onde as 5 com maior frequência total são: *'bug'*, *'enhancement'*, *'stale'*, *'to review'* e '*feture: sharing*'. A *label 'needs triage'* teve aumento considerável nos últimos meses, mas sua distribuição não é uniforme no decorrer do tempo de vida do projeto.

(a) *Label* '*enhancement*' destacada **(b)** *Label* '*bug*' destacada

Figura 3.5. Faixas que representam as *labels* mais presentes do repositório NextCloud em destaque.

Dentre as possibilidades para prover melhorias futuras a esses resultados preliminares, é relevante a aplicação de um algoritmo que agrupe as *labels* com funções parecidas, analisando a frequência destas em conjunto.

3.4. Cronograma de Atividades

Esta proposta consiste no uso de técnicas de visualização de informação para extrair padrões sobre o uso de *labels* em repositórios do GitHub aplicadas à tarefas, além de obter respostas para algumas perguntas, como se uso geral de *labels* ou de *labels* específicas contribuem com o gerenciamento de projetos, e se sim, quais *labels* influenciam positivamente no contexto geral de engenharia de software.

3.4.1. Lista de Atividades

As atividades que serão cumpridas no decorrer desta proposta são descritas a seguir:

1. Revisão Bibliográfica;
2. Coleta e pré-processamento dos dados por meio de *scripts*;
3. Seleção das técnicas de visualização de informação a serem utilizadas;
4. Estudo sobre linguagens de programação e bibliotecas a serem utilizadas;
5. Implementação de algoritmos para a aplicação das técnicas de visualização de informação selecionadas;
6. Análise das visualizações geradas;
7. Seleção dos repositórios a serem minerados;
8. Utilização de estudo de caso para análise dos resultados obtidos;
9. Escrita da monografia;

10. Apresentação à banca.

3.4.2. Cronograma

Tabela 3.1. Cronograma de atividades para o segundo semestre de 2019.

Atividade	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro
1	✓	✓	✓	✓	✓	✓	
2	✓	✓	✓	✓			
3			✓	✓	✓		
4			✓	✓	✓		
5			✓	✓	✓		
6			✓	✓	✓	✓	
7			✓	✓	✓		
8			✓	✓	✓	✓	
9			✓	✓	✓	✓	
10							✓

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.

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.

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.

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.

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.

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.