UNIFACS UNIVERSIDADE SALVADOR
MESTRADO EM SISTEMAS E COMPUTAÇÃO

ANTONIO CÉSAR BRANDÃO GOMES DA SILVA

UMA CARACTERIZAÇÃO DE PROJETOS DE SOFTWARE OPEN SOURCE DO PONTO DE VISTA DE ATRIBUTOS DE QUALIDADE DE PRODUTO

Salvador
2017

ANTONIO CÉSAR BRANDÃO GOMES DA SILVA

UMA CARACTERIZAÇÃO DE PROJETOS DE SOFTWARE OPEN SOURCE DO PONTO DE VISTA DE ATRIBUTOS DE QUALIDADE DE PRODUTO

Dissertação apresentada ao curso de Mestrado em Sistemas e Computação da UNIFACS Universidade Salvador, Laureate International Universities, como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Glauco de Figueiredo Carneiro

Salvador
2017

Ficha Catalográfica elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities

Silva, Antonio César Brandão Gomes da

Uma caracterização de projetos de software open source do ponto de vista de atributos de qualidade de produto - Antonio César Brandão Gomes da Silva. Salvador, 2017.

94 f.: il.

Dissertaçãoapresentada ao Curso de Mestrado em Sistemas e Computação, Universidade Salvador – UNIFACS, como requisito parcial paraobtençãodo grau de Mestre.

Orientador Prof. Dr. Glauco de Figueiredo Carneiro.

1. Desenvolvimento de software. 2. Planejamento de Release 3. Projetos Open Source. 4. Atributos de Qualidade de Produto. I. Carneiro, Glauco de Figueiredo, orient. II. Título.

CDD: 004.21

TERMO DE APROVAÇÃO

ANTONIO CÉSAR BRANDÃO GOMES DA SILVA

UMA CARACTERIZAÇÃO DE PROJETOS DE SOFTWARE OPEN SOURCE DO PONTO DE VISTA DE ATRIBUTOS DE QUALIDADE DE PRODUTO

Dissertação apresentada ao Mestrado em Administração da UNIFACS Universidade Salvador, Laureate International Universities, como requisito à obtenção de grau de Mestre, à seguinte banca examinadora:

Glauco de Figueiredo Carneiro – Orientador _____
Doutor em Ciência da Computação - Ufba - Unifacs
UNIFACS Universidade Salvador, Laureate International Universities

Éldman de Oliveira Nunes_____
Doutor em Computação pela Universidade Federal Fluminense
UNIFACS Universidade Salvador, Laureate International Universities

Eduardo Magno Lages Figueiredo _____
Doutor em Engenharia de Software pela Lancaster University, Inglaterra
Universidade Federal de Minas Gerais - UFMG

Salvador, 10 de fevereiro de 2017.

AGRADECIMENTOS

Algumas pessoas marcam a nossa vida para sempre, umas porque nos vão ajudando na construção, outras porque nos apresentam projetos de sonho e outras ainda porque nos desafiam a construí-los. Quando damos conta, já é tarde para lhes agradecer. Gostaria de agradecer em especial a essas pessoas:

Deus, ter fé é acreditar no invisível, no que não se pode ver, não se pode tocar, mas pode sentir concretamente. É na certeza desse teu amor infinito por nós, que primordialmente ofereço esta vitória.

Aos meus Pais, eternamente presentes. São os nossos alicerces, fontes intermináveis de incentivo, zelo, conforto, segurança, amor sem limites. Mestres da nossa dignidade, sonhadores dos nossos sonhos, cúmplices das nossas lágrimas. Qualquer palavra seria insuficiente para traduzir a gratidão por tudo.

À minha esposa Priscila, e minhas filhas, Amanda e Mariana, pelo amor incondicional, companheirismo e paciência. Vocês são meu suporte, meu alicerce e a minha vida.

Ao meu orientador, professor Doutor Glauco de Figueiredo Carneio, pela amizade, compreensão, apoio, dedicação e orientação que permitiu a busca de conhecimentos. Agradeço aos professores Miguel Monteiro da Universidade Nova de Lisboa (UNL) e Fernando Brito e Abreu do Instituto Universitário de Lisboa (ISCTE-IUL) pela atenção dispensada com valiosas contribuições ao longo deste trabalho. E por fim ao professor Eduardo Magno Lages Figueiredo e sua orientanda do doutorado Kattiana Constatino, pelo apoio e esforço na condução das atividades executadas neste trabalho.

“Talvez não tenha conseguido fazer o melhor, mas lutei
para que o melhor fosse feito. Não sou o que deveria ser,
mas Graças a Deus, não sou o que era antes"

Martin Luther King.

RESUMO

Contexto: Vários projetos Open Source Software (OSS) adotaram releases frequentes como uma estratégia para oferecer novos recursos e corrigir bugs. Essa necessidade de acelerar a entrega de software, fornecer as releases no mercado mais rápido e gerenciar os comentários frequentes dos desenvolvedores/usuários da comunidade impulsionou a execução de mudanças relevantes nas práticas de desenvolvimento de software. Dessa forma, cada projeto OSS tem suas próprias prioridades estabelecidas por suas respectivas comunidades. **Objetivo:** Identificar as principais características das iniciativas de releases frequentes de software em projetos OSS, as motivações por trás de sua adoção, estratégias aplicadas, bem como as vantagens e dificuldades encontradas. Além disso, buscar evidências que esclareçam as relações entre atributos de qualidades, prioridades atribuídas às demandas registradas e as formas pelas quais são entregues por releases de produtos. **Métodos:** Na primeira parte deste trabalho foi realizada uma Revisão Sistemática da Literatura (RSL) para alcançar um dos objetivos. Na segunda parte um estudo exploratório foi planejado, para que um conjunto de participantes identificasse atributos de qualidade através da análise de dados de repositórios de três conhecidos projetos OSS: Libre Office, Eclipse e Mozilla Firefox. **Resultados:** Na RSL foram incluídas 30 publicações de janeiro de 2006 a julho de 2016 e revela nove vantagens que caracterizam as abordagens de release de software em projetos OSS; Quatro desafios; Três possibilidades de implementação e duas motivações principais para a adoção das releases frequentes; E finalmente quatro estratégias principais para implementá-lo. Em relação ao estudo exploratório foi possível constatar que as evidências fornecidas pelos participantes sugerem que os desenvolvedores de comunidade OSS usam critérios/prioridades impulsionados por atributos de qualidade de produto de software específicos para planejar e executar releases de software. **Conclusão:** Este trabalho fornece uma compreensão atualizada e estruturada das abordagens de releases frequentes de software no contexto OSS, baseados em resultados obtidos sistematicamente a partir de uma lista de referências relevantes na última década. E por fim, uma indicação que a priorização de atributos de qualidade em projetos tende a considerar o perfil de cada projeto.

Palavras-chave: Revisão Sistemática da Literatura. Estudo Exploratório. Releases Frequentes. Projeto Open Source. Atributos de Qualidade.

ABSTRACT

Context: Several Open Source Software (OSS) projects have adopted frequent releases as a strategy to deliver both new features and fix bugs on time. The need to accelerate software delivery, supporting faster time-to-market and frequent community developers/users feedback are issues that have lead to relevant changes in software development practices. In this way, each OSS project has its own priorities established by its respective communities. **Goal:** Identify the main characteristics of software release initiatives in OSS projects, the motivations behind their adoption, strategies applied, as well as advantages and difficulties found. Besides that search for evidence that clarify the relationships between target attributes, priorities assigned to the registered issues and the ways they are delivered by product releases. **Method:** In the first step of this work it was conducted a Systematic Literature Review (SLR) to reach one of the goals. In the second step an exploratory study was planned to identify quality attributes through the data analysis of repositories of three well-known OSS projects: Libre Office, Eclipse and Mozilla Firefox.**Results:** The SLR includes 30 publications from January 2006 to July 2016 and reveals nine advantages that characterize software release approaches in OSS projects; four challenge issues; three possibilities of implementation and two main motivations towards the adoption of RR; and finally four main strategies to implement it. the exploratory study verified that evidence provided by the participants suggest that OSS community developers use criteria/priorities driven by specific software product quality attributes to plan and perform software releases. **Conclusion:** This work provides an up-to-date and structured understanding of the software release approaches in the context of OSS projects based on findings systematically collected from a list of relevant references in the last decade. And finally an indication that the prioritization of quality attributes in projects tend to consider the profile of each project.

Keywords: Systematic Literature Review. Exploratory Study. Frequent Releases. Open Source Software Projects. Quality Attributes.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

OSS	*Open Source Software*
RSL	*Revisão Sistemática da Literaturae*
IDE	*Integrated Development Environment*
URL	*Uniform Resource Locator*
SRD	*Sistema de Rastreamento de Demandas*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta os conceitos iniciais desta dissertação, problemas a serem abordados, objetivos, metodologia de pesquisa adotada, contribuições, além da estrutura dos próximos capítulos.

1.1 CONTEXTO

A efetividade e sucesso de práticas adotadas por projetos open source (OSS) têm despertado interesse da comunidade de engenharia de software (MICHLMAYR et al., 2015) (FITZGERALD, 2006) (GONZALEZ-BARAHONA et al., 2013) o desenvolvimento OSS difere do desenvolvimento proprietário em vários aspectos, tais como processos de desenvolvimento, estruturada equipe e incentivos ao desenvolvedor (JONSSON et al., 2015). Entender como esses projetos de software funcionam permitirá que as empresas tirem lições das melhores práticas relatadas e as apliquem em seus projetos internos (STOL et al., 2015) (RIGBY et al., 2012). De acordo com (ADAMS et al., 2016), existem mais de 400 distribuições OSS ativas. Devido à concorrência e à pressão dos usuários e desenvolvedores, os projetos de OSS precisam lançar novos recursos e correções de bugs em intervalos de tempo cada vez mais curtos. A adoção de releases freqüentes em OSS leva em conta que esses projetos geralmente têm usuários em todo o mundo que, ansiosamente, baixam cada nova versão assim que ela é lançada e a testam o mais completamente possível. A dispersão global de usuários significa que o código pode ser testado 24 horas por dia (THOMAS et al., 2009) O estudo acerca da qualidade de software é analisado através de diferentes perspectivas, principalmente em relação à qualidade de processos e à qualidade de produtos (Matos, 2012). Entre as principais normas existentes para análise da qualidade do produto destaca – se a ISO/IEC 25010 (ISO, 2011) que substituiu a norma “antiga” ISO/ IEC9126 (ISO/ IEC,2001).

1.2 ABORDAGEM DE PESQUISA

Nesta seção são elicitados os problemas, objetivos, metodologia e contribuições.

1.2.1 Problemas da Pesquisa

Considerando a variedade e quantidade de projetos, ainda não está claro na literatura se a adoção de releases frequentes contribuiu para o sucesso e qualidade destes projetos de software: Através da identificação das motivações, estratégias, vantagens e desvantagens serão possíveis caracterizar que resultados os projetos que adotaram releases frequentes obteve;

Dificuldades dos projetos na decisão de quais atividades serão priorizadas nas próximas releases: O conhecimento de quais atributos de qualidade devem ser priorizados na escolha de atividade a serem disponibilizadas no planejamento das releases.

1.2.2 Objetivos da Pesquisa

Geral

- OG1 - Identificar características comuns e específicas de projetos open source bem sucedidos em relação à qualidade de produto e dos processos de releases frequentes.

Específicos

a) OE1 - Identificar características, motivações e estratégias utilizadas na adoção das releases freqüentes;

b) OE2 - Identificar as principais vantagens e desafios enfrentados pelos projetos open source que adotaram releases frequentes;

c) OE3 - Analisar de forma qualitativa através de repositórios públicos de projetos open source a influência de atributo de qualidade de produto de software no planejamento das releases.

d) OE4 - Buscar evidências que mostrem a relação entre os atributos de qualidade, as prioridades atribuídas e como elas são entregues nas releases desses produtos. Analisar o uso de estudos de caso publicados na literatura no contexto da migração de sistemas legados para a nuvem.

1.2.3 Metodologia

A metodologia utilizada nessa dissertação foi inicialmente a revisão da literatura com foco nos temas: OSS, planejamento de releases e atributos de qualidade. Depois foi feito uma revisão sistemática e um estudo explorátorio. O Capítulo 3 apresenta com detalhes a metodologia de pesquisa adotada nesta dissertação.

1.2.4 Contribuições

A primeira contribuição está relacionada à caracterização de como são definidas a prioridades em relação ao planejamento das releases em projetos open source bem sucedidos, dessa forma, outros projetos podem se espelhar nessas soluções. A segunda é compartilhar o conhecimento da relação de vantagens/desvantagens da adoção de releases freqüentes para os projetos open source possibilitando que projetos possam decidir sobre adotar esse tipo de abordagem.

1.3 RESULTADOS OBTIDOS

Os resultados obtidos foram os seguintes artigos publicados em conferências internacionais:

(a) Agility and Quality Attributes in Open Source Software Projects Release Practices (SILVA, A. C. B. G. ; PAULA, A. C. M. ; CARNEIRO, G. F. ; MONTEIRO, M. P. ; ABREU, F.B., 2016) Para identificar, classificar e comparar evidências existentes na literatura de escala de tempo, motivações, estratégias, vantagens e desafios na adoção de releases frequentes em projetos OSS. Esse trabalho foi aceito na 10ª International Conference on the Quality of Information and Communications Technology (QUATIC 2016). O principal autor desse artigo foi responsável pelo planejamento e execução da revisão sistemática, além da elaboração do texto em conjunto com os demais autores.

(b) On the Impact of Product Quality Attributes on Open Source Project Evolution (SILVA, A. C. B. G. ; CARNEIRO, G. F. ; MONTEIRO, M. P. ; ABREU F. B. ; CONSTANTINO K. ; FIGUEIREDO, E., 2016) O impacto dos atributos de qualidade de produto na evolução dos projetos open source, com o objetivo de procurar evidências sobre

como os atributos de qualidade são priorizados em projetos open source. Esse trabalho foi aceito na 14ª International Conference on Information Technology : New Generations (ITNG, 2017). O principal autor desse artigo foi responsável pelo planejamento, coleta e análise dos dados junto aos participantes do estudo exploratório, além da elaboração do texto em conjunto com os demais autores.

Além disso, o seguinte trabalho foi submetido a 9ª International Conference on Enterprise Information Systems (ICEIS, 2017), e encontra-se no aguardo do resultado.

(c) The Influence of Software Product Quality Attributes on Open Source Projects: A Characterization Study (SILVA, A. C. B. G. ; CARNEIRO, G. F. ; MONTEIRO, M. P. ; ABREU F. B. ; CONSTANTINO K. ; FIGUEIREDO E.; PAULA, A.C.M., 2016). Esse trabalho teve o objetivo de procurar evidências sobre as relações entre esses atributos, as prioridades atribuídas aos problemas registrados e como eles são entregues nas releases dos produtos. O principal autor desse artigo foi responsável pelo planejamento, coleta e análise dos dados junto aos participantes do estudo exploratório, além da elaboração do texto em conjunto com os demais autores.

E por fim o seguinte trabalho será submetido a um journal internacional abordando o conteúdo completo da revisão sistemática, que será apresentada nessa dissertação.

(d) Frequent Releases in Open Source Software: A Systematic Review (SILVA, A. C. B. G. ; PAULA, A. C. M. ; CARNEIRO, G. F. ; MONTEIRO, M. P. ; ABREU F. B., 2017)

1.4 ESTRUTURA DESTA DISSERTAÇÃO

Os próximos capítulos seguem a seguinte estrutura:

Capítulo 2: Neste capítulo é apresentada a Fundamentação Teórica, são descritos os conceitos sobre projetos open source, o planejamento de releases desses projetos e as principais definições acerca de atributos de qualidade de produto de software.

Capítulo 3: Este capítulo descreve o processo metodológico e o ciclo da pesquisa para o desenvolvimento do estudo, cuja metodologia utilizada foi uma revisão sistemática e posterior estudo explorátorio aplicado;

Capítulo 4: Apresenta e discute os resultados da Revisão Sistemática da Literatura com foco na identificação de como projetos decidem adotar e usar releases freqüentes, quais foram às motivações para a adoção, estratégias aplicadas e as vantagens e dificuldades encontradas;

Capítulo 5: Este capítulo apresenta um estudo explorátorio com a finalidade de identificar evidências que possa associar a priorização de determinado atributos de qualidade no planejamento de releases em projetos open source;

Capítulo 6: Este capítulo consolida as considerações finais desta dissertação, são destacadas as contribuições e perspectivas para trabalhos futuros.

1.5 CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou os conceitos iniciais desta dissertação, problemas, objetivos, metodologia utilizada, contribuições, além da estrutura que segue a presente dissertação. O capítulo seguinte apresenta o referencial teórico essencial para a compreensão desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos utilizados nesta dissertação. Para esta finalidade, são discutidos os conceitos gerais sobre projetos open source, planejamento de releases e atributos de qualidade de produto de software.

2.1 PROJETOS OPEN SOURCE

Open Source Software (OSS) é um termo abrangente usado para definir o software que é desenvolvido e lançado sob algum tipo de licença "open source" (CROWSTON et al., 2003). Dessa forma open source não pode ser definido apenas pelo acesso ao código fonte, mas sim pelo atendimento aos seguintes critérios (PERENS et al., 1999):

a) **Redistribuição Livre:** A licença não deve restrigir a venda, ou a doação do software ou qualquer componente. A licença não deverá exigir royalty ou qualquer outra taxa para venda.

b) **Código Fonte:** O software deve incluir o código-fonte, e deve permitir a distribuição do código-fonte, bem como o projeto compilado. O código deve ser legível e inteligível por qualquer programador.

c) **Trabalhos Derivados:** A licença deve permitir modificações e trabalhos derivados, e deve permitir que eles sejam distribuídos sobre os mesmos termos da licença original.

d) **Integridade do autor do código fonte:** A licença pode restringir o código fonte de ser distribuído em uma forma modificada apenas se a licença permitir a distribuição de arquivos patch (de atualização) com o código fonte para o propósito de modificar o programa no momento de sua construção. A licença deve explicitamente permitir a distribuição do programa construído a partir do código fonte modificado. Contudo, a licença pode ainda requerer que programas derivados tenham um nome ou número de versão diferentes do programa original.

e) **Não discriminação contra pessoas ou grupos:** A licença não deve descriminar qualquer pessoa ou grupo de pessoas.

f) **Não discriminação contra áreas de atuação:** A licença não deve restringir qualquer pessoa de usar o programa em um ramo específico de atuação. Por exemplo, ela não deve proibir que o programa seja usado em uma empresa.

g) **Distribuição da Licença:** Os direitos associados ao programa devem ser aplicáveis para todos aqueles cujo programa é redistribuído, sem a necessidade da execução de uma licença adicional para estas partes.

h) **Licença não específica a um produto:** Os direitos associados ao programa não devem depender que o programa seja parte de uma distribuição específica de programas. Se o programa é extraído desta distribuição e usado ou distribuído dentro dos termos da licença do programa, todas as partes para quem o programa é redistribuído devem ter os mesmos direitos que aqueles que são garantidos em conjunção com a distribuição de programas original.

i) **Licença não restrinja outros programas**: A licença não pode colocar restrições em outros programas que são distribuídos juntos com o programa licenciado. Isto é, a licença não pode especificar que todos os programas distribuídos na mesma mídia de armazenamento sejam programas de código aberto.

j) **Licença neutra em relação a tecnologia:** Nenhuma cláusula da licença pode estabelecer uma tecnologia individual, estilo ou interface a ser aplicada no programa.

Apesar de serem apoiados por voluntários, os projetos OSS relevantes têm especialistas em suas comunidades que têm um conhecimento aprofundado do código fonte e componentes (YE et al., 2003). Eles geralmente são classificados de acordo com sua reputação na comunidade do projeto como resultado de suas contribuições ao longo das releases lançadas (OREG, 2008).

2.2 PLANEJAMENTO DE RELEASE EM PROJETOS OSS

As estratégias de releases usadas por comunidades OSS podem ser classificadas em baseadas em funcionalidades e baseadas em tempo (RUHE, 2010). O primeiro emite uma release quando algumas metas foram cumpridas, por exemplo, um certo conjunto de recursos

foram implementados e/ou um conjunto de bugs pendentes foram corrigidos (FOGEL, 2015). Já ao adotar a estratégia de release baseada em tempo uma data é estabelecida para que seja disponibilizada essa nova versão (MICHLMAYR et al., 2012). Evidências da literatura sugerem uma preferência para estratégias baseadas no tempo entre projetos OSS (Michlmayr et al.,2012) (MICHLMAYR et al., 2015). Uma vez que a estrategia baseada em funcionalidades apresenta algumas dificuldades que podem ser superadas pela adoção de releases baseadas em tempo (MICHLMAYR et al., 2015). Entre os principais problemas enfretados destacam-se (MICHLMAYR et al., 2015). *Corrida no desenvolvimento*: Já que não existe uma data delimitada para a release, esta pode ser definida a qualquer momento, dessa forma desenvolvedores tendem a concluir todas as suas demandas, uma vez que não sabe quando haverá outra oportunidade para disponibilizar suas funcionalidades. *Dificuldade no planejamento*: Usuários ou empresas que atuam como consultores ou integradores de componentes apresentam essa preocupação, pois com um calendário de releases imprevisível o planejamento de ações a serem tomadas não podem ser otimizadas. *Software desatualizado*: normalmente nessa estrategia releases estáveis utilizados pelos usuários finais são versões antigas e ultrapassadas em relação às últimas versões de desenvolvimento *Ciclo vicioso de atrasos*: Assim que é definido o momento da release desenvolvedores passam a tentar acelerar suas correções, e caso não seja possivel cumprir o prazo de uma determinada demanda, essa só poderá ser corrigida numa release futura.

As adoções de releases baseadas em tempo em projetos OSS levam em consideração que esses projetos geralmente têm usuários espalhados pelo mundo, que avidamente obtém cada nova versão, tão logo seja liberado, e seja testado completamente. A dispersão global de usuários significa que o código pode ser testado 24 horas por dia (THOMAS et al., 2009). Para isso, contam com voluntários para analisar, implementar e testar as demandas registradas em ferramentas de bugs para integrá-los às últimas versões (ADAMS et al., 2016). O processo de elaboração de manutenção e/ou desenvolvimento é muitas vezes tratada através da utilização de Sistema de Rastreamento de Demandas (SRD) (nessa dissertação o termo demandas será utilizado para representar o termo *Issues*). Tipicamente, um SRD pode registrar para cada demanda seu tipo (por exemplo: defeito, melhoria, patch, tarefa), seu estado (por exemplo: novo, atribuído, resolvido, encerrado), a data de abertura e de cada mudança de estado, o autor, comentários, e indicações de severidade e/ou prioridade (BIJLSMA et al., 2012).

Varios aspectos podem influenciar a prioridade na resolução dessas demandas. *Tickets* recorrentes em SRD implicam em correção de erros (melhorando a confiabilidade) ou adição de novas funções (melhorando a funcionalidade), normalmente esses são os que a comunidade geralmente votam por atribuir uma ordem de programação (CROWSTON et al., 2003). No entanto, outros atributos de qualidade de produto de software podem ser críticos para o sucesso do projeto.

2.3 ATRIBUTOS DE QUALIDADE DE PRODUTOS DE SOFTWARE

A ISO/IEC 25010 é o padrão internacional para qualidade de produto de software (ISO, 2011). Ela provê um modelo de qualidade composto de várias características de qualidade de software, e essas possuem algumas subcaracterísticas, como mostra a figura 1. Cada atributo e suas subcaracterísticas serão apresentados a seguir:

Figura 1 - Atributos de qualidades e suas subcaracterísticas

Fonte: ISO (2011).

2.3.1 Funcionalidade

A capacidade do software para fornecer funções que atendam às necessidades declaradas e implícitas quando o software é usado sob condições especificadas. É composto pelas seguintes subcaracterísticas:

a) Adequação: Grau em que o conjunto de funções abrange todas as tarefas especificadas e os objetivos do usuário.
b) Precisão: Grau a que um produto ou sistema fornece os resultados corretos com o grau necessário de precisão.
c) Completude: Grau a que as funções facilitam a realização de tarefas e objetivos específicos.

2.3.2 Eficiência

A capacidade do software para fornecer o desempenho requerido, em relação à quantidade de recursos utilizados, sob condições estabelecidas. É composto pelas seguintes subcaracterísticas:

a) Comportamento em relação ao tempo: Grau a qual a resposta, os tempos de processamento e as taxas de produção de um produto ou sistema, ao executar suas funções, atendem aos requisitos.
b) Utilização de recursos: Grau no qual a quantidade e tipos de recursos utilizados por um produto ou sistema, no desempenho de suas funções, atendem aos requisitos.
c) Capacidade: Grau a que os limites máximos de um produto ou parâmetro do sistema satisfazem os requisitos.

2.3.3 Compatibilidade

A capacidade de um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes e / ou executar as funções requeridas, ao mesmo tempo em que compartilha o mesmo ambiente de hardware ou software. É composto pelas seguintes subcaracterísticas:

a) Coexistência: Grau em que um produto pode desempenhar suas funções necessárias de forma eficiente ao compartilhar um ambiente comum e recursos com outros produtos, sem prejuízo para qualquer outro produto.
b) Inteoperabilidade: Grau em que dois ou mais sistemas, produtos ou componentes podem trocar informações e usar a informação que foi trocada.

2.3.4 Usabilidade

A capacidade do software para ser compreendido, aprendido, usado e apreciado pelo usuário, quando usado em condições especificadas. É composto pelas seguintes subcaracterísticas:

a) Facilidade de Reconhecimento: Grau em que os usuários podem reconhecer se um produto ou sistema é apropriado para suas necessidades.
b) Aprendizabilidade: Grau a que um produto ou sistema pode ser usado por usuários especificados para atingir objetivos específicos de aprender a usar o produto ou sistema com eficácia, eficiência, liberdade de risco e satisfação em um contexto especificado de uso.
c) Operabilidade: Grau em que um produto ou sistema tem atributos que o tornam fácil de operar e controlar
d) Proteção a erro do usuário: Grau em que um sistema protege os usuários contra erros
e) Estética da interface do usuário: Grau em que uma interface de usuário permite uma interação satisfatória para o usuário.
f) Acessibilidade: Grau em que um produto ou sistema pode ser usado por pessoas com a mais ampla gama de características e capacidades para atingir um objetivo especifico em um contexto especificado de uso.

2.3.5 Confiabilidade

A capacidade do software para manter seu nível de desempenho quando usado em condições especificadas. É composto pelas seguintes subcaracterísticas:

a) Maturidade: Grau em que um sistema, produto ou componente atende às necessidades de confiabilidade em condições normais de operação.
b) Disponibilidade: Grau em que um sistema, produto ou componente é operacional e acessível quando necessário para uso.
c) Tolerância às falhas: Grau em que um sistema, produto ou componente funciona como pretendido apesar da presença de falhas de hardware ou software.
d) Recuperabilidade: Grau em que, em caso de uma interrupção ouuma falha, um produto ou sistema pode recuperar os dados diretamente afetados e re-estabelecer o estado desejado do sistema.

2.3.6 Segurança

A capacidade de um produto ou sistema proteger informações e dados para que pessoas ou outros produtos ou sistemas tenham o grau de acesso a dados adequado aos seus tipos e níveis de autorização. É composto pelas seguintes subcaracterísticas:

a) Confidencialidade: Grau que um produto ou sistema garante que os dados sejam acessíveis somente para aqueles autorizados a ter acesso.
b) Integridade: Grau em que um sistema, produto ou componente impede o acesso não autorizado ou modificação de programas ou dados.
c) Não Repúdio: Grau em que as ações ou eventos podem ser provados, de modo que os eventos ou ações não possam ser repudiados mais tarde.
d) Responsabilidade: Grau em que as ações de uma entidade podem ser rastreadas exclusivamente para a entidade.
e) Autenticidade: Grau a que a identidade de um sujeito ou recurso pode ser provado.

2.3.7 Manutenibilidade

A capacidade do software e ser modificado. As alterações podem incluir correções, melhorias ou adaptação do software, mudanças no ambiente, nos requisitos e especificações funcionais. É composto pelas seguintes subcaracterísticas:

a) Modularidade: Grau em que um sistema ou programa de computador é composto por componentes discretos, de forma que uma mudança em um componente tem um impacto mínimo sobre outros componentes.
b) Reusabilidade: Grau em que um ativo pode ser usado em mais de um sistema, ou na construção de outros ativos.
c) Analisabilidade: Grau de eficácia e eficiência com que é possível avaliar o impacto de uma alteração pretendida num produto ou sistema para uma ou mais das suas partes, ou para diagnosticar se um produto possui deficiências ou causas de falhas, ou para identificar partes a serem modificadas .
d) Modificabilidade: Grau a que um produto ou sistema pode ser eficaz e eficientemente modificado sem introduzir defeitos ou degradar a qualidade do produto existente.

e) Testabilidade: Grau de eficácia e eficiência com que os critérios de teste podem ser estabelecidos para um sistema, produto ou componente e testes podem ser realizados para determinar se esses critérios foram cumpridos.

2.3.8 Portabilidade

A capacidade de um software ser transferido de um ambiente para outro. É composto pelas seguintes subcaracterísticas:

a) Adaptabilidade: Grau em que um produto ou sistema pode ser adaptado de forma eficaz e eficiente para hardware ou software diferente em ambientes operacionais, de desenvolvimento, ou de uso.
b) Instalabilidade: Grau de eficácia e eficiência com que um produto ou sistema pode ser instalado e/ou desinstalado com sucesso em um ambiente especificado.
c) Capacidade de Ser Substituído: Grau em que um produto pode substituir outro produto de software especificado para a mesma finalidade no mesmo ambiente.

2.4 CONCLUSÃO DO CAPÍTULO

Este capítulo discorreu acerca do referencial teórico necessário para a compreensão desta dissertação. Apresentou o conceito de projetos open source, planejamento de releases em projetos open source e atributos de qualidade de produto de software. A seguir será apresentada a metodologia adotada nesta dissertação.

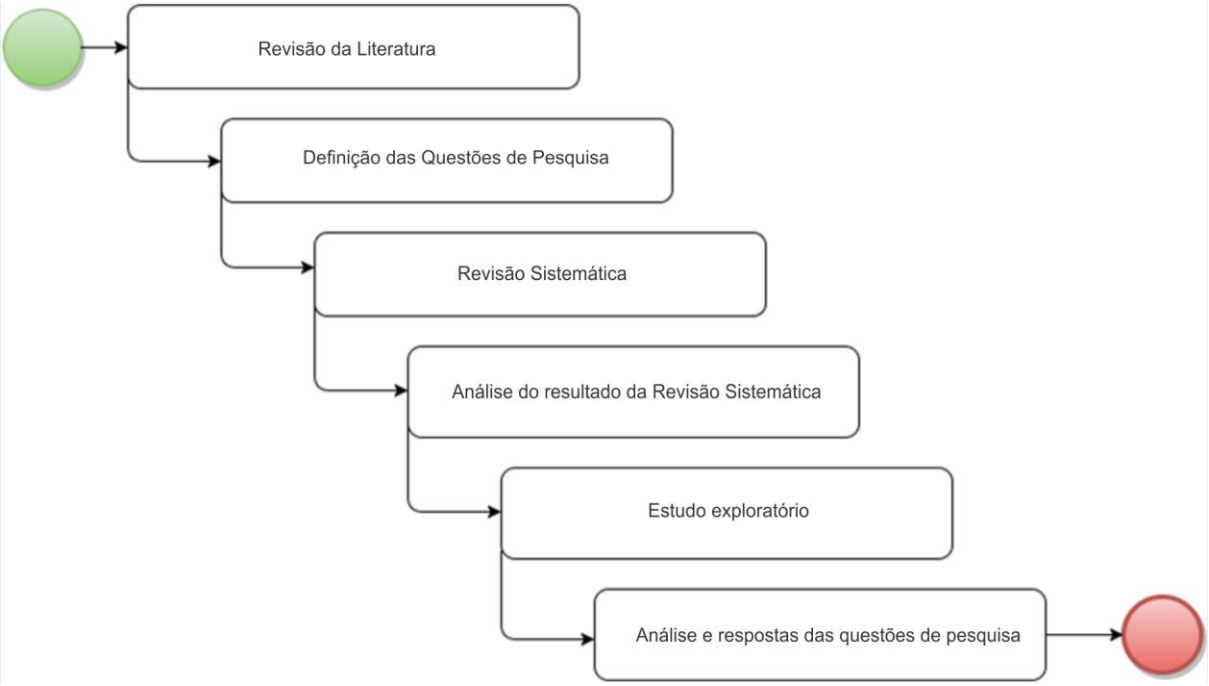
3 METODOLOGIA DE PESQUISA

Este capítulo descreve a metodologia de pesquisa utilizada nesta dissertação. As seções que se seguem apresentam como a pesquisa foi conduzida para atingir os objetivos propostos.

3.1 ETAPAS DA METODOLOGIA DE PESQUISA

As etapas que constituem a metodologia adotada nesta dissertação são ilustradas na Figura 2.

Figura 2 - Etapas da Metodologia de Pesquisa



A revisão da literatura realizada na primeira etapa desta dissertação permitiu levantar estudos relevantes ao tema abordado. Essa etapa auxiliou na definição das questões de pesquisa que nortearam a condução deste trabalho. Em sequência, a revisão sistemática é realizada para coletar evidências da literatura relacionadas às questões de pesquisa desta dissertação. Uma vez coletadas estas evidências, foi feita uma análise desses resultados, depois foi definido um estudo explorátorio. E por fim, a análise e respostas das questões de pesquisa.

3.2 REVISÃO DA LITERATURA

A revisão da literatura foi realizada no período de fevereiro a outubro de 2015 e possibilitou a coleta de trabalhos relacionados ao objetivo de pesquisa desta dissertação.

O principal objetivo foi identificar documentos que apresentassem conteúdo relevante sobre migração de sistemas para a computação em nuvem.

A compilação desses trabalhos foi base para o referencial teórico apresentado no Capítulo 2 e para a definição das questões de pesquisa.

3.3 DEFINIÇÃO DAS QUESTÕES DE PESQUISA

A definição das questões de pesquisa que norteiam esta dissertação foi realizada em novembro de 2015. As questões de pesquisa foram consolidadas através da revisão da literatura, dos problemas de pesquisa e dos objetivos gerais e específicos identificados anteriormente.

Os itens a seguir apresentam as Questões de Pesquisa (QPD) desta dissertação e em quais capítulos estão sendo tratadas de forma a garantir plena cobertura nos seus tratamentos:

a) **Questão de Pesquisa da Dissertação 1 (QPD1):** Quais os estudos mais relevantes publicados na literatura que abordam planejamento rápido de releases em projetos open source? Essa questão foi definida com foco em responder OG1 e OE1.
b) **Questão de Pesquisa da Dissertação 2 (QPD2):** Quais as principais estrategias e motivaçãoes que levam projetos open source adotarem releases frequentes? Essa questão foi definida com foco em responder OG1 e OE1.
c) **Questão de Pesquisa da Dissertação 3 (QPD3):** Quais as principais vantagens e dificuldades na adoção de releases frequentes em projeto open source? Essa questão foi definida com foco em responder OG1 e OE2.
d) **Questão de Pesquisa da Dissertação 4 (QPD4):** Quais atributos de qualidade são priorizados em projetos que adotaram releases frequentes? Essa questão foi definida com foco em responder OG1 e OE3.

e) **Questão de Pesquisa da Dissertação 5 (QPD5):** Quais as influências que determinados atributos de qualidade exercem, na priorização das atividades no planejamento das releases frequentes, em projetos open source? Essa questão foi definida com foco em responder OG1 e OE4.

3.4 REVISÃO SISTEMÁTICA

Foi realizada uma Revisão Sistemática de Literatura com o objetivo de identificar as principais características das iniciativas de releases frequentes de software em projetos OSS, as motivações por trás de sua adoção, as estratégias aplicadas, bem como as vantagens e dificuldades encontradas. Os resultados obtidos serão apresentados no próximo capítulo e foram suficientes para responder às questões: **QPD1**, **QPD2** e **QPD3**. Após o resultado da Revisão Sistemática, foi feita uma análise visando identificar os principais projetos open source que adotaram essa prática e qual seria a influência que os atributos de qualidade de produto de software exercem nesses projetos, com o objetivo de planejar a proxima fase.

3.5 ESTUDO EXPLORATÓRIO

Com o intuito de aprofundar o conhecimento sobre o relacionamento entre os projetos que adotaram releases rápidas e a influência que os atributos de qualidade exercem nesses projetos, um estudo exploratório foi proposto com foco em responder a questão de pesquisas: **QPD4** e **QPD5**.

3.6 ANÁLISE E RESPOSTAS DAS QUESTÕES DE PESQUISA

A partir das evidências encontradas na literatura tanto através da revisão sistemática como no resultado do estudo explorátorio foi possível analisar as questões de pesquisa desta dissertação e consequentemente apresentarem respostas para cada uma delas.

3.7 CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou a metodologia utilizada nesta dissertação, às etapas seguidas, questões de pesquisa, além de como foi feita a análise e respondidas as questões de pesquisa. O capítulo seguinte apresenta a Revisão Sistemática de Literatura.

4 REVISÃO SISTEMÁTICA

Este capítulo descreve a revisão sistemática de literatura adotada para identificar como projetos decidem adotar e usar releases freqüentes, quais foram as motivações para a adoção, estratégias aplicadas e as vantagens e dificuldades encontradas nesses cenários. As próximas seções também apresentam o planejamento, a execução e a análise dos resultados.

4.1 CONTEXTO

Em contraste com um processo de revisão não estruturado, uma Revisão Sistemática de Literatura (BRERETON et al., 2007) (KITCHENHAM ; CHARTERS, 2007) reduz o viés e segue uma sequência precisa e rigorosa de passos metodológicos para a pesquisa da literatura. Revisões Sistemáticas de Literatura (RSL) dependem de protocolos de revisão bem-definidos e avaliados para extrair, analisar, e documentar resultados e servem como subsidio para os estágios mostrados na figura 3.

A motivação para uma Revisão Sistemática da Literatura é a necessidade de busca de evidências na literatura sobre como projetos OSS decidem pela adoção e uso de Releases Freqüentes e quais são as motivações por trás dessa adoção; estratégias aplicadas, bem como as vantagens e dificuldades relatadas nesses cenários às evidências coletadas e discutidas nesta Revisão Sistemática da Literatura tem o objetivo de obter e compartilhar uma visão da literatura para que a comunidade OSS possa ter mais confiança e, portanto, ser capaz de tomar uma decisão sobre adoção ou não de processo de release freqüente. A principal contribuição deste trabalho é revelar as motivações por trás da adoção de release freqüente, estratégias aplicadas e a identificação das potenciais vantagens e dificuldades enfrentadas nesse sentido.

4.2 PLANEJAMENTO DA REVISÃO

Neste item é apresentado o planejamento feito para a execução da revisão sistemática.

4.2.1 Especificando as questões de pesquisa da Revisão

Com o objetivo de responder as questões de pesquisa a seguir foi conduzida uma revisão metodológica da pesquisa existente:

a) **QPRSL1. Qual a escala de tempo aplicada no planejamento de releases de software no contexto de projeto open source?** Conhecimento sobre a definição de planejamento de releases freqüentes da perspectiva de projetos que adotaram, pode ser usado como referência para novos projetos e potenciais adotantes.

b) **QPRSL2. Quais as principais motivações para a adoção de releases freqüentes no contexto de projetos open source?** Identificar objetivos, propostas e motivações para a utilização de releases freqüentes em projetos open source ajudará usuários em melhor caracterizar suas necessidades e, portanto, proporcionar condições adequadas para uma adoção bem sucedida.

c) **QPRSL3. Quais às principais estratégias adotadas para a adoção de releases freqüentes no contexto de projetos open source?** O conhecimento de estratégias eficazes adotadas pelos profissionais é um ponto de partida para a construção de um corpo de conhecimento sobre o uso de releases freqüente de software.

d) **QPRSL4. Quais são as principais vantagens e desafios relacionados com a adoção de releases freqüentes no contexto de projetos open source?** O conhecimento de estratégias bem sucedidas e problemas levantados pela adoção de releases freqüente de software contribuirão para organizações de desenvolvimento de software e outros projetos OSS serem mais confiante na adoção dessa prática.

Estes quatro questões de pesquisa não são independentes um do outro. No caso de **QPRSL2** e **QPRSL3**, motivações específicas podem levar à seleção de uma estratégia específica. Além disso, existe a possibilidade de que uma estratégia específica pode trazer vantagens e desafios específicos na adoção da release freqüente, revelando uma estreita relação entre **QPRSL3** e **QPRSL4**.

4.2.2 Período de publicações

Em relação ao período de tempo, consideraram-se artigos publicados em revistas e conferências a partir de janeiro de 2006 a janeiro de 2016.

4.3 CONDUZINDO A REVISÃO

Esta fase é responsável por executar o protocolo da revisão. As subseções seguintes descreverão as etapas realizadas cujo objetivo será a seleção dos estudos potencialmente relevantes alinhados às quatro questões de pesquisa desta revisão sistemática.

4.3.1 Identificação da pesquisa

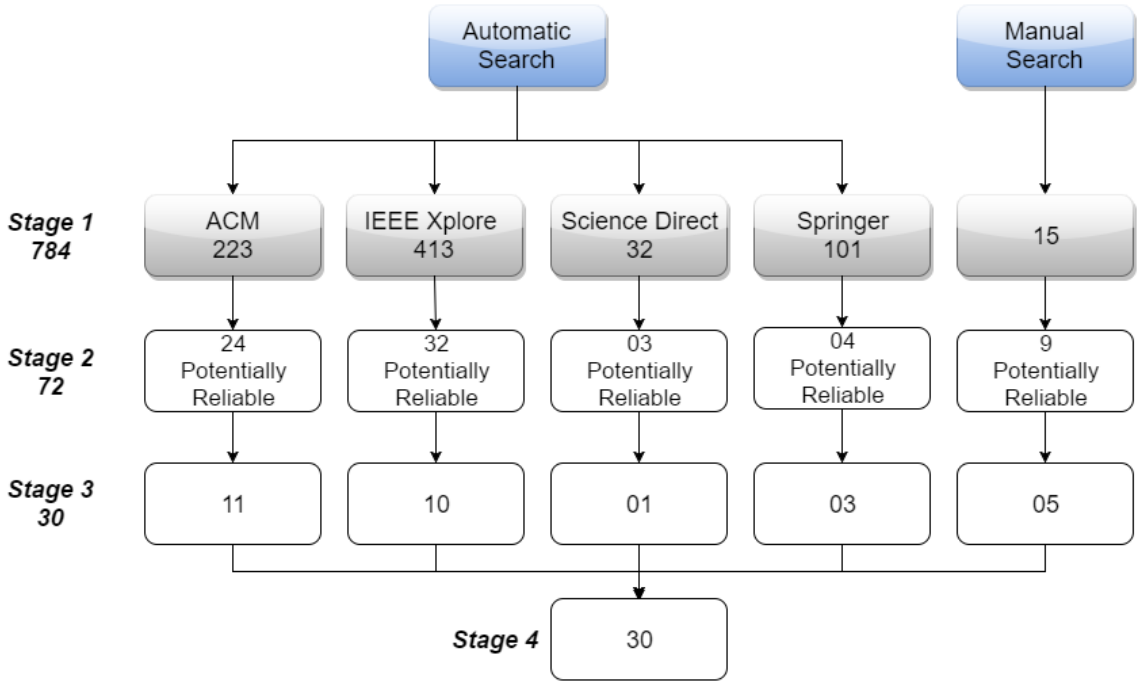
Com base nas questões da seção 4.2.1, palavras-chave foram extraídas e utilizadas para pesquisar as fontes primárias do estudo. A string de busca será apresentada a seguir e usa a mesma estratégia citada em (CHEN ; BABAR, 2011).

((”rapid release" or "fast release" or "frequent release" or "release history" or "quick release") and ("oss" or "open source" or "open source software"”)

4.3.2 Seleção de estudos primários

Os passos a seguir guiam a seleção de estudos primários representada na Figura 3.

Figura 3 - Estágios do Processo de Seleção de Estudos



4.3.2.1 Estágio 1

Resultados da string de busca obtidos automaticamente a partir dos sites de busca – Submissão da string de busca nos seguintes repositórios: ACM Digital Library, IEEE Xplore, Science Direct e Springer Link. A justificativa para selecionar estas bibliotecas é a sua relevância como fontes em engenharia de software (ZHANG et al., 2011). Dessa forma foi executada a string de busca em cada um desses repositórios, foi considerado somente título, palavras-chave e resumo. De acordo com Kithenham (KITCHENHAM ; CHARTERS, 2007) o uso de procedimentos por busca automática não são suficientes por si só, sendo que alguns pesquisadores defendem fortemente o uso de pesquisas, dessa forma foi adicionada uma busca manual para obter uma lista de estudos de journals e conferências. As duplicatas foram descartadas.

4.3.2.2 Estágio 2

Identificação de estudos potencialmente relevantes, com base na análise do título e do resumo, descartando estudos que são claramente irrelevantes para a pesquisa. Quando havia

alguma dúvida sobre se um estudo deve ser incluído ou não, foi incluído para consideração em estágios posteriores.

4.3.2.3 Estágio 3

Aplicar critérios de inclusão e exclusão na leitura da introdução, métodos e conclusão – Os estudos selecionados em etapas anteriores foram revisados, através da leitura da introdução, a seção de metodologia e conclusão. Em seguida, foram aplicados os critérios de inclusão apresentados no quadro 1 e exclusão apresentados no quadro 2. Nesta fase, sempre que surgiram dúvidas impedindo uma conclusão, foi feita a leitura do estudo na íntegra.

4.3.2.4 Estágio 4

Obter os estudos primários e efetuar uma avaliação crítica deles – Nesse estagio, a partir da lista obtida foram analisados os critérios de qualidade (DYBA ; DINGSOYR, 2008) estabelecidos no Quadro 3

Quadro 1 - Critérios de Inclusão

Critério de Inclusão (CI)	**Descrição**
CI1	As publicações devem ter sido publicadas em *journal* ou *conferências* e escritos em inglês.
CI2	Trabalhos envolvendo um estudo empírico ou que possuam lições aprendidas (relatórios de experiências).
CI3	Se vários artigos relatam o mesmo estudo apenas o último artigo será incluído.
CI4	Artigos que se referem a no mínimo uma das questões de pesquisa.

Quadro 2 - Critérios de Exclusão

Critério de Exclusão (CE)	**Descrição**
CE1	Estudos que não tenham foco em releases frequentes.
CE2	Estudos meramente com base na opinião de especialistas sem fornecer uma evidência.
CE3	Publicações que são versões anteriores do último trabalho publicado.
CE4	Publicações publicadas antes de janeiro de 2006 (nota: publicações após janeiro 2016 também não estão incluídos neste SLR, porque este foi o mês em que o processo de coleta terminou)

Quadro 3 - Critérios de Qualidade (Dyba e Dingsoyr, 2008)

Critério de Qualidade (CQ)	Descrição
CQ1	O artigo é baseado em uma pesquisa (ou é apenas um relatório de lições aprendidas baseado na opinião de especialistas)?
CQ2	Existe uma declaração clara dos objetivos da pesquisa?
CQ3	Existe uma declaração adequada do contexto em que a pesquisa foi realizada?
CQ4	O projeto de pesquisa possui o modelo adequado para lidar com os objetivos da investigação?
CQ5	A estratégia de recrutamento era apropriada aos objetivos da pesquisa?
CQ6	Havia um grupo de controle com o qual comparar os tratamentos?
CQ7	Os dados foram coletados em uma forma que aborda as questões de pesquisa?
CQ8	A análise dos dados foi suficientemente rigorosa?
CQ9	Existe uma declaração clara dos resultados?
CQ10	O estudo possui valor para a pesquisa ou prática?

4.3.3 Extração dos dados

Todas as informações relevantes sobre cada estudo foi gravado em uma planilha. Esta informação foi útil para resumir os dados e mapeá-los para suas fontes. Os seguintes dados foram extraídos dos estudos: (i) nome e autores; (II) tipo de artigo (revista, conferência, oficina); (III) quantidade de citações; (IV) ano de Publicação; (V) questão de pesquisa; (VI) os resultados e conclusões; (VII) benefícios; e (VIII) as limitações e desafios. Foram analisados os dados extraídos quantitativamente, em um esforço para responder às nossas questões de pesquisa.

4.3.4 Estudos potencialmente relevantes

Os resultados obtidos tanto na busca automática quanto na manual foram incluídos em uma única planilha. Nesta fase, foi registrado um total de 784 estudos, ou seja, 769 estudos da busca automática e mais 15 estudos da busca manual (Estágio 1). O próximo passo foi efetuar a leitura dos títulos e resumos para selecionar os estudos, resultando em 72 artigos (Estágio 2). No (Estágio 3) foram feito a leitura das introduções, seções sobre metodologia e as conclusões e aplicados os critérios de qualidade, nessa fase a ordem dos repositorios

analisados baseou-se na ordem alfabética, dessa forma o primeiro a ser analisado foi o ACM, seguido por IEEE, ScienceDirect e Springer, essa ordem influenciou na no tratamendo das duplicidades entre as base, e por fim foram definidos 30 estudos considerados para a fase seguinte. Depois de aplicar os critérios de qualidade, restaram 30 artigos para responder as três questões de pesquisa - **QPRSL1, QPRSL2 e QPRSL3, QPRSL4** (Estágio 4). Todos os estudos selecionados são listados no Apêndice A e referenciados com "S" seguido do número do artigo. O quadro 4 apresenta uma visão geral do processo de seleção por fonte de dados públicos.

Quadro 4 - Processo de Seleção por Repósitorio de Dados Públicos

Repositórios de Dados Públicos	Resultado da Busca	Estudos Relevantes	Efetividade da Busca
ACM	223	11	4,9%
IEEE	413	10	2,4%
Science Direct	32	1	3,1%
Springer	101	3	2,9%
Inclusão Manual	15	5	33,3%

4.4 ANÁLISE E RESULTADOS

Esta subseção apresenta os resultados desta revisão sistemática para responder as questões de pesquisa **QPRSL1, QPRSL2, QPRSL3 e QPRSL4**. Como resultado da análise dos estudos selecionados, os elementos-chave relacionados foram organizados de acordo com as quatro questões de pesquisa como mostra a Figura 4. A figura fornece uma visão geral das questões e conceitos relacionados a cada **QPRSL** juntamente com estudos selecionados. Os nós estão numerados para identificar os elementos da estrutura de acordo com a **QPRSL** que está relacionado. Nó 1 representa conceitos relacionados com a definição da escala de tempo das releases frequentes, neste caso **QPRSL1**: (1.1) ciclo regular, (1.2) Fluxo continuo (1.3) ciclo curto sem intervalo regular. O Nó 2 representa as motivações para implementar as RF, No caso de **QPRSL2** : (2.1) Atratividade e aumento do numero de participantes de projetos OSS, (2.2) Manutenção e melhora no MarketShare. nó 3 representa quatro possíveis estratégias para implementar RF (**QPRSL3**): (3.1) no contexto de desenvolvimento movido a teste, (3.2) através de processo de lançamento automatizado, (3.3) Distribuição Contínua e

(3,4) Atraves de releases baseada em tempo. As seguintes vantagens foram identificados no literatura (**QPRSL4**- parte 1): Time to Market (4.1.1), Qualidade: Eficiência, comentários, Satisfação do Cliente (4.1.2), Testes mais efetivos (4.1.3) Entrada de novos colaboradores (4.1. 4), alto ritmo de inovação (4.1.5), o planejamento eficaz e monitorado (4.1.6), Inclusão de novas funcionalidades (4.1.7), correção de erros (4.1.8), e atualizações de segurança (4.1.9). Os desafios (**QPRSL4**- parte 2) relatados estavam relacionados a pressão com o tempo (4.2.1), Dívida Técnica (4.2.2), Dependência da comunidade (4.2.3) e confiabilidade (4.2.4). A Figura 5 apresenta os estudos selecionados e as respectivas questões de pesquisa a qual se concentram, 22 estudos abordaram questões relacionadas com **QPRSL1** 15 estudos referem a **QPRSL2** 21 estudos foram relacionados para **QPRSL3** e, finalmente, 12 artigos responderam a **QPRSL4.** A Figura 6 mostra a distribuição temporal dos estudos selecionados.

Figura 4 - Mapa Mental

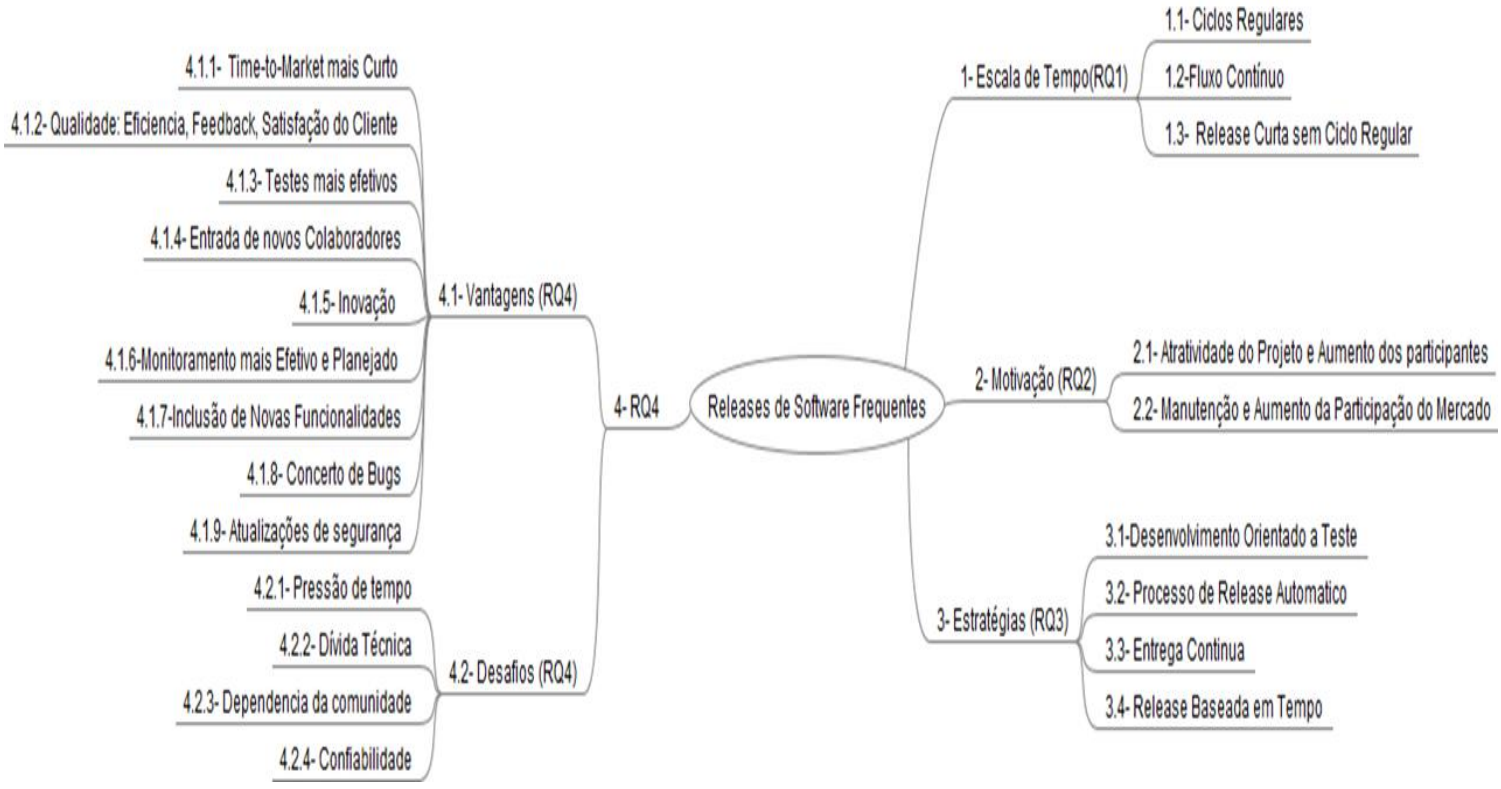


Figura 5 - Seleção por Questão de Pesquisa

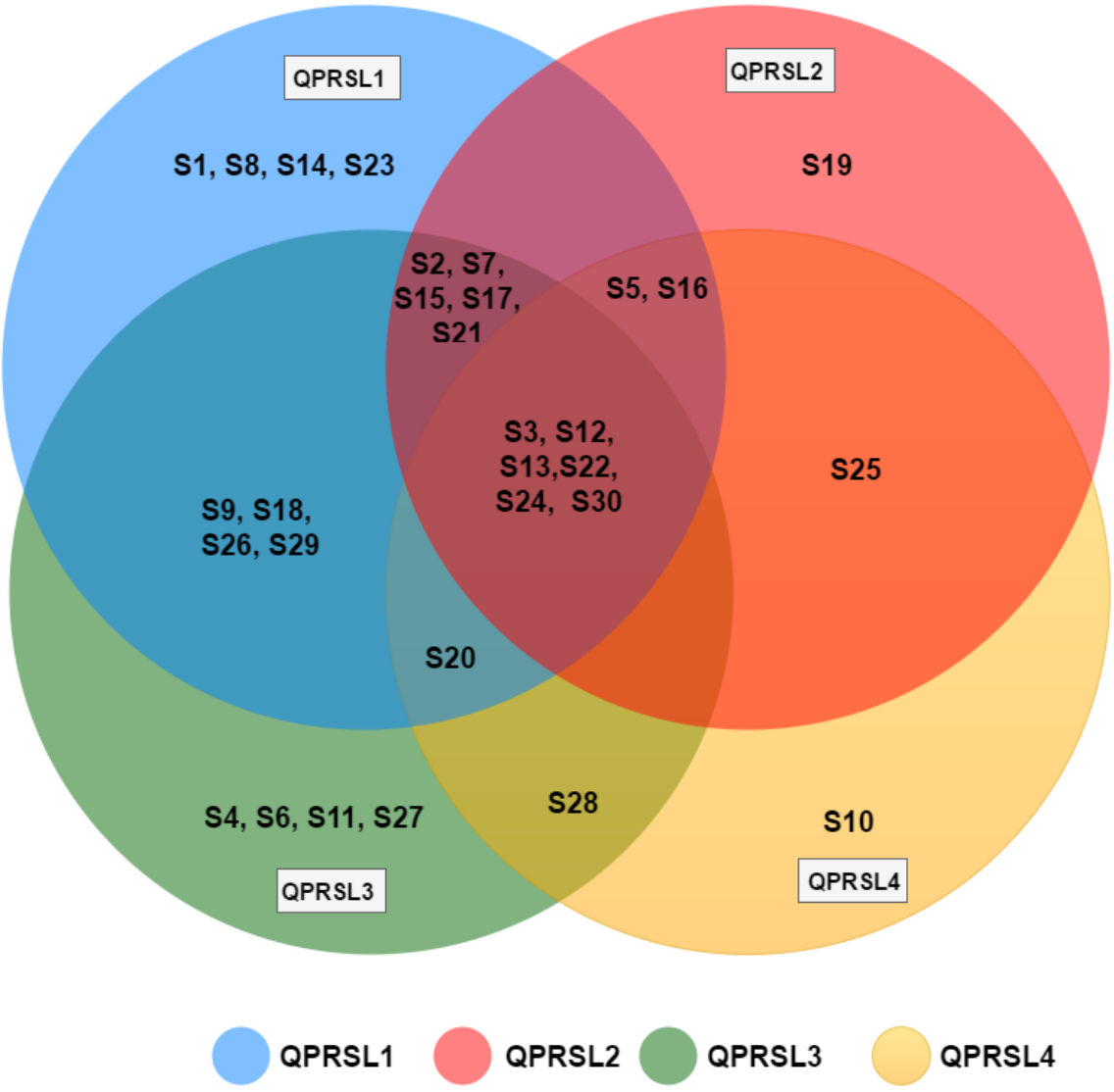
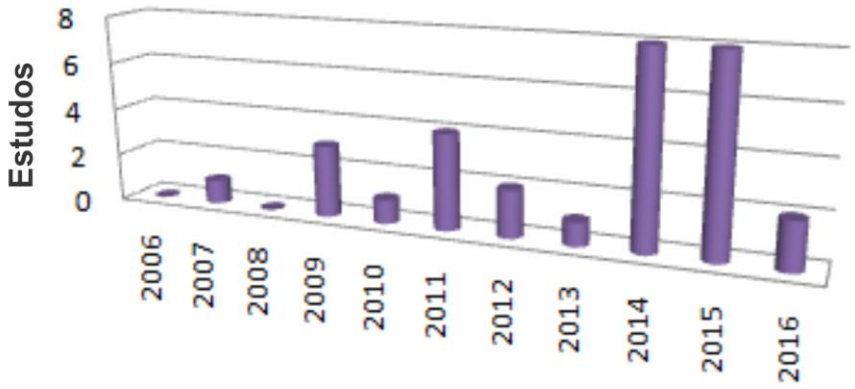


Figura 6 - Distribuição Temporal dos Estudos



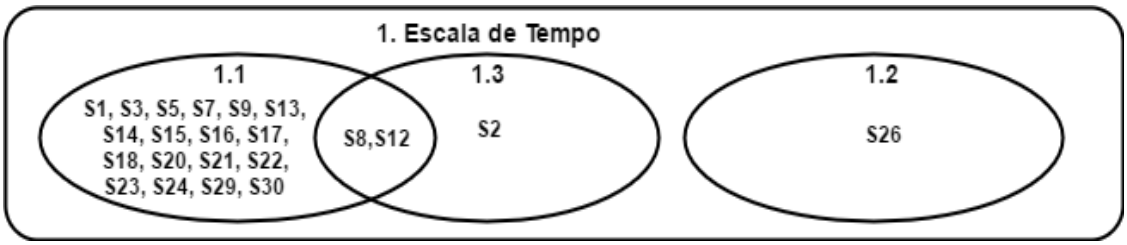
	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
Total Incluído	0	1	0	3	1	4	2	1	8	8	2

4.4.1 Escala de tempo aplicada no planejamento de releases de software no contexto de projeto open source (QPRSL1)

Este subitem tem o objetivo de discutir os trabalhos selecionados referentes à **QPRSL1: Qual a escala de tempo aplicada no planejamento de releases de software no contexto de projeto open source?**

Foram encontrados as seguintes evidências em estudos selecionados: lançamentos freqüentes pode ser caracterizado por *Ciclo Regulares* ou seja, releases fornecidos em intervalos regulares (S1, S3, S5, S7, S8, S9, S12, S13, S14, S15, S16, S17, S18, S20, S21, S22, S23, S24, S29, S30). No entanto, há estudos que também incluem releases freqüentes cujos intervalos de tempo em que são fornecidos novas versões do projeto de software não são regulares, isto é, em *Ciclo Curto Sem Intervalo Regular* (S2, S8, S12). Outra possibilidade de releases frequentes é a sua aplicação em um modo de *Fluxo Contínuo* (S26). Figura 7 representa a distribuição de artigos entre esses três tipos mencionados de escalas de tempo, em que 1.1 representa *Ciclo Regulares*, 1.2 *Fluxo Contínuo*, 1.3 *Ciclo Curto Sem Intervalo Regular*.

Figura 7 - Divisão dos Estudos por Escala de Tempo



No caso de *ciclos regulares* e de acordo com **S30**, muitos projetos se adaptaram para uma estratégia de gerenciamento de releases com base em tempo devido a primeiras experiências bem sucedidas de projetos como o projeto GNOME. Um grande número de projeto OSS basearam-se em tempo escolhendo um intervalo de release de até seis meses. Distribuições de Linux, como Fedora, são exemplos dessa prática.

S15 reconhece que o advento do desenvolvimento de projetos open source resultou em um modelo de desenvolvimento livre de restrições de negócios tradicionais, e capacitados pelo (potencialmente) milhares de desenvolvedores e um número ainda maior de testadores. De acordo com este modelo, as releases sucessivas podem ser disponibilizadas no mesmo dia. De acordo com este cenário, **S8** destacou a linha do tempo das releases do projeto Chrome

OSS e identificou a sua frequência como entre um e dois meses entre cada versão. **S16** também destaca que releases freqüentes fornece novas versões a partir no intervalo entre 2 semanas e dois meses. **S16** também observa que, em FR as modificações em cada versão são muito limitada, o que faz com que o número de falhas por releases sejam menores, em comparação com as versões tradicionais.

Em **S1**, os autores também utilizam dados do projeto Chrome para examinar sua história de releases, analisando relatórios de erros, dados de correções, e as estatísticas de uso. Eles observam que o Chrome fornece evidências de um sistema em evolução rápida, com ciclos de releases curtas e uma base de usuários que muito rapidamente adota novas versões assim que estiverem disponíveis (devido em grande parte do Chrome exigir atualizações automáticas obrigatórias). **S5** utiliza detectores de clones para compreender a evolução e para calcular a distribuição de mudança global entre todas as versões existentes do Firefox entre junho de 2011 e fevereiro de 2013. A abordagem dá idéias de como uma mudança de ciclo de releases podem influenciar a evolução de um software, mostrando uma tendência de aumento do risco em ciclos de releases mais rápidas, mesmo se as modificações de código estiverem menor. **S7** investiga as consequências e o impacto da metodologia de release rápida sobre a segurança do navegador Mozilla Firefox. Os dados resultantes mostra que as FR no Firefox não resultaram em taxas mais elevadas de vulnerabilidade. O padrão exibido pela divulgação da vulnerabilidade no Firefox é o resultado de usuários mal intencionados que têm que re-aprender e re-adaptar suas ferramentas em resposta a uma base de código que muda rapidamente. Os artigos **S3, S13, S17, S21, S22, S23 e S24** discutem a metodologia adotada pela Mozilla Firefox, que passou de uma abordagem tradicional versão de software com base em 12 - 18 meses para a entrega de novos releases a cada seis semanas. **S18** relata que Google Plus pode liberar novas alterações em 36 horas e o facebook pode chegar lançar novas versões duas vezes por dia durante a semana.

Em **S29**, é mencionado que estudos recentes começaram a avaliar o conceito de entrega contínua e releses frequentes a partir da perspectiva de qualidade de software. **S9** destaca que equipes ágeis se esforçam para entregar software funcionando em intervalos freqüentes variando de duas a quatro semanas. **S14** menciona que a idade do projeto e o tempo entre as releases é um fator que pode afetar o tamanho da equipe de desenvolvimento e a quantidade de esforço.

Considerando a possibilidade de *fluxo contínuo*, **S26** menciona que uma engenharia de software rápida e contínua se refere à "capacidade organizacional para desenvolver, liberar releases e aprender com software em ciclos paralelos rápidos.

No caso de *Ciclo Curto Sem Intervalo Regular* **S2** apresenta uma abordagem com uma série de melhores práticas de engenharia concebidas especificamente para atingir um ciclo de release semanal para um produto de software hospedado. Nota-se que o problema da gestão de release rápidos para um produto hospedado é susceptível a se tornar ainda mais importante no futuro.

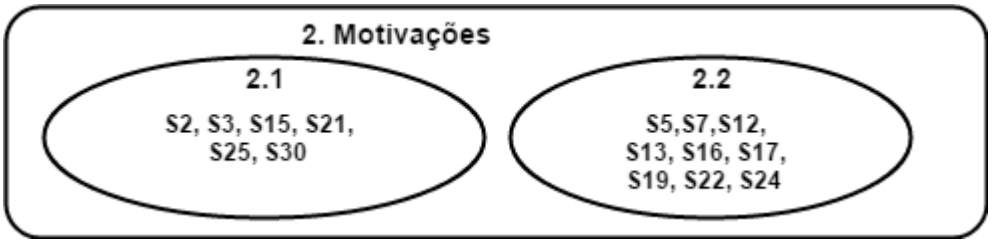
Em **S12**, os autores confirmam mais uma vez que as relases semanias é normalmente referido como um ciclo de release rápida, enquanto release mensal ou anual é tipicamente referido como um ciclo de release tradicional. Os autores também concluíram que 98% das atividades levantadas em sistemas com ciclo de release rápido (neste caso do projeto Firefox) foram entregues por uma ou mais releases. Por outro lado, os projetos com release tradicionais (neste caso ArgoUML), apenas o 34% a 60% de seus problemas foram entregues em um ou mais lançamentos. Isso fornece uma evidência preliminar de que projetos de OSS com releases frequentes tendem a abordar atividades mais cedo do que projetos que utilizam releases tradicionais.

4.4.2 Principais motivações para a adoção de releases freqüentes no contexto de projetos open source?

Este subitem tem o objetivo de discutir os trabalhos selecionados referentes à **QPRSL1: Quais as principais motivações para a adoção de releases freqüentes no contexto de projetos open source?**

Duas motivações principais foram identificadas como favorável para a adoção de releases frequentes: *atratividade do projeto / aumento de participantes* (S2, S3, S15, S21, S25, S30) e *manutenção e melhoria de participação no mercado* (S5 , S7, S12, S13, S16, S17, S19, S22, S24). Figura 8 representa a distribuição de papéis entre estes dois tipos de motivações 2.1 representando *atratividade do projeto / aumento de participantes* e 2.2 *manutenção e melhoria de participação no mercado*.

Figura 8 - Divisão dos Estudos por Motivações



S30 apresenta relatos de projetos em que o aumento do número de participantes, onde a maioria são voluntários, foi motivada pela atração e desafios associados às atividades de releases curtas. Além disso, levanta a possibilidade de feedback do usuário em um intervalo de tempo possível mais curto. A manutenção e melhoria da divisão de mercado ocorre como resultado do ritmo de mudanças e, portanto, novas funcionalidades fornecidas pelo software.

S15 e S30 apontam um aumento no número de participantes, a maioria dos quais se originam a partir da comunidade open source. O modelo de OSS amadureceu e participação em projetos de código aberto pode adicionar status para seus participantes. **S30** chama este modelo de "orientado a voluntário". **S2** destaca a pressão dos clientes que incentiva uma maior atenção por parte da equipe do projeto, para atender às demandas dos clientes. **S15** destaca o desenvolvimento do modelo OSS, que devido às suas características incentiva releases baseados no tempo. **S3** relata que um ciclo de relase mais curto fornece vários benefícios para as empresas e usuários finais. Empresas obtem feedback rápido sobre os novos recursos e correções de bugs e releases se tornam um pouco mais fácil para planejar (planejamento de longo prazo vs curto prazo). Os clientes se beneficiam, uma vez que têm acesso mais rápido a novos recursos, correções de bugs e atualizações de segurança. **S21** diz que uma das propriedades fundamentais de projetos OSS é que os membros da comunidade estão voluntariamente envolvidos em um projeto. **S25** relata que há um reconhecimento generalizado em toda a indústria de software que projetos OSS pode produzir sistemas de alta qualidade e funcionalidade, desenvolvimento OSS software é baseado em uma ideia relativamente simples: o núcleo original do sistema é desenvolvido localmente , em seguida, um protótipo de sistema é liberado, para que outros programadores possam livremente, modificar e redistribuir o código fonte desse sistema.

S5 relata que alguns projetos de software adotaram releases rápidos para competir melhor e com soluções mais rápidas. **S7** observa que a sobrevivência de um projeto de software implica a introdução frequente de novos recursos. **S12** destaca que os usuários e

colaboradores podem ficar frustrados quando uma atividade desenvolvida não está integrado em uma versão futura. **S13** observa que a principal motivação para mudar-se para ciclos rápidos (ou curtos) de releases é o tempo de colocação no mercado. **S19** aponta que versões do sistema frequentes que são realizadas em linha com as expectativas do usuário para uma maior capacidade de resposta e tempos de ciclo mais curtos.

S17 observa que a mudança para ciclos mais rápidos foram impulsionadas pela necessidade de atender às demandas dos usuários, em linha com a evolução dos padrões web e concorrência no mercado de browsers. **S24** ressalta que o projeto Firefox adotou o modelo de release rápida para ser capaz de competir com o Chrome, que já utilizava. A expansão rápida de plataformas móveis resultou em aumento da concorrência entre as aplicações, fazendo releases rápidas muito importante para as aplicações se manterem competitivas. **S16** destaca que o Firefox já estava a perder participação no mercado para o Chrome quando decidiu adotar as releases frequentes.

Finalmente, o avanço das plataformas móveis **S17** trouxe um grande incentivo para relases rápidos em que a maior facilidade de acesso resultou em um aumento da concorrência entre as aplicações. Isso fez as releases rápidas mais importantes para a aplicação ou serviço para se manter competitivo.

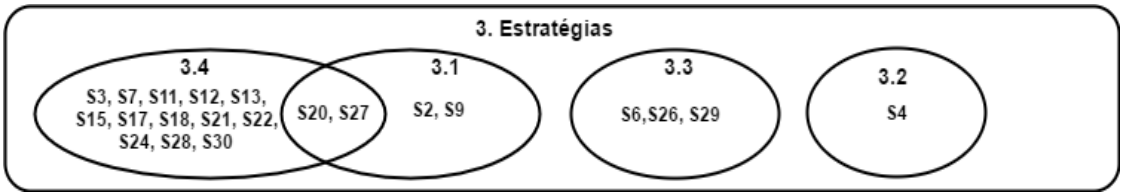
4.4.3 Principais estratégias para a adoção de releases freqüentes no contexto de projetos open source?

Este subitem tem o objetivo de discutir os trabalhos selecionados referentes à **QPRSL3: Quais as principais estratégias para a adoção de releases freqüentes no contexto de projetos open source?**

A estratégia de *release baseada em tempo* é mencionado como relevantes pelos seguintes estudos selecionados: (S3, S7, S11, S12, S13, S15, S17, S18, S20, S21, S22, S24, S27, S28, S30). Outros estudos mencionam um processo de qualidade eficaz associado com uma variedade de {*Ferramentas de teste automatizado* (S2, S6, S9, S27) como um aspecto essencial para apoiar releases freqüentes. *entrega contínua* é relatado como relevante para as empresas que praticam releases rápidas nas abordagens (S26, S29). Isto requer o conhecimento, gestão e monitorização do processo, bem como algum tipo de *processo automatizado*(S4). A figura 9 representa a distribuição de papéis entre estes quatro estratégias

para implementar versões de software, em que 3.1 representa *Ferramentas de teste automatizado,* 3.2 *processo automatizado,* 3.3 *entrega contínua* e 3.4 *release baseada em tempo*.

Figura 9 - Divisão dos Estudos por Estratégias



De acordo com **S28**, enquanto que uma estratégia de release baseada em tempo proporciona vários benefícios, é importante apontar que não necessariamente ela beneficia todos os projetos. Neste caso, não é aconselhável implementar versões regulares se houve pouco trabalho feito que justificasse uma nova versão. O ponto-chave é que uma implementação bem sucedida de releases baseados no tempo é baseada na confiança entre os seus colaboradores e o gerente do processo de release, bem como das estruturas de controle adequadas que devem ser aceitas pelos desenvolvedores. Uma vez que a versão baseada no tempo é escolhido, é hora de determinar o intervalo de release. Para este fim, **S28** identifica cinco fatores que afetam a escolha do intervalo: (i) a regularidade e previsibilidade; (ii) natureza do projeto e dos seus utilizadores; (iii) fatores comerciais; (iv) o custo e esforço; e (v) os efeitos de rede (necessidade de sincronizar o calendário de release do projeto com os horários de outros projetos a partir do qual ele pode alavancar benefícios).

S3, S7, S13, S21, S22, S24 descreve a estratégia do Mozilla, que é caracterizada por cada versão do Firefox substituir completamente a anterior. Além disso, cada nova versão passa pelo seguinte fluxo de 4 canais de release:

Nightly: Integra novos recursos de repositórios de código-fonte dos desenvolvedores, logo que eles estão prontos.

Aurora: herda as características do Nightly a cada seis semanas. Recursos que precisam de mais desenvolvimento são desativados e parte para o ciclo seguinte de importação.

Beta: recebe apenas os novos recursos de Aurora que estão programados para a nova versão do Firefox.

Principal: recebe a versão estavel do Beta.

Esta comutação entre os canais é definida a cada 6 semanas (**S12, S17**). **S11** também destaca no Mozilla a revisão do código como um mecanismo básico para a validação e implantação de patches. **S13** chama esse processo de modelo do trem, já que cada novo recurso e correção de erros flui ao longo dos repositórios. **S21** observa que o apoio da comunidade é composta basicamente por voluntários. **S27** aponta que o projeto Firefox foi forçado a contratar recursos de teste extras e corte na cobertura de testes de regressão.

S18 descreve a estratégia do Chrome, que, como Mozilla passa por um intervalo de 6 semanas com transições de código entre 3 canais: *Desenvolvimento*, *Beta* e *Estável*. Quando o desenvolvimento de uma nova versão começa releases anteriores mudam para os seus canais subsequentes.

S15 observa que a dispersão mundial de co-desenvolvedores significa que o código pode ser testado 24 horas por dia. **S30** explica que uma estratégia baseada no tempo especifica uma data para o lançamento com antecedência e um cronograma é tornada pública. Antes do release, há uma data de corte em que todos os recursos são avaliados no que diz respeito a estabilidade e maturidade. Uma decisão é então feita para saber se os recursos podem ser incluídos na release ou se deve ser adiada para o seguinte comunicado.

S4 aponta que, quando um processo de release está bem controlada e estavel (isto é, automática quando possível), as organizações podem adotar os ciclos de releases curtos.

S2 observa que é fundamental para um processo de controle de qualidade usar alguma ferramenta de testes automatizados de forma integrada, a fim de apoiar releases frequentes. **S9** salienta que o foco em testes é crítico para pequenas equipes que suportam releases rápidos. **S20**destaca que os projetos começaram a reconhecer o desenvolvimento orientado a teste como estratégico contra a concorrência, garantindo um software de alta qualidade.

S26 relata um mapeamento sistemático com base na entrega contínua e observa que a evolução contínua dos recursos conduzirão a este processo, permitindo, por exemplo, a entrega mensal de releases. **S29** também resalta a idéia de que as entregas contínuas são um fenômeno recente que é adequado para a releases rápidas de aplicações modernas. **S6** antes de tudo mostra a diferença entre a integração contínua, centra-se na automação do processo de compilação em um servidor central, e entrega contínua, que estende a abordagem anterior

para todos os fluxos de trabalho necessários para o teste e a implantação de uma nova compilação, consequentemente, simplifica a release do software e permite ciclos de feedback mais curtos entre desenvolvedores e clientes.

4.4.4 Principais vantagens e desafios relacionados com a adoção de releases freqüentes no contexto de projetos open source?

Este subitem tem o objetivo de discutir os trabalhos selecionados referentes à **QPRSL4: Quais são as principais vantagens e desafios relacionados com a adoção de releases freqüentes no contexto de projetos open source?**

Vantagens: Os principais pontos positivos associados aos lançamentos rápidos são: retorno rápido de necessidades do cliente, entrega rápida de novos recursos, correções rápidas de bugs, correções de segurança com liberação imediata (**S3, S22, S24**), aumento da eficiência (**S22**), entrada de novos colaboradores (**S20**), E maior foco na qualidade por parte de desenvolvedores e testadores (**S16, S22, S28, S30**). **S22** apresenta um trabalho que associam releases rápidas e testes. Ele relata que as releases freqüentes impulsionados por testes permitem um feedback maior e mais rápido em cada versão, e também aumenta o incentivo para os desenvolvedores entregarem um produto de qualidade. Destaca-se a maior eficiência devido a maior tempo de pressão. O estudo também acrescenta à lista de benefícios uma maior eficácia nos testes, devido ao tempo de teste reduzido, os testes são mais específicos e focados. Os testes focados também são gerenciados de forma mais fácil. **S16** ressalta a facilidade de planejamento e teste, uma vez que os testes são mais focados e executados com maior freqüência, facilitando o monitoramento do progresso e da qualidade (**S3, S22, S24**). **S24** propõe o aumento do ritmo de inovação, A alta taxa de releases encoraja a equipe a tentar continuamente novas soluções e novas ferramentas. Além disso, uma maior taxa de releases proporciona mais oportunidades de marketing para a empresa. S28 Nota que os intervalos de releases curtos permitem projetos OSS mais competitivos em comparação com projetos proprietários, uma vez que a release rápida traz vantagens significativas sobre os concorrentes que usam o ciclo tradicional. Por exemplo, o o ciclo beta de alguns produtos OSS corresponde a várias versões de novos produtos que chegam ao mercado. Outro fator levantado é o aumento da reputação e da satisfação dos colaboradores, pois eles vêem o seu código sendo rapidamente usado pelos usuários. **S30** nota uma tendência quanto à maturidade crescente da prática de OSS, com um aumento na sua significância e potencial econômico. Os projetos de

OSS estão adotando cada vez mais versões rápidas para permitir uma maior qualidade e sustentabilidade. O feedback mais rápido que estas práticas permitem também fornece mais informações sobre que partes do software necessitam uma maior atenção. **S20** apresenta um estudo de caso em que a equipe - que inicialmente era de pequeno porte e ocupava um pequeno escritório - aumentou para 20 desenvolvedores após adoção de release rápida e tomou quase todo um andar do prédio.

Desafios: Os principais desafios apontados são: Confiabilidade das novas versões (S12, S13, S22, S25), aumento da "dívida técnica" (S5, S22), pressão sentida pelos colaboradores (S10, S22) e dependência da comunidade (S16, S28, S30) . **S22** apresenta os principais pontos fracos de releases rápidas. Por um lado, os testes tornam-se mais focados, mas por outro lado, também se torna praticamente impossível testar todas as opções possíveis. Além disso, o curto tempo disponível não permite testar requisitos de qualidade de teste, por exemplo, desempenho. Outro ponto mencionado é o aumento da pressão sobre a equipe, que pode levar a exaustão. Por fim, o aumento da dívida técnica, uma vez que permite menos tempo para atividades como a refatoração. A negligência dessas questões pode comprometer a qualidade do software e impacta negativamente no longo prazo. **S25** também confirma que os requisitos de confiabilidade para o software exigem um tempo maior, o que impactaria nas releases frequentes. **S12** aponta que alguns problemas geralmente são adiados em um ciclo de release rápida. Finalmente, (**S13**) relata que ciclos curtos e prazos mais curtos implicam testes mais curtos, o que, a longo prazo, pode produzir versões menos estáveis. Da mesma forma, **S5** também afirma que a adoção de releases rápidas dificulta a manutenção de atividades de reengenharia, o que, em longo prazo, resulta no aumento do débito técnico. **S10** apresenta um estudo que conclui que a divisão tradicional entre operadores e desenvolvedores que ocorre em muitas organizações dificulta a adoção de releases rápidas. Em seu estudo, **S16** reivindica que a dificuldade em atrair um grande número de voluntários para a comunidade de teste faz com que os testes em releases rápidos sejam mais executados nos fins de prazo. **S28** afirma que os projetos são mantidos exclusivamente por voluntários requerem um planejamento significativo para lidar com períodos de pouca atividade, por exemplo, Natal.

4.4.5 Projetos de OSS relatados nos estudos selecionados

No quadro 5 apresentamos uma lista selecionada de projetos de OSS que foram identificados nos estudos. Este SLR, com foco na implementação de práticas de lançamento

de software. Na tabela, cada software Projeto é referido como "P" seguido por um número. A tabela também contém o nome do projeto, a Estudo que relatou suas práticas eo respectivo tempo de liberação adotado por cada projeto de software. Nas frases seguintes descreveremos brevemente cada um desses projetos. O projeto de software **P1** (Apache Cocoon) é um framework baseado em Spring (desde a versão 2.2 do Cocoon) construído em torno do Conceitos de separação de preocupações e desenvolvimento baseado em componentes. Cocoon implementa estes Conceitos em torno da noção de pipelines componentes, cada componente no pipeline especializado em Uma determinada operação. O projeto de software **P2** (Apache Tomcat) é uma implementação de código aberto Das tecnologias Java Servlet, JavaServer Pages, Java Expression Language e Java WebSocket. **P3** (ArgoUml) é ferramenta de modelagem uml open source. **P4** (Debian Linux) é um sistema operacional livre (OS) que é amplamente utilizado em todo o mundo. O **P5** (Eclipse) é um IDE suportado pelo Eclipse foundation, uma comunidade open source de ferramentas, projetos e grupos de trabalho colaborativos O **P6** (GNOME) é um sistema de desktop semelhante ao Windows que fornece um conjunto de aplicativos no Linux, FreeBSD, IRIX e Solaris. **P7** (Compilador GNOME) é uma coleção de compiladores originalmente Escrito como o compilador para o sistema operacional GNU. **P8** (Google Chrome) é um freeware web desenvolvido pelo Google. **P9** (O kernel do Linux) é um sistema operacional semelhante a Unix núcleo. **P10** (Mozilla Firefox) é também um navegador web freeware desenvolvido no contexto da Projeto Mozilla. Foi bem conhecida pela sua adoção da de releases rápida em detrimento da abordagem tradicional (KHOMH et al., 2012). **P11**(LibreOffice) é um conjunto de ferramentas de escritório gratuito e de código aberto desenvolvido Contexto do projeto Apache. **P12** (Plone) é um sistema de gerenciamento de conteúdo de código aberto Em cima do servidor de aplicativos de código aberto Zope e do Gerenciador de Conteúdo Estrutura. **P13** (Subversion) é um sistema de controle de versão de código aberto desenvolvido como um projeto de A Apache Software Foundation. O projeto P14 (X.Org) fornece uma implementação open source do sistema X.

Quadro 5- Projeto - Estudos - Release – Escala de Tempo

ID	ProjetoOSS	Estudos	Tempo de Release	Escala de
P1	Apache Cocoon	S19	Não informado nos estudos	RCSCR
P2	ApacheTomcat	S19	Não informado nos estudos	FC,RCSCR
P3	Argo Uml	S12	26 semanas	RCSCR
P4	Debian Linux	S28,S30	15-18 meses	FC,RCSCR
P5	Eclipse	S12	6 semanas	CR,FC
P6	Gnome	S25,S28,S30	6 meses	FC,RCSCR

ID	ProjetoOSS	Estudos	Tempo de Release	Escala de
P7	GNU Compiler	S8,S28,S30	6 meses	FC,RCSCR
P8	Google Chrome	S1,S8,S18,S24,S29	6 semanas	CR,FC
P9	Linux Kernel	S15,S18,S28,S30	2-3 meses	FC,RCSCR
P10	Mozilla Firefox	S1,S3,S5,S7,S12, S16,S17,S21,S22, S23,S24,S29	6 semanas	CR,FC
P11	LibreOffice	S28,S30	3-6 meses	FC,RCSCR
P12	Plone	S28,S30	6 meses	FC,RCSCR
P13	Subversion	S8	<=6 meses	FC,RCSCR
P14	X.org	S28,S30	6 meses	FC,RCSCR

4.5 LIMITAÇÕES E AMEAÇAS À VALIDADE

As seguintes ameaças à validade foram consideradas na interpretação dos resultados desta avaliação.

Validade da conclusão. Pode haver um viés na extração de dados. No entanto, este foi abordado através da definição de um formulário de extração de dados para garantir a extração consistente de dados relevantes e responder às questões de pesquisa. Os resultados e as implicações são baseados nos dados extraídos.

Validade interna. Uma possível ameaça é o viés de seleção. Esta ameaça foi abordada durante a etapa de seleção da revisão, ou seja, os estudos incluídos nesta revisão foram identificados através de um processo de seleção exaustiva, que dispõe de múltiplos estágios.

Validade da construção. Os estudos identificados a partir da revisão sistemática foram acumulados a partir de vários bancos de dados da literatura, abrangendo journals e anais relevantes. Uma possível ameaça é o viés na seleção de publicações. Esta questão é abordada através de especificação de um protocolo de pesquisa que define as questões de investigação e objetivos do estudo, critérios de inclusão e exclusão, string de busca na pesquisa que pretendemos utilizar, a estratégia de pesquisa e estratégia para a extração de dados.

Validade Externa. A quantidade de estudos de caso selecionados na caracterização pode não ter sido suficiente para generalizar os resultados. Entretanto, em função do protocolo adotado e dos critérios de qualidade, inclusão e exclusão foram utilizados recursos para se

garantir que os artigos selecionados para a resposta das questoes de pesquisa sejam parte de uma amostra significativa para o período de publicacao adotado.

4.6 CONCLUSÃO DO CAPÍTULO

Este capítulo discorreu acerca da Revisão Sistemática da Literatura realizada nesta dissertação. Foi apresentada a identificação de estudos primários, os critérios de inclusão, exclusão e qualidade usados na revisão sistemática, as fontes de informação e as estratégias de busca manual e automática em bases de dados. A seguir será apresentada a caracterização de estudos de caso.

5 ESTUDO EXPLORATÓRIO

Este capítulo apresenta um estudo exploratório para investigar a influência dos atributos de qualidade em práticas de releases em três projetos OSS (Libre Office, Eclipse e Mozilla Firefox).

5.1 TRABALHOS RELACIONADOS

Alguns estudos relataram a importância e influência que os atributos de qualidade de software têm no ciclo de desenvolvimento de software de um produto (PEREPLETCHIKOV et al., 2005; OFFUTT, 2002). Essa influência também é válida para projetos OSS. Duas pesquisas nessa direção foram conduzidas por Henningsson e Wohlin (2002). A primeira centrou-se na literatura para captar a compreensão dos atributos de qualidade na comunidade. A segunda é uma entrevista focada na percepção do mercado sobre a prática e compreensão dos atributos de qualidade em um contexto comercial. Os autores concluíram que é claro, tanto na literatura como no comércio, que existem relações entre vários atributos de qualidade do software.

A Fundação Eclipse também realizou pesquisas anuais com a comunidade do Eclipse para entender melhor algumas questões, como por exemplo, a percepção do usuário sobre as funcionalidades do Eclipse, ou como os usuários e desenvolvedores participam e se envolvem na comunidade OSS (FUNDATION, 2009).

5.2 ESTUDO EXPLORATÓRIO

Nessa seção, é apresentado um estudo exploratório para analisar como atributos de qualidade de software influenciam a priorização de demandas a serem implementadas/resolvidas e entregues em releases futuras. Estudos exploratórios têm a intenção de estabelecer as bases para um trabalho empírico mais aprofundado (WOHLIN, 2012).

Por esta razão, não há nenhum grupo de controle para comparar. A estratégia adotada consistiu em pedir aos desenvolvedores, fora da comunidade dos projetos selecionados, a

busca de evidências nos dados fornecidos pelos repositórios dos projetos OSS selecionados. Este estudo pretende abordar as seguintes questões de pesquisa:

QPEE1 - Considerando os dados disponíveis em repositórios públicos, quais atributos de qualidade de produto de software são priorizados nas práticas de release de software em projetos OSS? A identificação de atributos de qualidade de software em projetos OSS é fundamental para entender suas prioridades ao longo de sua evolução. Na prática, dependendo dos atributos de qualidade de software que são priorizados, os problemas específicos são direcionados e implementados nas próximas versões, enquanto outros serão adiados ou até descartados.

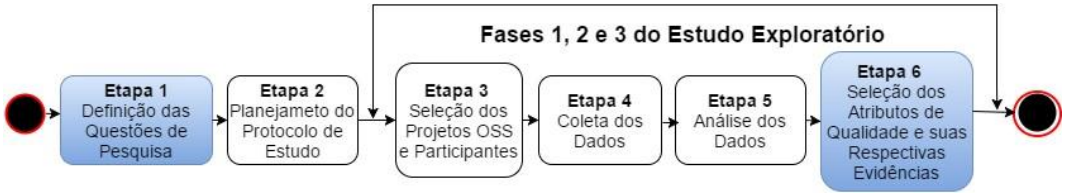
QPEE2 - Que estratégias podem ser usadas para identificar os atributos prioritários de qualidade de software dos repositórios de projeto OSS? Identificar estratégias para encontrar atributos de qualidade de software priorizados são úteis para entender o uso de recursos por projetos OSS. Práticas bem-sucedidas identificadas podem ser referenciadas e seguidas por profissionais do mercado bem como por outros projetos OSS.

QPEE3-Quais são os valores atribuídos pelos desenvolvedores aos campos "prioridade" e "gravidade" dos problemas registrados nos repositórios de bug dos projetos relacionados aos atributos prioritários e não prioritários de qualidade de produto de software? Essas evidências podem mostrar como os projetos de software OSS abordam outras questões, que não estão diretamente relacionados aos prováveis atributos de qualidade de software priorizados**.**

QPEE4-As demandas relacionadas aos atributos prioritários de qualidade dos produtos de software são entregues em releases logo após serem registrados? O tempo decorrido para entregar um problema pode ser uma evidência de sua prioridade, de acordo com os atributos de qualidade de software relacionados a ele.

Para responder as essas perguntas, foi definido um protocolo e executado três fases. Essas fases que compõem este estudo são apresentadas na figura 10. Este protocolo se concentrou na análise de dados que têm relações com os problemas de demandas em três projetos OSS. Foi realizada uma sessão tutorial, para apresentar aos participantes as ferramentas e recursos que poderiam ser usados para executar as tarefas.

Figura 10 - Visão Geral do Estudo Exploratório



5.2.1 Coleta de Dados

Coleta de dados: A coleta de dados foi realizada a partir de respostas fornecidas pelos participantes. Para responder a QPEE1, coletamos informações relacionadas a atributos de qualidade. Para responder a QPEE2, coletamos evidências com base em referências fornecidas pelos participantes, por exemplo, telas de impressão e URLs dos quais a informação relevante é acessível. Para responder a QPEE3 e QPEE4, foi analisado as demandas indicadas pelos participantes.

Critérios de Avaliação: Após o tutorial, foi analisado os dados fornecidos pelos participantes de acordo com os dois critérios a seguir. O critério 1 foi a indicação de pelo menos um atributo de qualidade consistente com os dados do repositório analisado. Portanto, indicações inadequadas de atributos de qualidade não atendem a este critério. O critério 2 foi indicação de evidência a partir dos repositórios que apóiam a escolha do atributo de qualidade selecionado. Portanto, indicações inadequadas de evidência não atendem a este critério.

5.2.2 Projetos OSS Selecionados

Este estudo baseou-se em três projetos altamente conceituados: **Mozilla Firefox, Eclipse e Libre Office**. Quatro razões foram consideradas para essa seleção. Em primeiro lugar, adotaram a implementação de releases freqüentes (GAMALIELSSON, 2014) (COSTA et al., 2014). Em segundo lugar, eles têm uma comunidade de desenvolvedores muito ativa, composta de 290 (Libre Office), 1087 (Mozilla Firefox) e 113 (Eclipse) membros, respectivamente. Em terceiro lugar, todos os projetos fornecem um vasto leque de documentação, acessível ao público. Finalmente, esses projetos foram citados na literatura em diferentes estudos de engenharia de software, o que aumenta seu valor como base para o presente estudo (GAMALIELSSON, 2014).

5.2.2 Protocolo de Estudo

Sete participantes participaram deste estudo para responder as questões de pesquisa QPEE1 a QPEE4. Eles foram solicitados a registrar as evidências coletadas relacionadas à análise que realizaram no formulário do questionário. Além disso, os participantes foram convidados a descrever suas estratégias, bem como sua experiência ao usar os repositórios de projeto OSS para realizar as tarefas. Os participantes dispunham de uma semana para realizar as tarefas solicitadas.

O estudo envolveu sete participantes recrutados de um mestrado e doutorado em Ciências da Computação de duas universidades diferentes. Esse número de participantes ofereceu um *trade-off* razoável entre o esforço para planejar e executar o estudo e análise qualitativa detalhada e a generalização dos resultados (PFLEEGER, 1995; YIN ; CAMPBELL, 2003). Eles foram todos voluntários e nenhuma compensação foi fornecida para a sua participação neste estudo.

O interesse foi em fazer observações baseadas em uma análise qualitativa de repositórios de projetos OSS, sobre a possível influência de atributos de qualidade de produto de software na programação de demandas pendentes que podem ter afetado a evolução do produto de software, ao invés de testar hipóteses de causalidade usando inferência estatística. Para serem elegíveis para inclusão, os participantes preencheram um questionário de pré-estudo apresentado na tabela 6 em que foi possível descrever seu perfil, experiência em práticas de release de software, projetos OSS e atributos de qualidade de produtos de software. Na Tabela 7, as respostas correspondentes usando a seguinte escala: Nenhum / Baixo / Médio / Alto.

A Sessão Tutorial: Antes das tarefas, os participantes participaram de uma sessão tutorial focando em como buscar evidências no projeto *Open Stack IAAS Cloud Platform* usando dados fornecidos por ferramentas como: *Bugzilla*, *Git* (sistema de gerenciamento de configuração), *Gerrit* (ferramenta de revisão de código para o Git) e um *Wiki*. Essas ferramentas também são usadas pelos projetos OSS selecionados para este estudo. Foi selecioando um projeto diferente na sessão tutorial para evitar viés nos resultados do estudo. A primeira parte do tutorial focou em como entender e pesquisar dados usando os repositórios de ferramentas para o projeto *Open Stack*. A segunda parte focou em usar os repositórios

selecionados para identificar evidências que poderiam estabelecer relações com os atributos de produto de software correspondentes priorizados pelo *Open Stack IAAS Cloud Platform*.

Quadro 6 - Questionário Pré-Estudo

ID	Perguntas	Respostas
OSS-1	Minha experiência como usuário de produtos de software Open source é?	Nenhum/Baixo/ Medio/Alto
OSS-2	Minha experiência como membro/desenvolvedor de produtos De software open source é?	Nenhum/Baixo/ Medio/Alto
PRS-1	Meu conhecimento teórico em releases de software é?	Nenhum/Baixo/ Medio/Alto
PRS-2	Minha experiência como usuário de releases de software é?	Nenhum/Baixo/ Medio/Alto
PRS-3	Minha experiência com o desenvolvedor de releases de Software é?	Nenhum/Baixo/ Medio/Alto
AQ-1	Meu conhecimento teórico em atributos de qualidade de produto de software é?	Nenhum/Baixo/ Medio/Alto
AQ-2	Minha experiência com o usuário em demandas relacionadas a atributos de qualidade de produto de software é?	Nenhum/Baixo/ Medio/Alto
AQ-3	Minha experiência como desenvolvedor de demandas relacionadas a atributos de qualidade de produto de software é?	Nenhum/Baixo/ Medio/Alto

Quadro 7 - Resposta dos Participantes

ID	P1	P2	P3	P4	P5	P6	P7
OSS-1	Nenhum	Médio	Pouco	Alto	Médio	Pouco	Alto
OSS-2	Nenhum	Nenhum	Nenhum	Nenhum	Nenhum	Nenhum	Médio
PRS-1	Médio	Pouco	Pouco	Alto	Médio	Pouco	Pouco
PRS-2	Médio	Pouco	Pouco	Alto	Médio	Médio	Alto
PRS-3	Médio	Médio	Pouco	Alto	Médio	Nenhum	Médio
AQ-1	Pouco	Médio	Médio	Médio	Pouco	Pouco	Pouco
AQ-2	Pouco	Médio	Médio	Pouco	Nenhum	Médio	Médio
AQ-3	Pouco	Médio	Pouco	Nenhum	Nenhum	Nenhum	Pouco

5.3 ANÁLISE DOS DADOS

Esta seção apresenta a análise das respostas fornecidas pelos participantes às questões de pesquisa acima mencionadas. Na primeira fase alguns participantes indicaram os atributos

de qualidade melhor relacionado aos projetos e quais estratégias foram utilizadas, na segunda fase foram analisadas demandas com alto grau de severidade e urgência, com o objetivo de focalizar questões consideradas relevantes no contexto dos projetos analisados, já na fase 3 foram analisados a relação entre as demandas com maior grau de severidade e os atributos de qualidade selecionados pelos participantes na fase 1 do projeto.

5.3.1 Primeira Fase

Na primeira fase deste experimento, pediu-se aos participantes que indicassem os atributos de qualidade de produto de software que melhor se relacionam com cada projeto, juntamente com as evidências respectivas que se justificam a indicação. O objetivo foi responder às seguintes questões de pesquisas.

QPEE1 - Considerando os dados disponíveis em repositórios públicos, quais atributos de qualidade de produto de software são priorizados nas práticas de release de software em projetos OSS ? e **QPEE2 - Que estratégias podem ser usadas para identificar os atributos prioritários de qualidade de software dos repositórios de projeto OSS?** O quadro 8 apresenta os atributos de qualidade indicados pelos participantes para os três projetos. Os seguintes atributos, já mencionados na seção 2, foram considerados na tabela: *Funcionalidade (F), Eficiência (E), Compatibilidade (C), Usabilidade (U), Confiabilidade (CO), Segurança (S), Manutenibilidade (M)* e *Portabilidade (P*). Nas subseções seguintes serão apresentados os pontos principais apresentados pelos 7 participantes nessa primeira fase.

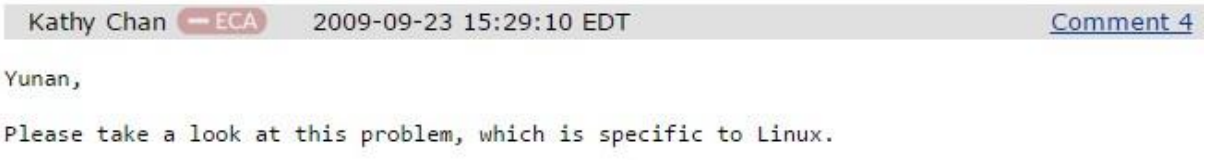
Quadro 8 - Atributos de Qualidade Selecionados pelos Participantes

	Eclipse								MozillaFirefox								LibreOffice							
	F	E	C	U	CO	S	M	P	F	E	C	U	CO	S	M	P	F	E	C	U	CO	S	M	P
P1		1			1			1		1						1	1			1				
P2	1		1	1			1	1		1		1		1	1	1	1	1		1			1	
P3		1	1	1	1		1	1	1	1	1	1	1	1		1	1		1	1				
P4			1				1						1		1					1			1	
P5			1				1						1		1					1			1	
P6							1								1								1	
P7	1		1	1				1			1			1		1	1		1	1				1
Tot	2	2	5	3	2	0	5	4	1	3	2	2	3	3	4	4	4	1	2	6	0	0	4	1

5.3.1.1 Participante 1

Para o projeto **Eclipse**: *Portabilidade*, *Confiabilidade* e *Eficiência* foram identificados. O participante justificou utilizando trechos do portal do projeto, a seguinte frase justificou a seleção dos atributos: *"é uma ferramenta de código aberto que fornece integração com várias outras ferramentas. Ele é executado em várias versões de sistemas operacionais (Portabilidade). A confiabilidade também é uma preocupação da comunidade Eclipse durante sua operação. Para lidar com esse problema, os testes são planejados e executados para encontrar falhas na tentativa de corrigi-los caso ocorram"* finalmente, o participante justifica a escolha de *Eficiência* com esta frase: *"tempo de resposta e consumo de recursos também são fatores importantes para Eficiência e são priorizados em questões correspondentes que abordam este assunto"*. Um exemplo de preocupação com *Portabilidade* é ilustrado no excerto a seguir, onde um desenvolvedor escreve para outro desenvolvedor *"dê uma olhada neste problema, que é específico para o Linux"*. A figura 11 (https://bugs.eclipse.org/bugs/showbug.cgi?id=290182i) representa essa evidência apresentada pelo participante.

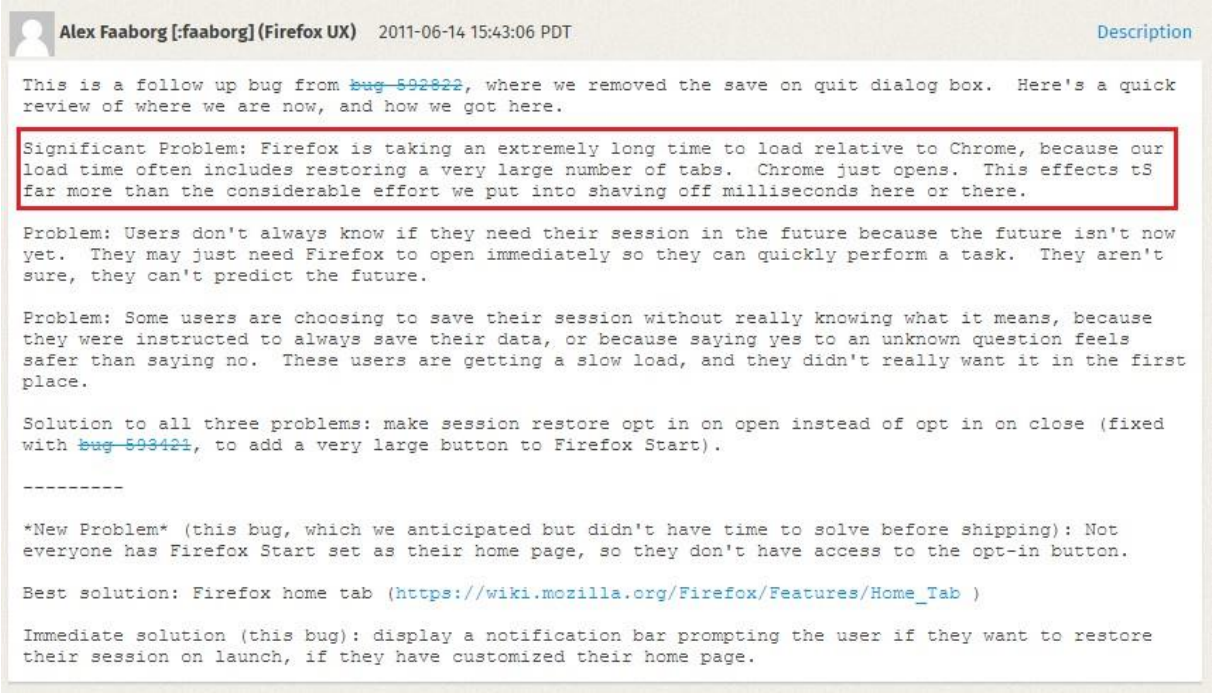
Figura 11 - Evidência de Portabilidade no Eclipse



O Participante 1 indicou *Portabilidade* e *Eficiência* como atributos de qualidade para **Mozilla Firefox**. De acordo com o participante, a indicação de *Portabilidade* estava relacionada à necessidade do "*navegador web ser compatível com diferentes tecnologias e também ser executado em diferentes sistemas operacionais"*. O manifesto do Mozilla Firefox afirma que a eficácia da Internet como recurso público depende da interoperabilidade (protocolos, formatos de dados, conteúdo), inovação e participação descentralizada em todo o mundo. E o projeto parece seguir este princípio. Em relação a *Eficiência*, este participante encontrou comentários registados nos *bugs* do Bugzilla, comparando Mozilla Firefox com Chrome, um dos seus principais concorrentes. O comentário seguinte, produzido por um dos desenvolvedores, é um bom exemplo dessa corroboração: *"problema significativo: o Firefox está demorando muito tempo para carregar em relação ao Chrome, porque nosso tempo de carga geralmente inclui a restauração de um grande número de Tabs."* A figura 12

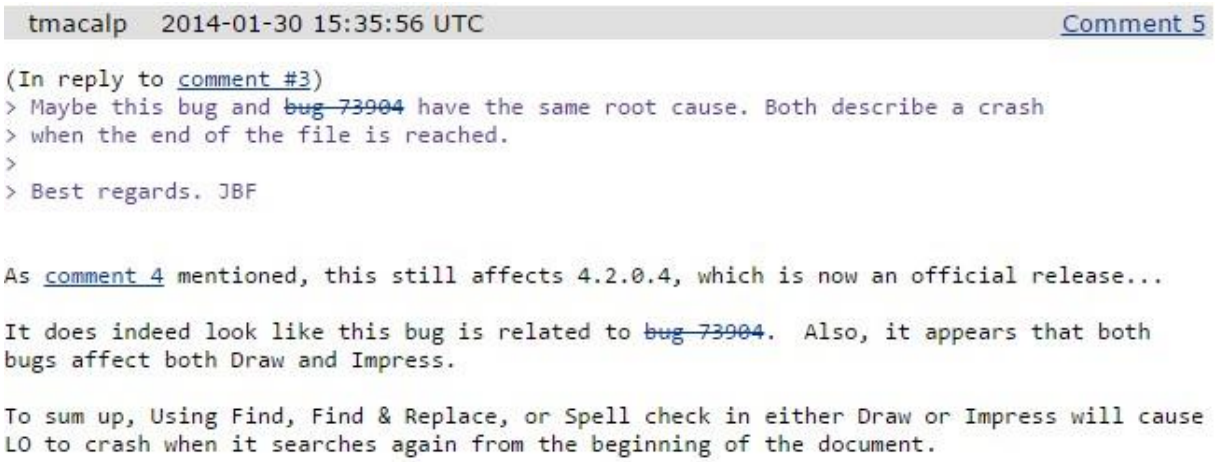
(https://bugzilla.mozilla.org/showbug.cgi?id=664314i) representa essa evidência apresentada pelo participante.

Figura 12 - Evidência de Eficiência no Mozilla Firefox



O mesmo participante identificou os atributos *Funcionalidade* e *Usabilidade* para o projeto **LibreOffice**. A razão para a indicação desses atributos foram identificadas como: `` é um software de suíte de escritório, que exige características intrínsecas ao seu domínio de aplicação e fortemente adaptadas às necessidades do usuário, ou seja, deve fornecer recursos que atendam às expectativas de seus usuários, Deve ser facilmente compreendido (fácil de usar) e ter uma interface gráfica atraente e moderna, contribuindo para um uso intuitivo e amigável ". O participante também identificou que os problemas relacionados a esses atributos foram marcados no repositório do bug do projeto como Alta prioridade para o LibreOffice. Por exemplo, o seguinte problema relata um erro relacionado à funcionalidade de localizar e substituir. Durante a interação um desenvolvedor menciona que o problema ocorreu em uma versão específica: *"isso ainda afeta o 4.2.0.4, que agora é uma versão oficial e estará disponível no LibreOffice 4.2.1"*. A figura 13 (https://bugs.documentfoundation.org/showbug.cgi?id=74104i) representa essa evidência apresentada pelo participante.

Figura 13 – Evidência de Funcionalidade no LibreOffice



5.3.1.2 Participante 2

Para o projeto **Eclipse**, *Funcionalidade*, *Compatibilidade*, *Usabilidade*, *Manutenibilidade* e *Portabilidade* foram propostos como critérios de qualidade. De acordo com o participante `` *Eclipse é um ambiente multi-plataforma completo, compatível com diferentes linguagens de programação (C, Java, PHP, etc.) e sistemas operacionais (Windows, Linux, iOS, etc.)*Como evidência, o participante apresentou uma tabela contendo estatísticas gerais sobre o status e as prioridades atribuídas à solução de problemas do projeto Eclipse. Esta tabela revela que as prioridades mais elevadas são dadas aos componentes da interface do usuário e do *Core* do Eclipse. Essa priorização também leva em conta a porcentagem de demandas relacionadas a esses componentes.

O Participante 2 propôs *Eficiência*, *Segurança*, *Usabilidade*, *Manutenibilidade* e *Portabilidade* como atributos para o projeto **Mozilla Firefox**. O participante observou que "*O Firefox é um navegador da Web disponível para muitas plataformas, incluindo Windows, OS X, Linux, Android e IOS e compatível com as tecnologias da Web atualizadas"*. O participante apresentou uma lista contendo a distribuição de problemas registrados nos repositórios de bugs através de seus vários status possíveis (atribuído, fechado, novo, reaberto, resolvido, não confirmado e verificado). Considerando o status e as soluções médias de bugs do projeto Mozilla Firefox, o participante concluiu que os usuários especialistas costumam marcar problemas relacionados a Usabilidade (Barra de Navegação e

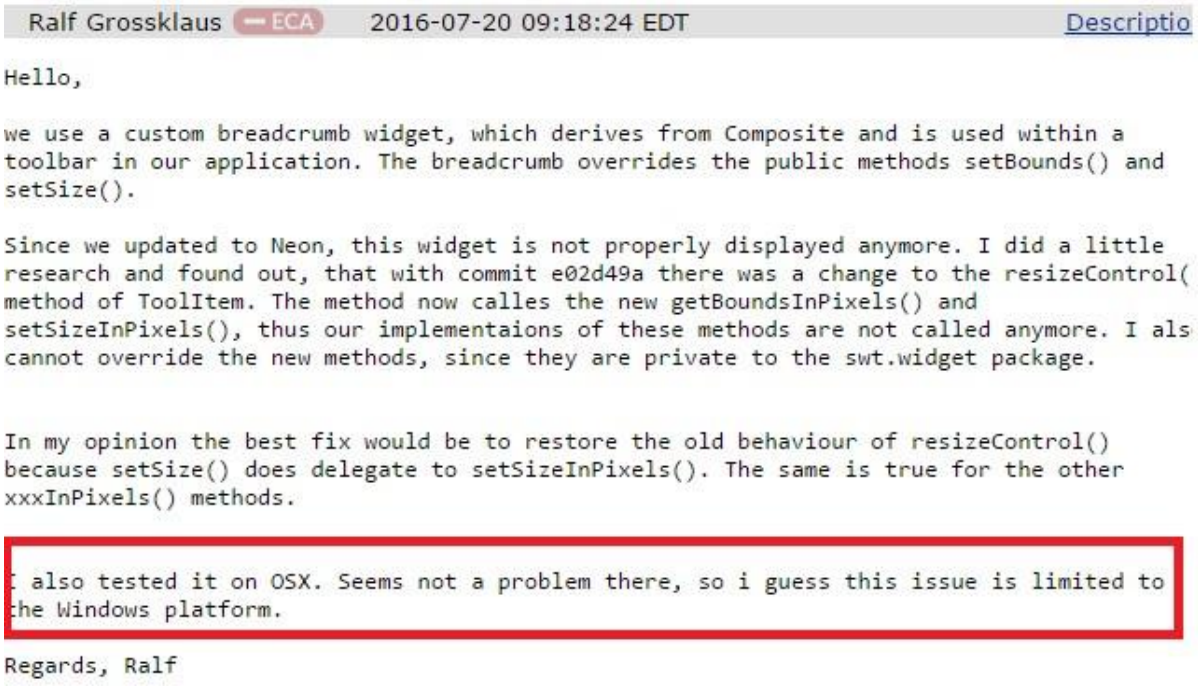
Personalização com Guia), Portabilidade , e Segurança (restauração da sessão, acesso deficiente) como prioridade alta no projeto.

Para o projeto **LibreOffice**, os atributos selecionados foram *Funcionalidade*, *Eficiência Usabilidade* e *Manutenibilidade*. O participante mencionou que o `` *Libre Office é um pacote de escritório poderoso com uma interface limpa e rico em ferramentas de produtividade. Ele inclui várias aplicações, tais como Writer (processamento de texto), Calc (planilha), Impress (apresentações), Draw (gráficos vetoriais e fluxogramas), Base (banco de dados) e Math(edição de fórmula). Funciona em Linux, iOS, Android e Windows "*. Da mesma forma que o projeto Eclipse, o participante também apresentou uma tabela contendo estatísticas gerais sobre o status e as prioridades atribuídas à solução dos problemas Libre Office. Sobre esta tabela, o participante destacou que "*ficou claro que de acordo com os dados dos últimos 500 bugs registrados, os termos mais freqüentemente usados entre a prioridade mais alta incluem: crash, merge, update, document, display, Calc, Wizard".*

5.3.1.3 Participante 3

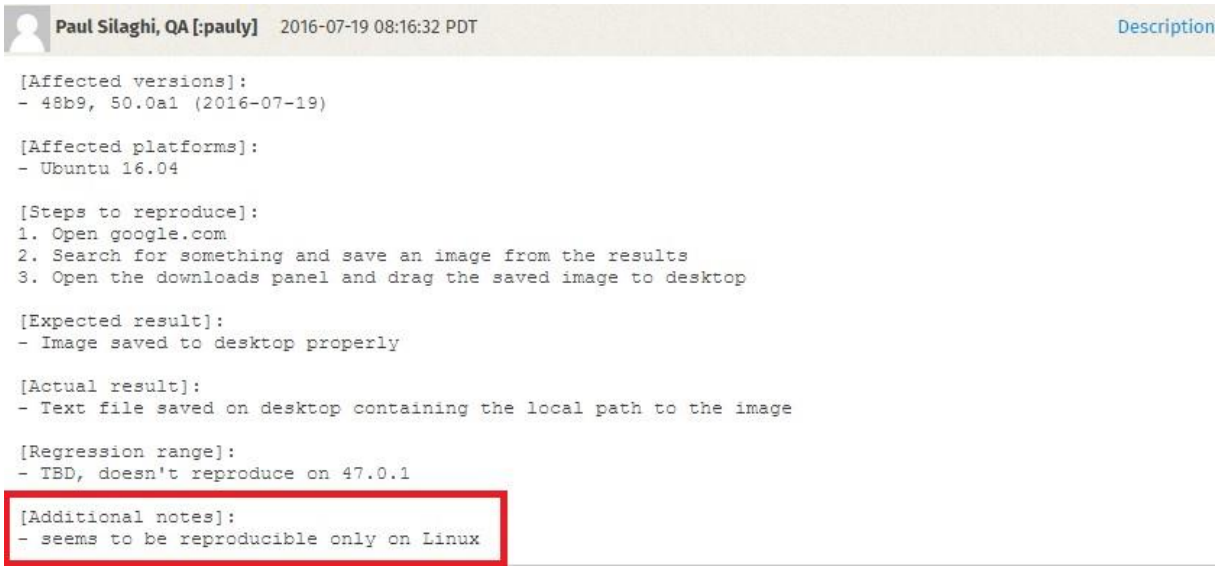
Para o projeto **Eclipse**, *Eficiência, Usabilidade, Manutenibilidade, Portabilidade, Confiabilidade* e *Compatibilidade* foram sugeridos. A justificativa revela uma preocupação, por parte da comunidade, em cumprir os critérios de qualidade de software apresentados a seguir. `` Eclipse fornece IDEs e plataformas para praticamente todas as linguagens e arquiteturas "; "*IDEs construídos em plataformas extensíveis para criar IDEs de desktop, web e cloud"*; "Desenvolva seu software onde quer que você vá." Para este participante, o bug mais proeminente foi evidenciado como "*A barra de ferramentas não exibe widgets personalizados corretamente*". Esse problema ocorreu em uma versão recém-lançada do Eclipse Neon e na plataforma Windows, como mostrado em "*Eu também testei no OS X. Parece não haver um problema, então eu acho que este problema é limitado à plataforma Windows*". A figura 14 (https://bugs.eclipse.org/bugs/showbug.cgi?id=498196i) representa essa evidência apresentada pelo participante.

Figura 14 – Evidência de Portabilidade no Eclipse



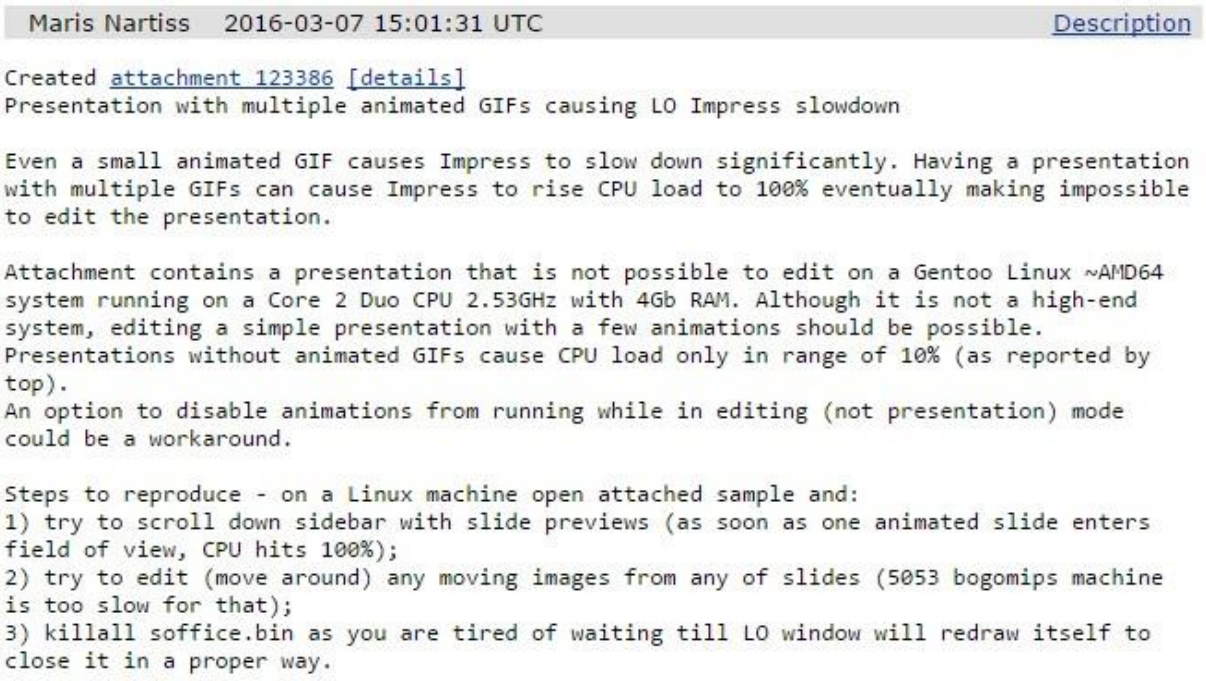
Funcionalidade, *Confiabilidade*, *Segurança*, *Compatibilidade*, *Portabilidade*, *Usabilidade* e *Eficiência* foram propostas como atributos de qualidade para **Mozilla Firefox**. O participante justificou que a comunidade está preocupada com diversos critérios de qualidade "*Firefox para desktop - Confiável, flexível, rápido ",` `Firefox para iOS - Fast, intelligent, Yours*"; "*Comprometido com você, sua privacidade e uma Web aberta", "você pode modificar o Firefox de acordo com suas necessidades*". A principal evidência apontada pelo participante foi um bug referente a uma versão do Mozilla Firefox, que ocorreu ao transferir uma imagem salva pelo navegador para a pasta de download. Esse comportamento foi evidenciado apenas no Linux, como afirmado *" parece ser reproduzível somente no Linux*". A figura 15 (https://bugzilla.mozilla.org/showbug.cgi?id=1287823**i**) representa essa evidência apresentada pelo participante.

Figura 15 - Evidência de Portabilidade no Mozilla Firefox



Para **LibreOffice**, *Usabilidade* e *Compatibilidade* foram propostos como critérios de qualidade relevantes. Como justificativa, o participante cita que o sistema *"inclui várias aplicações que o tornam um poderoso pacote de escritório livre e open source, é compatível com uma ampla gama de formatos de documentos como o Microsoft Word, Excel, PowerPoint e Publisher. Além disso, ele tem um padrão moderno e aberto, o Open Document Format (ODF)".* A partir de evidências apresentadas, destaca-se a preocupação dos desenvolvedores com o seguinte bug "*Múltiplos GIFs animados causam 100 % de utilização da CPU* ". Para corrigir esse bug, foi feito um ajuste de código para melhorar o desempenho da CPU, conforme apresentado em "*Para alcançar mais com a abordagem atual, foi necessário reimplantar AnimatedGIF import*". A figura 16 (https://bugs.documentfoundation.org/showbug.cgi?id=98500i) representa essa evidência apresentada pelo participante, já a figura 17 (https://cgit.freedesktop.org/libreoffice/core/commit/?id=285744fef87f4ca0278834b97d7f618bdba5f4c0i) representa a visão do gerrit demonstrando as modificações que foram feitas no código.

Figura 16 - Evidência de Usabilidade no LibreOffice



Maris Nartiss 2016-03-07 15:01:31 UTC Description

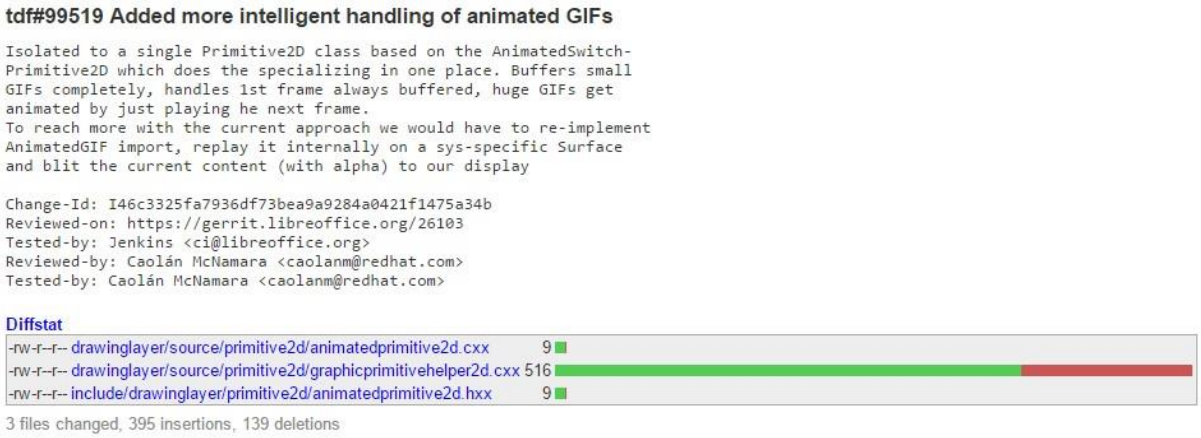
Created attachment 123386 [details]
Presentation with multiple animated GIFs causing LO Impress slowdown

Even a small animated GIF causes Impress to slow down significantly. Having a presentation with multiple GIFs can cause Impress to rise CPU load to 100% eventually making impossible to edit the presentation.

Attachment contains a presentation that is not possible to edit on a Gentoo Linux ~AMD64 system running on a Core 2 Duo CPU 2.53GHz with 4Gb RAM. Although it is not a high-end system, editing a simple presentation with a few animations should be possible.
Presentations without animated GIFs cause CPU load only in range of 10% (as reported by top).
An option to disable animations from running while in editing (not presentation) mode could be a workaround.

Steps to reproduce - on a Linux machine open attached sample and:
1) try to scroll down sidebar with slide previews (as soon as one animated slide enters field of view, CPU hits 100%);
2) try to edit (move around) any moving images from any of slides (5053 bogomips machine is too slow for that);
3) killall soffice.bin as you are tired of waiting till LO window will redraw itself to close it in a proper way.

Figura 17 - Ajuste Feito no Código LibreOffice



tdf#99519 Added more intelligent handling of animated GIFs

Isolated to a single Primitive2D class based on the AnimatedSwitch-Primitive2D which does the specializing in one place. Buffers small GIFs completely, handles 1st frame always buffered, huge GIFs get animated by just playing he next frame.
To reach more with the current approach we would have to re-implement AnimatedGIF import, replay it internally on a sys-specific Surface and blit the current content (with alpha) to our display

Change-Id: I46c3325fa7936df73bea9a9284a0421f1475a34b
Reviewed-on: https://gerrit.libreoffice.org/26103
Tested-by: Jenkins <ci@libreoffice.org>
Reviewed-by: Caolán McNamara <caolanm@redhat.com>
Tested-by: Caolán McNamara <caolanm@redhat.com>

Diffstat

-rw-r--r-- drawinglayer/source/primitive2d/animatedprimitive2d.cxx 9
-rw-r--r-- drawinglayer/source/primitive2d/graphicprimitivehelper2d.cxx 516
-rw-r--r-- include/drawinglayer/primitive2d/animatedprimitive2d.hxx 9

3 files changed, 395 insertions, 139 deletions

5.3.1.4 Participante 4

Para o projeto **Eclipse**, *Compatibilidade* e *Manutenibilidade* foram propostos. A seguinte citação foi fornecida como justificativa: `` O Eclipse possui um alto grau de compatibilidade com várias plataformas de desenvolvimento. Possui uma plataforma bem projetada e bem organizada para enviar bugs ao seu desenvolvedor. "A evidência apresentada ao Manutenibilidade foi a URL do projeto Bugzilla. Para Compatibilidade, o seguinte URL do Eclipse (http://www.eclipse.org/home/newcomers.php) foi apresentado.

Para o projeto **Mozilla Firefox**, foram propostos *Manutenibilidade* e *Confiabilidade*. O participante forneceu as seguintes evidências: este projeto *"fornece uma plataforma para a apresentação de bugs pelos desenvolvedores, bem como seu status. Ele usa um sistema de liberação de versão em que as versões são liberadas depois de submeter-se a testes de confiabilidade."* A principal evidência apresentada para justificar Manutenibilidade foi a URL do Bugzilla do projeto.

Para **LibreOffice**, *Manutenibilidade* e *Usabilidade* foram propostos. A escolha é justificada apontando que "O projeto apresenta uma plataforma bem projetada e bem organizada para apresentar bugs a desenvolvedores. Ele incentiva os usuários a participar na melhoria do software, através de feedbacks. O site LibreOffice tem uma aba Comunidade para a participação da comunidade. A evidência apresentada a Manutenibilidade foi a URL do projeto do Bugzilla. Para Usabilidade, o participante apresentou o URL de LibreOffice, que destaca as informações que explicam como novos usuários podem participar e interagir com os produtos do projeto. ({https://www.libreoffice.org/community/get-involved/)

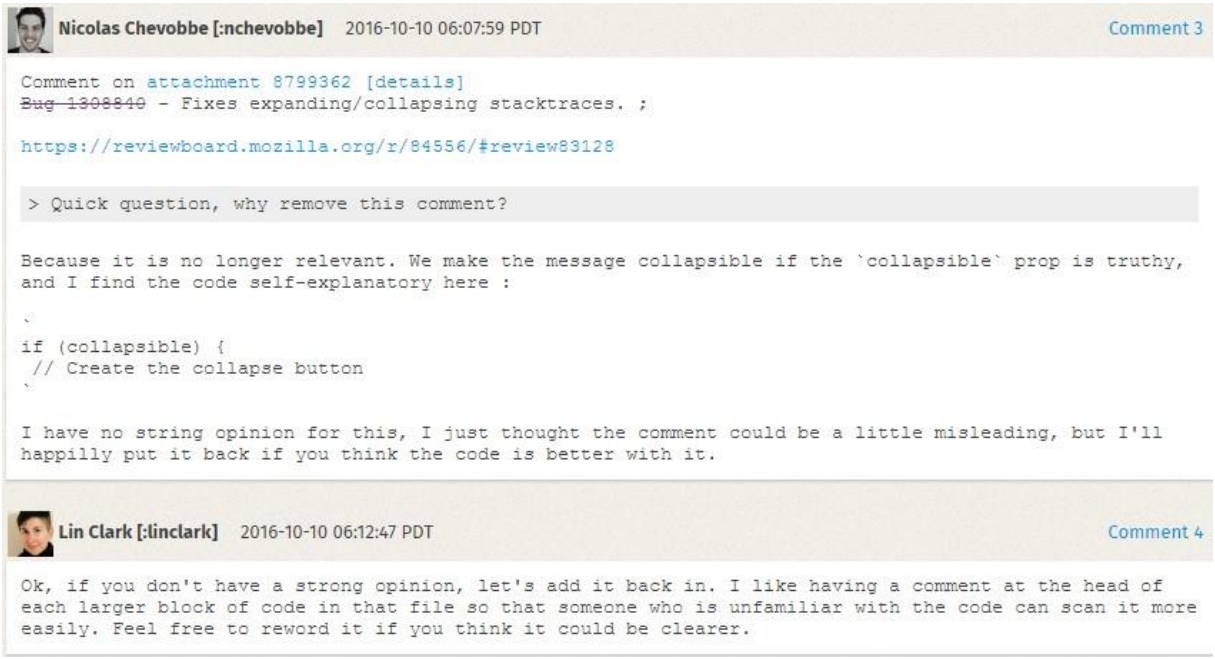
5.3.1.5 Participante 5

Para **Eclipse**, *Confiabilidade* e *Manutenibilidade* foram propostos. Como justificativa, "*Compatibilidade porque o Eclipse é um software de código aberto com foco no desenvolvimento de uma plataforma extensível para a criação de novos projetos. Manutenibilidade porque Eclipse fornece acesso aos bugs e melhorias propostos por seus usuários, e ao código de desenvolvimento*". A evidência apresentada para justificar Manutenibilidade no projeto Eclipse foi um ciclo de correção de bugs de alta prioridade. A figura 18 (https://bugs.eclipse.org/bugs/showbug.cgi?id=505535**i**) representa essa evidência apresentada pelo participante.

Figura 18 - Evidência de Manutenibilidade no Eclipse

Para **Mozilla Firefox**, *Confiabilidade* e *Manutenibilidade* foram propostos. Como justificativa “confiabilidade porque há um estudo especificamente para cada versão de release do software, cada versão de release é mantida separada das outras de acordo com a confiabilidade que os usuários têm os recursos pertencentes a cada versão, depois de terem sido testados". Manutenibilidade porque "Mozilla Firefox fornece uma infra - estrutura para o gerenciamento e correção de bugs de acordo com as versões do sistema". A evidência apresentada para justificar Manutenibilidade foi um problema que um desenvolvedor fazendo o ajuste remove o comentário do código Alegando que o código era auto-explicativo `` *Porque não é mais relevante. Eu acho que o código se explica por si mesmo*". No entanto, outro desenvolvedor relata que esta não é uma boa desculpa e pede a re-adição do comentário desde “Alguém que não está familiarizado com o código pode compreendê-lo mais facilmente”. A figura 19 (https://bugzilla.mozilla.org/showbug.cgi?id=1308840i) representa essa evidência apresentada pelo participante.

Figura 19 - Evidência de Manutenibilidade no Mozilla Firefox



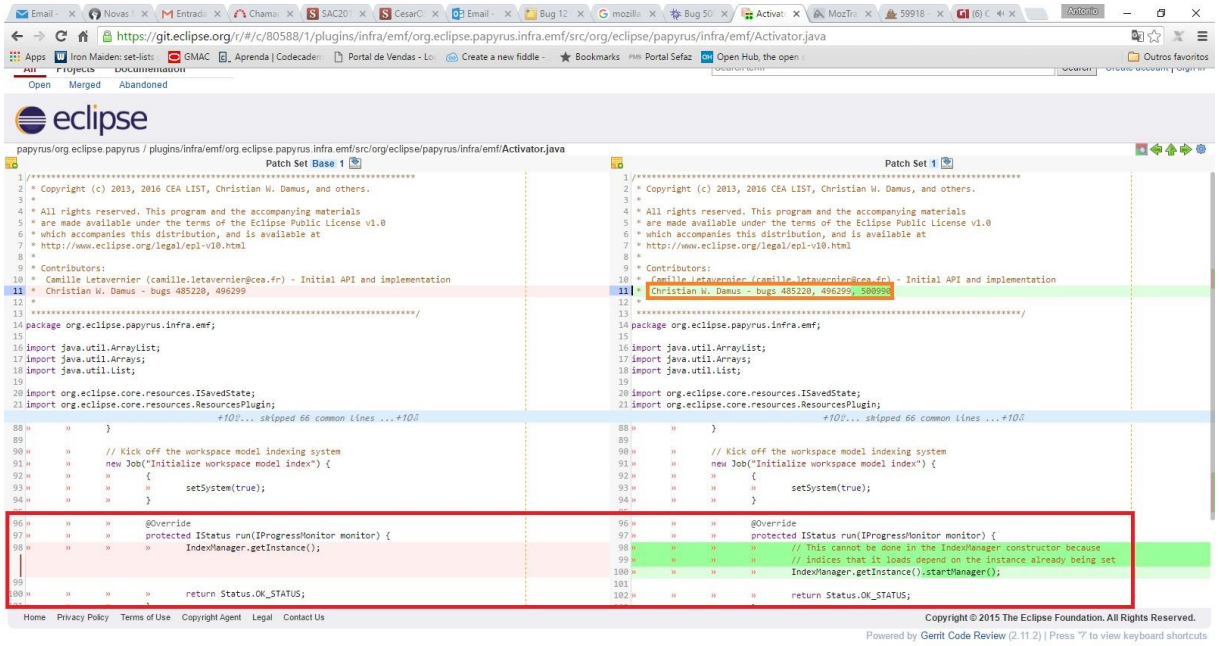
Para o projeto **LibreOffice**, *Manutenibilidade* e *Usabilidade* foram sugeridos como atributos de qualidade relevantes. A justificação afirma que o LibreOffice é "um projeto de software de código aberto que incentiva seus usuários, sejam eles desenvolvedores ou não, a testar o sistema para que eles contribuam para a melhoria da qualidade, fornecendo-lhes informações sobre seu uso" Manutenibilidade é justificada porque "o site oficial do LibreOffice fornece um ambiente propício à identificação de bugs de sistema e controle do

desenvolvimento de correções para esses bugs". A evidência citada sobre Usabilidade foi uma amostra dos URLs do projeto que incentivam os usuários a dar sua opinião sobre os programas Libre Office. Para Manutenibilidade, a URL do Bugzilla do projeto é apresentada, juntamente com as informações de que esta infra-estrutura descreve os bugs levantados pelos desenvolvedores, seu grau de gravidade e relatório de status para ajudar a monitorar cada correção de bugs. O site também convida os usuários a participar. (https://www.libreoffice.org/community/developers).

5.3.1.6 Participante 6

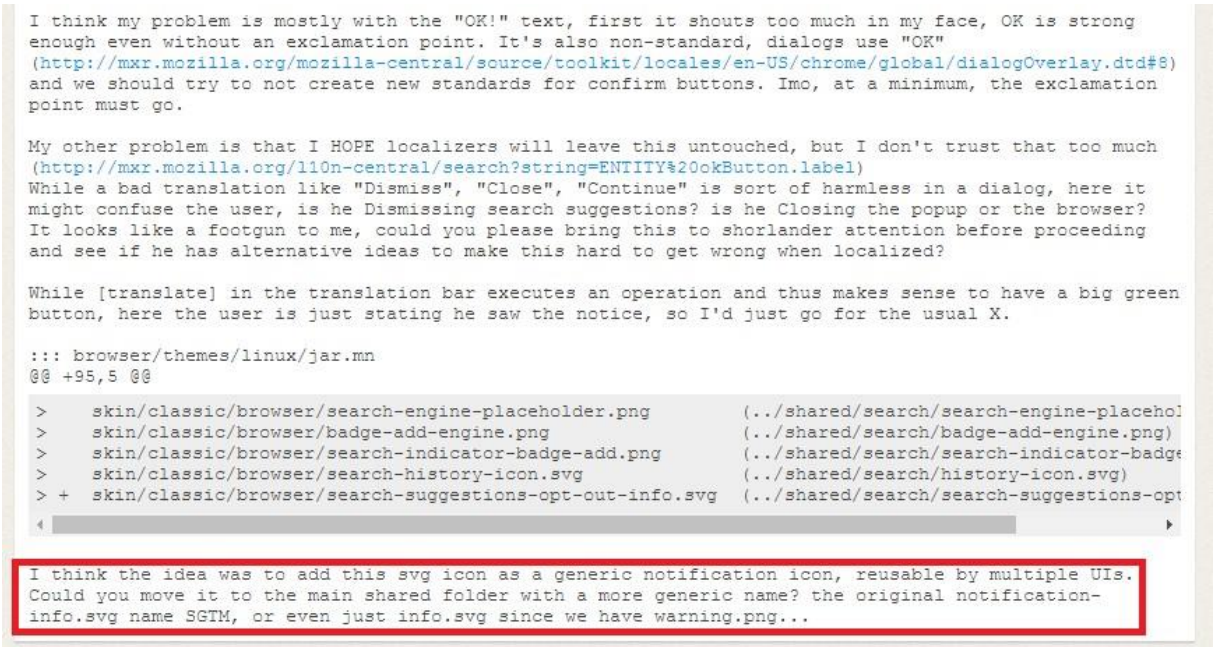
Para **Eclipse**, *Manutenibilidade* foi proposto. Como justificativa, porque "*O Projeto Eclipse é testado durante cada compilação por um extenso conjunto de testes automatizados. A maioria dos testes automatizados são utilizados para correção de programas, dessa forma a principal ferramenta para execução desses testes é JUnit, um framework que é utilizado para a construção de teste unitários. Caso algum erro seja encontrado ao executar esses testes unitarios um bug é gerado na ferramenta de gerenciamento de bug*". Evidenciando a testabilidade do projeto, umas das subcaracterísticas da Manutenibilidade. Além disso o eclipse incentiva a comunidade a testar as versões que são disponibilizadas e reportar os bugs encontrados na ferramenta de gerenciamento de bugs. Para evidenciar o participante apresentou o controle de modificação de código. A figura 20 (https://git.eclipse.org/r/#/C/80588/1/plugins/infra/emf/org.eclipse.payrus.infra.emf/src/org/eclipse/papyrus/infra/emf/Activator.javai) ilustra essa justificativa

Figura 20 - Evidência de Manutenibilidade no Eclipse



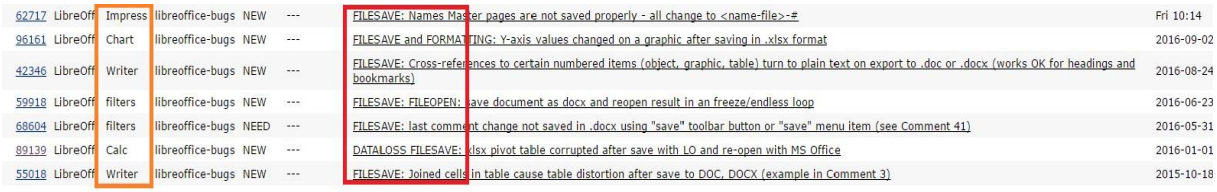
Para **Mozilla Firefox**, *Manutenibilidade* foi apresentado. Como justificativa, Manutenibilidade porque "*No Mozilla Firefox é observado uma grande importância a parte de teste, utilizando a ferramenta moztrap para o gerenciamento de testes manuais. Cada vez que é lançada uma novas funcionalidade no Firefox , o QA cria um conjunto de testes para isso. Estes testes são executados várias vezes antes da funcionalidade ser liberada*". O participante também aponta o uso de reusabilidade no projeto "Reusabilidade foi possível constatar quando um usuário solicita uma nova demanda, no caso a implementação de uma opção de busca por sugestão, ao implementar a demanda o usuário explica qual foram os procedimentos efetuados e, por fim, explica que está criando um ícone de notificação genérico, para ser reutilizado em outros projetos. A figura 21 (https://bugzilla.mozilla.org/show bug.cgi?id=959567**i**) representa essa evidência apresentada pelo participante.

Figura 21 - Evidência de Manutenibilidade no Mozilla Firefox



Para **LibreOffice**, *Manutenibilidade* foi apresentado . Como justificativa, porque "*LibreOffice disponibiliza compontentes que são utilizados em diversos outros produtos, ocasionando uma intensa reusabilidade de código.*" Foi possível identificar traços de reusabilidade no projeto Libre Office ao analisar uma demanda nova de um usuário, e a interação entre os integrantes do projeto para a implementação dessa nova demanda. A figura 22 (https://bugs.documentfoundation.org/buglist.cgi?quicksearch=filesavei) representa essa evidência apresentada pelo participante.

Figura 22 - Evidência de Manutenibilidade no LibreOffice

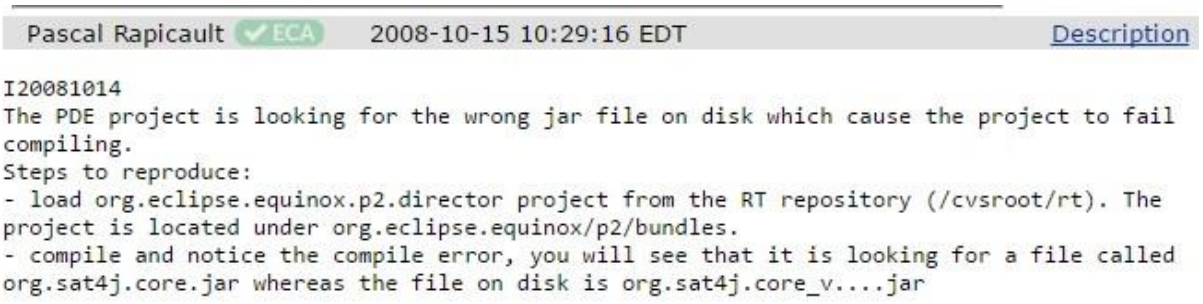


5.3.1.7 Participante 7

No **Eclipse** foram sugerido *Portabilidade*, *Compatibilidade*, *Funcionalidade* e *Usabilidade*. A justificativa utilizada pelo participante foi para: Portabilidade,"*O Eclipse é testado e validade em diversas platoformas", Compatibilidade "Eclipse 3.4 será compatível com Eclipse 3.3 (e, portanto, com 3.2, 3.1 e 3.0).", Usabilidade "Eclipse é bem estabelecido*

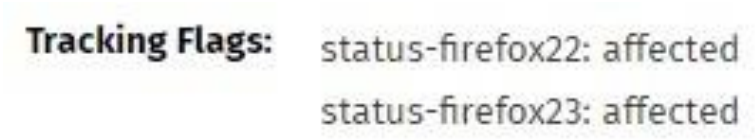
como o IDE multiplataforma de escolha, mas tornou-se muito mais do que isso. A extensa e diversificada gama de aplicações que estão sendo construídas com base no código Eclipse e as capacidades em constante mudança dos sistemas subjacentes nos quais ele é executado estão nos levando a empurrar os limites de nossa tecnologia em quase todas as dimensões. Esta área de trabalho marca o início de um novo foco multianual em inovação, para garantir que o Eclipse SDK continue a ser uma base vibrante, poderosa e dinâmica para o uso da nossa comunidade.", Funcionalidade "*Vamos continuar a investir em nossas ferramentas de desenvolvimento Java, investigando a melhor maneira de lidar com hardware futuro, linguagem Java e recursos IDE*". A principal evidência apresentada foi referente a um bug de Funcionalidade que causava o projeto falhar ao ser compilado, a figura 23 (hhttps://bugs.eclipse.org/bugs/showbug.cgi?id=250946i) apresenta essa situação.

Figura 23 - Evidência de Funcionalidade no Eclipse



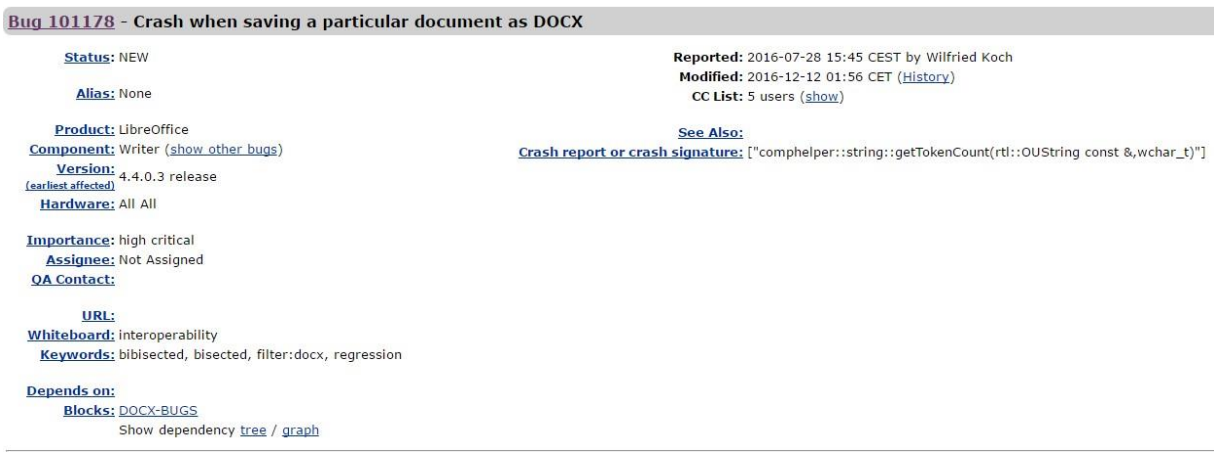
Para **Mozilla Firefox**, *Portabilidade*, *Compatibilidade* e *Segurança* foram apresentados. A justificativa utilizada pelo participante foi "*Uma vez que os problemas referente a instalação ou a compatibilidade pode impedir do usuário ter o primeiro contato com o produto. Por se tratar de um browser, a questão de segurança e privacidade é de grande importância para que esse produto atraia seus clientes*." A principal evidência apresentada por esse participante foi um bug relacionado a compatibilidade do produto, mostrando as versões que estarão sendo afetadas por essa correção. A figura 24 (https://bugzilla.mozilla.org/show bug.cgi?id=901314i) está retratando o exemplo do participante.

Figura 24 - Evidência de Compatibilidade no Mozilla Firefox



Para o **LibreOffice** foram sugeridos os seguintes atributos de qualidade: *Portabilidade*, *Compatibilidade*, *Funcionalidade* e *Usabilidade*. Para justificar a Portabilidade o participante cita que o "*Libre Office está disponível para os seguintes sistemas operacionais: Linux x64 (deb), Linux x64 (rpm), Linux x86 (deb), Linux x86 (rpm), Mac OS X x86-64 (10.8 or newer required), Windows, Windows x86-64 (Vista or newer required)"*. No caso da compatibilidade, "*Libre Office está disponível nas seguintes versões: 5.1.5, 5.1.6, 5.2.2"*. Funcionalidade foi justificado pelo seguinte comentario "*O LibreOffice é uma suite de escritório poderosa - sua interface limpa e ferramentas ricas em recursos ajudam a liberar sua criatividade e aumentar sua produtividade. O LibreOffice inclui várias aplicações que o tornam o mais poderoso pacote de escritório Open Source no mercado*". E finalmente Usabilidade "*O LibreOffice é projetado com grande atenção à acessibilidade, para tornar a suíte conveniente e confortável de usar, e para atender aos usuários com necessidades especiais. Você também pode ler outras informações de acessibilidade úteis em nosso wiki.*" A principal evidência apresentada pelo participante foi um bug referente a um erro no aplicativo ao tentar salvar um arquivo no formato DocX, conforme mostrado na figura 25 (https://bugs.documentfoundation.org/showbug.cgi?id=101178i).

Figura 25 - Evidência de Funcionalidade no LibreOffice

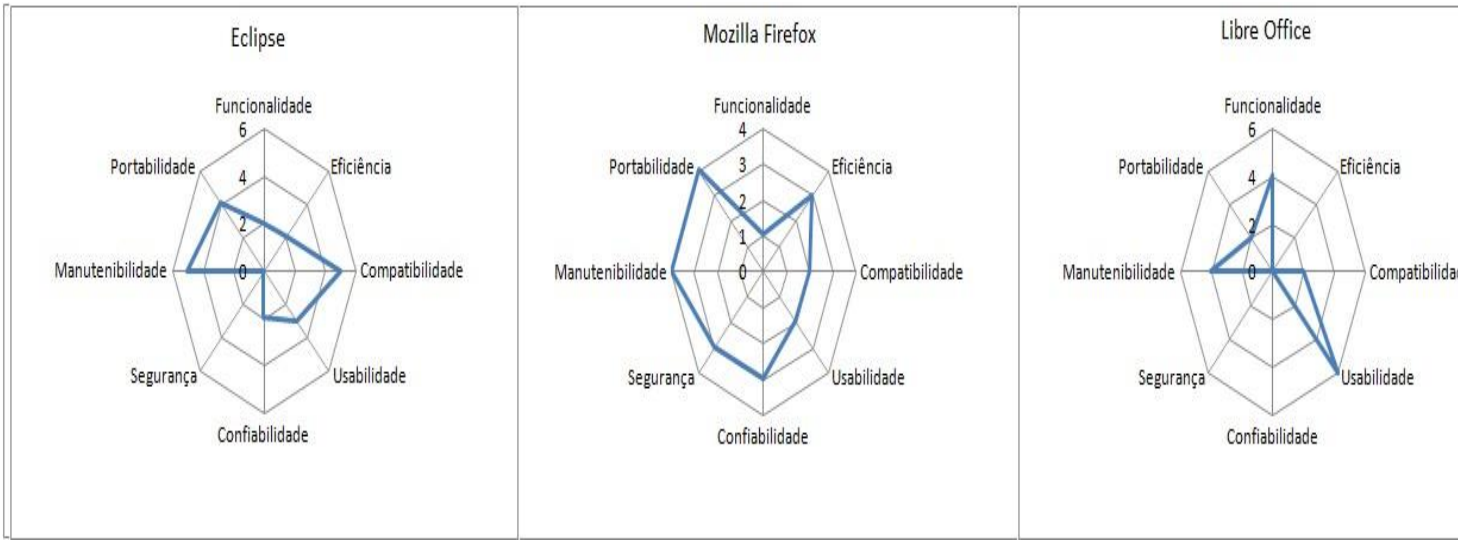


5.3.1.8 Análise dos Dados Fase 1

Figura 26 apresenta a quantidade de atributos indicados por cada participante deste estudo. 5 (71 %) participantes indicaram para o Eclipse os atributos *Compatibilidade* e

Manutenibilidade; *Portabilidade* foi indicado por 4 (57%) participantes. No caso do Mozilla Firefox, *Manutenibilidade* e *Portabilidade* foram indicados por 4 (57%) dos participantes, enquanto *Eficiência, Confiabilidade* e *Segurança* foram também indicadas para este projecto por 3 (42 \%) participantes. Finalmente, o LibreOffice tinha *Usabilidade* como o principal atributo de qualidade com 6 indicações. A razão possível para este resultado é a preocupação da comunidade em construir um produto amigável e intuitivo, bem como a concorrência com outros produtos similares. *Funcionalidade* e *Manutenibilidade* foram mencionados por 4 participantes. Foi possivel constatar que o LibreOffice tinha menos uniformidade entre os atributos indicados. Por exemplo, *Confiabilidade* e *Segurança* não foram mencionados e *Eficiência* e *Portabilidade* receberam apenas 1 voto cada.

Figura 26 – Resultados Fase 1



O quadro 9 apresenta a distribuição de demandas indicados pelos participantes com respectivos atributos de qualidade de produto de software associados para os projetos de destino.

Quadro 9 - Demandas Indicadas pelos Participantes da Fase 1

Atributo de Qualidade	**Eclipse**	**MozillaFireFox**	**LibreOffice**
Funcionalidade	1	0	5
Eficiência	3	2	1
Compatibilidade	2	1	1
Usabilidade	0	0	4
Confiabilidade	1	2	0
Seguranca	0	0	0
Manutenibilidade	2	2	0
Portabilidade	6	3	0

5.3.1.9 Estratégias Utilizadas pelos Participantes

Para destacar os atributos de qualidade selecionados, os participantes 1, 3, 5, 6 e 7 adotaram a estratégia de filtrar os problemas pelo *nível de severidade* juntamente com sua respectiva *prioridade*. Esses participantes justificaram o uso desta estratégia considerando que os projetos que adotam a abordagem de releases freqüentes precisam promover a seleção de demandas e novos recursos que possam ter um impacto significativo no produto de software. O Participante 4 adotou outra estratégia. Ele procurou atributos de qualidade relevantes analisando cada perfil de projeto de software e analisando dados nos portais dos projetos de software e wikis. No entanto, esta estratégia revelou-se pouco eficaz, considerando que apenas a seleção de atributos de qualidade através de portais e wikis não refletem necessariamente a relação desses atributos com o planejamento de release desses projetos. Finalmente, o participante 2 realizou uma análise quantitativa dos problemas encontrados nos sistemas de rastreamento de demandas dos projetos, considerando o tempo de resposta/solução para os componentes principais dos projetos com base nos atributos de qualidade escolhidos pelo participante.

5.3.2 Segunda Fase

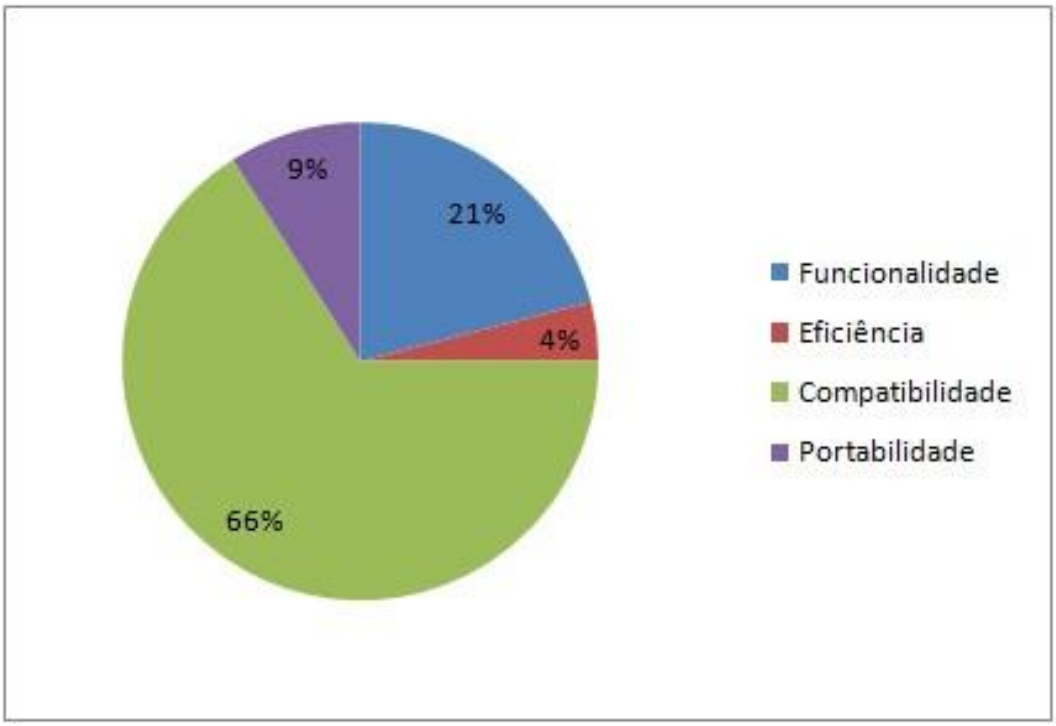
A segunda fase teve como objetivo identificar atributos de qualidade que foram identificadas em demandas que têm gravidade atribuída como *blocker* ou *critical* para comparar esses atributos com aqueles indicados pelos participantes na primeira fase. A razão para buscar demandas com esses tipos de gravidade é focalizar questões consideradas relevantes no contexto dos projetos analisados. Durante esta pesquisa, alvejou-se as mesmas versões/releases dos projetos indicados pelos participantes. Esta segunda fase compreendeu a seguinte questão de pesquisa: **QPEE3: Quais são os valores atribuídos pelos desenvolvedores aos campos "prioridade" e "gravidade" dos problemas registrados nos repositórios de bug dos projetos relacionados, aos atributos prioritários e não prioritários de qualidade de produto de software?**

5.3.2.1 Eclipse

A pesquisa das demandas do Eclipse considerou as versões 3.3.2, 3.4, 4.4, 4.5 e 4.6, bem como gravidade *blocker*. Como um todo, ele retornou 86 demandas. As questões foram analisadas uma a uma para ser caracterizada.

A figura 27 apresenta os resultados desta caracterização das demandas para o projeto Eclipse. Das 88 demandas, 44 foram associadas com algum atributo de qualidade, 66\% eram sobre problemas de *Compatibilidade*, com uma das versões anteriores ou com alguns dos pacotes coexistentes. O Eclipse é um grande projeto com vários pacotes coexistentes e muitos problemas foram relatados relacionados a esse problema. 21\% das questões levantadas relacionam-se com *Funcionalidade* e 9\% *Portabilidade*, principalmente na instalação. Apenas 4\% referem-se a *Eficiência*. Outros atributos não foram identificados para qualquer demanda. Por coincidência, todas as questões tinham o mesmo nível de prioridade.

Figura 27 – Atributos de Qualidade das Demandas do Eclipse

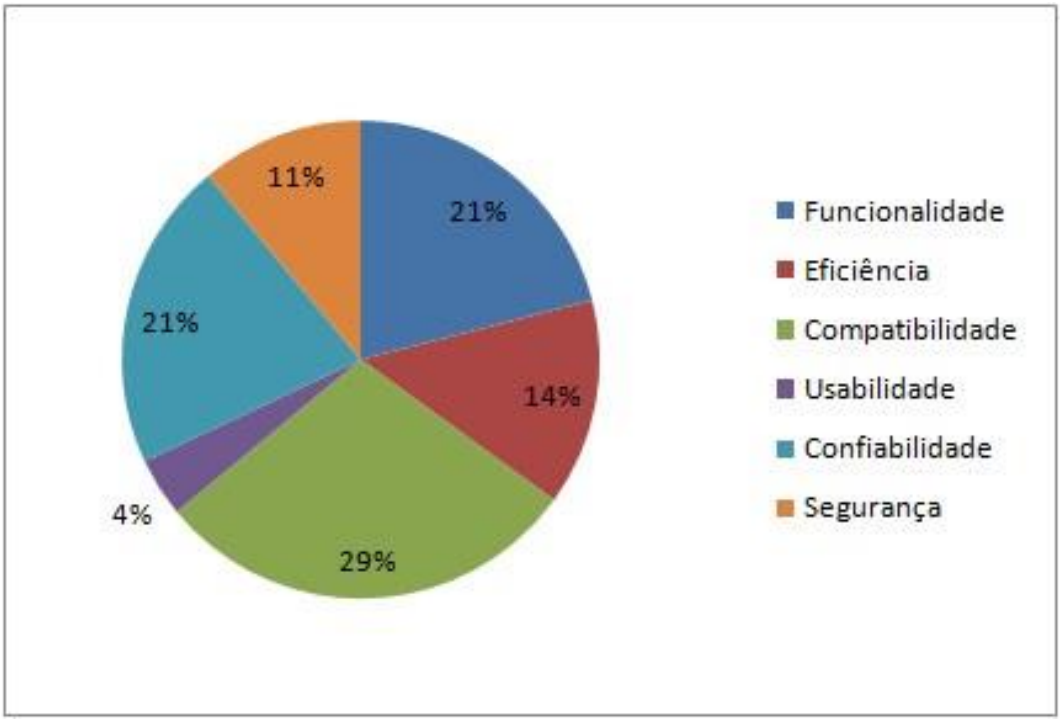


5.3.2.2 Mozilla Firefox

Para o Mozilla Firefox, foram consideradas as seguintes versões: 23, 42, 43, 45, 48 e 49. Além disso, as gravidades consideradas foram *blocker* e *critical*. Nestas configurações, no

total, foram encontradas 28 demandas e foi realizada uma análise individual para identificar qual o atributo de qualidade mais representativo. A figura 28 apresenta os resultados. A primeira percepção é que vários atributos foram identificados nesta amostra, conforme apresentado na tabela 3 na percepção dos participantes, quando eles também mencionam mais uniformemente todos os atributos de qualidade para o Firefox. Os atributos mencionados foram *Compatibilidade* 29%, *Funcionalidade* e *Confiabilidade* com 21%, *Eficiência* com 14%, *Segurança* e *Usabilidade* com 4%.

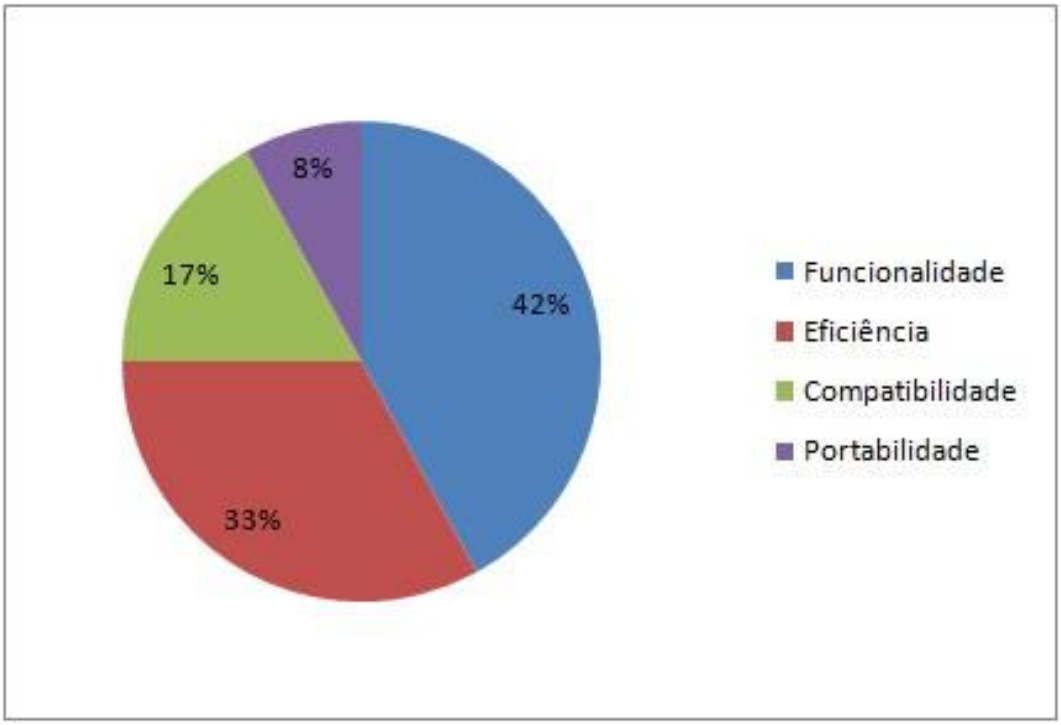
Figura 28 – Atributos de Qualidade das Demandas do Mozilla Firefox



5.3.2.3 LibreOffice

Finalmente, para o Libreoffice, foram consideradas as seguintes versões: 3.4.3, 4.0.0.3, 4.1.0.2, 4.2.0.3, 4.2.4.2, 5.0.0.5 e 5.0.3.2. Os níveis de gravidade considerados foram *blocker* e *critical*, dos quais 11 demandas foram encontradas. As demandas foram analisadas individualmente. Figura 29 apresenta os resultados desta caracterização para o Libre Office. Das 11 demandas retornados, 42% eram sobre *Funcionalidade*, 33% referem-se a *Eficiência*, 17% *Compatibilidade* e 8% referem-se a *Portabilidade*. Outros atributos não foram identificados para qualquer demanda.

Figura 29 – Atributos de Qualidade das Demandas do LibreOffice



5.3.3 Terceira Fase

A terceira fase consistiu em buscar o repositório de demandas, para verificar aqueles que não foram indicados pelos participantes, mas teve um alto grau de severidade nos três projetos e relacionado aos atributos de qualidade selecionados pelos participantes apontados na figura 26. Também verificamos as demandas implementadas/resolvidas em releases próximos ou distantes da data de registro. Desta forma, é possível verificar a consistência dos atributos selecionados e também verificar como foram priorizados no planejamento da release.

A estratégia adotada foi definir inicialmente um filtro que pudesse selecionar as demandas de maior prioridade e severidade no Bugzilla e que já estivessem finalizadas ou resolvidas. O passo seguinte foi ler a descrição da demanda. Se alguma evidência fosse encontrada, que apontasse para um determinado atributo de qualidade, essa demanda seria selecionada e todo seu fluxo seria analisado para confirmá-lo. Dessa foram seria possível detectar as relações de prioridade no planejamento da release, dependendo do atributo de qualidade indicado.

Finalmente, com esta fase, responde-se a questão de pesquisa **QPEE4: As demandas relacionadas aos atributos prioritários de qualidade dos produtos de software são entregues em lançamentos logo após serem registrados?**

5.3.3.1 Eclipse

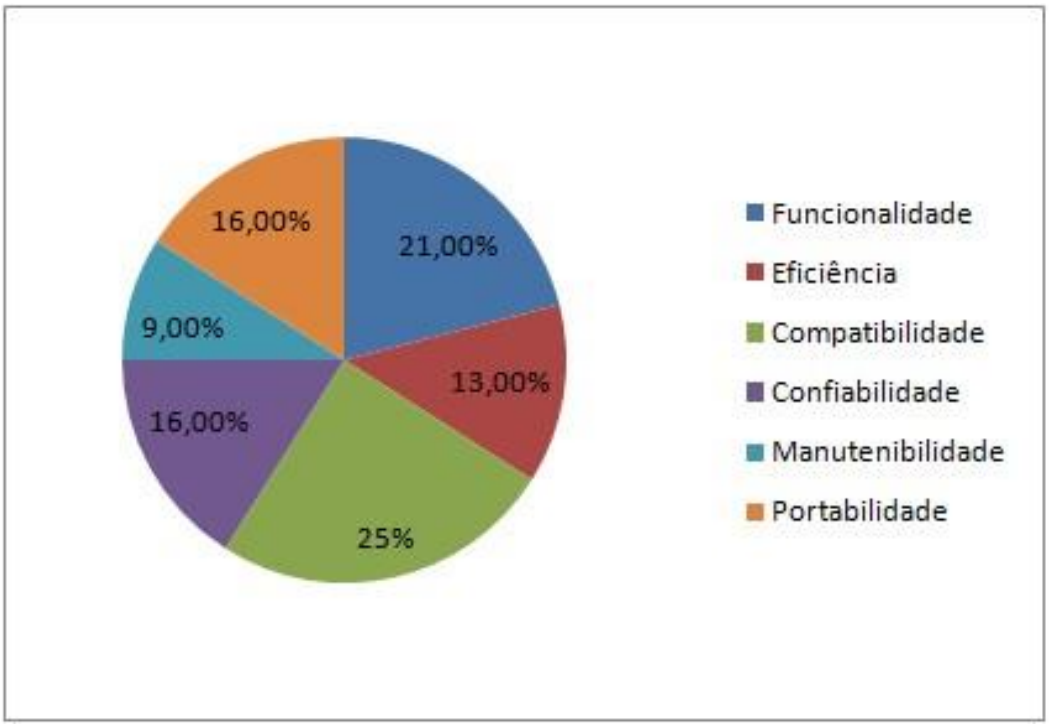
Para o Eclipse, foi definido o seguinte filtro: (*Status: Resolved, Closed; Resolution: Fixed; Priority: P1; Severity: Blocker, Critical; Classification: Eclipse)*. O foco foi sobre os seguintes atributos de qualidade: *Funcionalidade, Eficiência, Compatibilidade, Usabilidade, Confiabilidade, Manutenibilidade* e *Portabilidade*.

Esta consulta retornou uma lista de 162 demandas, que resultou em 24 demandas que tiveram algumas conexões com os atributos de qualidade selecionados. Um exemplo que se destacou foi a demanda 1310554 que inclui a seguinte descrição "*GTK3] Problema com a edição de tabela/árvore*". Essa demanda foi registada em 29 de Junho de 2014.

Esta descrição nos permitiu fazer uma conexão com a *funcionalidade* do produto. No entanto, ao analisar o relatório de erros para obter informações mais detalhadas, estabelecemos uma conexão com a *Compatibilidade,* porque o usuário apontou que ao usar uma versão do GTK (o toolkit multi-plataforma para criar interfaces gráficas de usuário) alguns comandos se comportariam de forma anormal e que com uma versão mais recente do GTK nenhum comando estaria funcionando conforme a descriçao a seguir *"A edição funciona corretamente com GTK2 GTK3 <= 3.8 - funciona, mas um menu de contexto não funciona.Clique com o botão direito do mouse em um campo de edição, Deixar o modo de edição e as ações (Cortar, Copiar, Colar ...) não funcionam GTK3> = 3.10 - você não pode entrar no modo de edição.* ".

Finalmente, a figura 30 mostra os atributos de qualidade distribuídos. Das 24 demandas retornadas, 21% foram sobre problemas relacionados a *Funcionalidade,* 13% foram sobre *Eficiência*, 25% sobre *Compatibilidade,* 16% sobre *Confiabilidade*, 9% sobre *Manutenibilidade* e 16% sobre *Portabilidade*.

Figura 30 – Atributos de Qualidade das Demandas do Eclipse



5.3.3.2 Mozilla Firefox

Para o Mozilla Firefox, o seguinte filtro foi definido para consultar os bugs: \(*Status: Resolved, Closed; Resolution: Fixed; Priority: P1; Severity: Blocker, Critical; Product: Firefox)*. O foco foi nos seguintes atributos de qualidade: *funcionalidade*, *eficiência*, *compatibilidade*, *usabilidade*, *manutenibilidade* e *ortabilidade*. Esta consulta retornou 83 demandas. A análise da lista levou a 9 demandas que tinham relação com os atributos de qualidade selecionados.

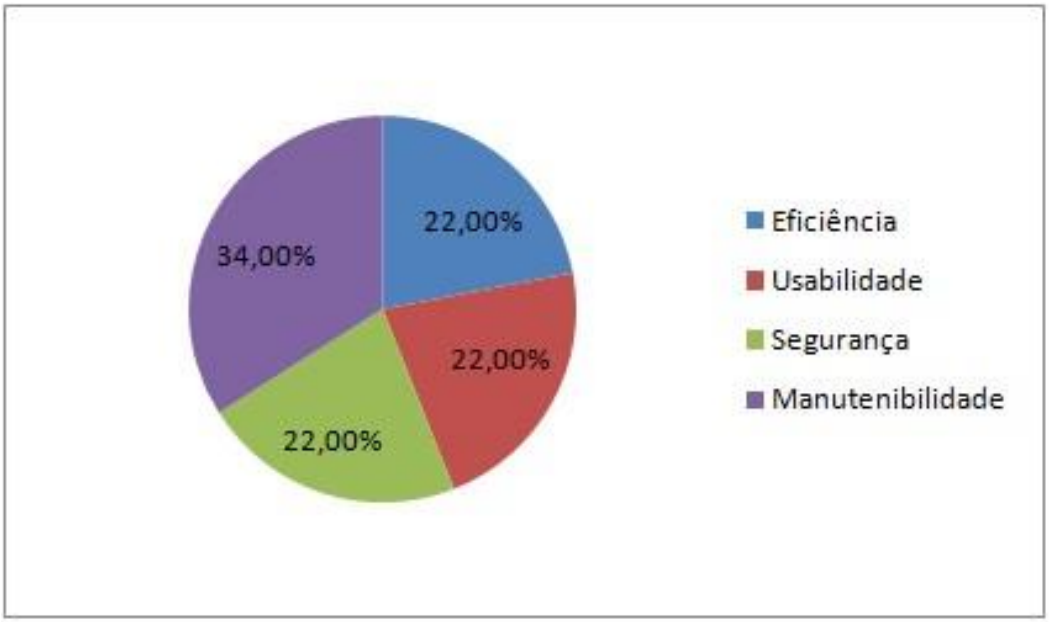
Um exemplo é o problema 1310554, que tem a seguinte descrição "*updateChildIndex em PlacesSyncUtils é extremamente ineficiente*" e foi registrado em 16 de outubro de 2016, permitindo a identificação de questões relacionadas à eficiência do produto. Ao analisar o relatório de erros para obter mais detalhes, encontramos uma conexão com o e*ficiência*, já que o usuário apontou que ter alguns marcadores dentro de uma pasta faz com que o Firefox faça uma chamada para cada marcador, cada chamada durando entre 5 e 10 segundos *Se você tem 10000 marcadores em uma única pasta, a aplicação desses registros leva muitas horas, durante as quais o Firefox está completamente sem resposta*." A instrução PlacesSyncUtils

mostra que isso é chamado 10000 vezes, com cada chamada demorando cerca de 5-10 segundos. .

O diálogo entre os usuários revela que, em 24 de outubro de 2016, foi realizada uma modificação para simplificar o modo como os *bookmarks* são sincronizados e que esta atualização estaria disponível nas versões atuais e subseqüentes.

Figura 31 mostra a distribuição entre os atributos. Das 9 demandas retornadas, 22% foram sobre problemas de e*ficiência*, 22% *usabilidade*, 22% *segurança* e 34% referem-se a *manutenibilibade*.

Figura 31 – Atributos de Qualidade das Demandas do Mozilla Firefox



5.3.3.3 LibreOffice

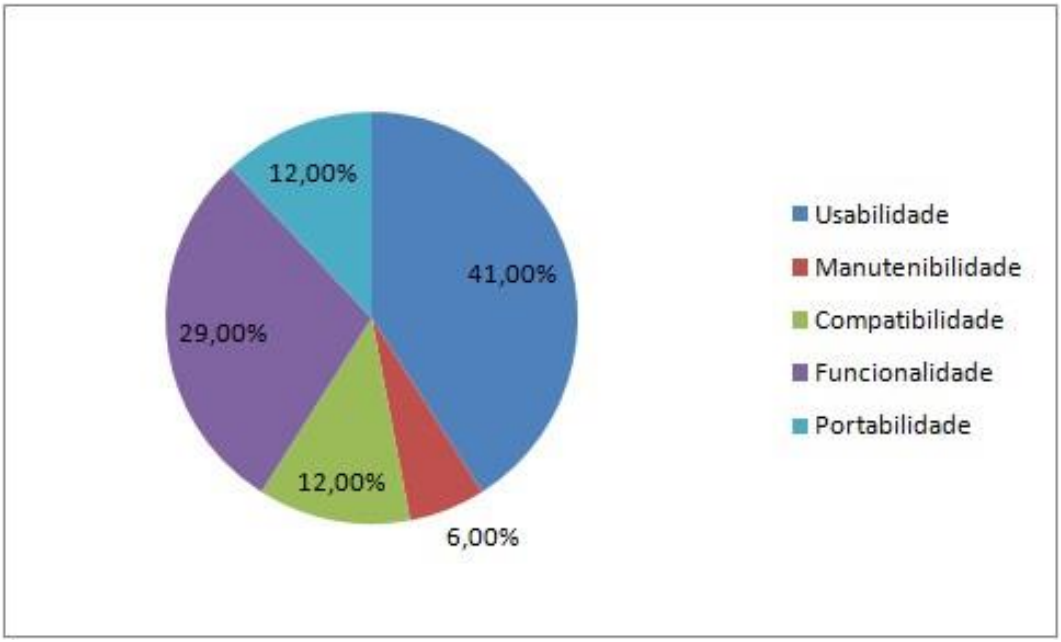
Para o LibreOffice, o seguinte filtro de bugs foi definido: *(Status:Resolved, Closed; Resolution:Fixed; Priority: Highest; Severity:Blocker, Critical)*. Centrando-se nos seguintes atributos de qualidade: *funcionalidade*, *compatibilidade*, *usabilidade*, *portabilidade* e *manutenibilidade*.

Como resultado desta consulta, foram encontradas 112 demandas. As descrições correspondentes foram lidas com foco nos atributos de qualidade sugeridos pelos participantes. Se o texto tivesse alguma seção que pudesse estar associada aos atributos de

qualidades indicadas, esta demanda seria analisada mais a fundo. Por exemplo, na demanda 84752 diz: "*Documento apresentando Formulários Inutilizáveis*". Esta seção chama a atenção para o termo "inutilizáveis" , que sugere uma relação com o atributo de u*sabilidade*. Quando acessamos o problema para obter mais informações, descobrimos que quando o usuário fez a atualização, a execução de alguns controles tornou-se muito mais lenta deixando o sistema quase inutilizável, conforme a seguinte descrição de um dos usuários: "*Atualizamos de 4.2 para 4.3 e descobrimos que documentos com controles de formulário são muito lentos para serem usados*.". Este problema expõe uma ligação com o atributo de *usabilidade*.

Finalmente, ao analisar a lista de demandas, foram detectadas 18 demandas que tinham uma relação com os atributos de qualidade selecionados. A figura 32 apresenta como ficou a distribuição entre atributos. A partir dos problemas selecionados 41% foram sobre *usabilidade*, 29% *funcionalidade*, 12% *compatibilidade*, 12% *portabilidade* e 6% referem-se a *manutenibilidade*.

Figura 32 – Atributos de Qualidade das Demandas do LibreOffice



5.3.4 Demandas Entregues em Releases

Por fim, foi realizada a análise baseada nas demandas indicadas nas fases 1 e 3. Nessa análise, foi levado em consideração se as demandas foram implementadas/resolvidas nas

seguintes versões relativas à data de registro. Desta forma, é possivel encontrar alguma relação de prioridade no planejamento da release de acordo com o atributo de qualidade indicado. A estratégia adotada nesta etapa foi encontrar através das informações fornecidas nas demandas a versão que foi encontrada o erro/ajuste e também em qual versão este foi disponibilizado. Com essas informações, acessamos o portal do projeto para encontrar informações sobre releases dos projetos e identificamos quantas versões foram implantadas até que a correção dos problemas fosse disponibilizada. Este fator está sendo chamado de intervalo entre releases.

O quadro 10 apresenta um resumo dos resultados das demandas na fase 1. O quadro 11 apresenta as demandas da fase 3, essas tabelas mostram os atributos de qualidade, o id da demanda e intervalo entre a versão disponibilizada e a correção correspondente feita. Apesar de apresentar uma lista com algumas demandas, não foi possível identificar uma determinada priorização de atributos de qualidades de produto em relação a entrega de releases proximas ao registro da demanda, nos projetos selecionados.

Quadro 10 - Súmario de Demandas Resolvidas na Fase 1

		Atributode Qualidade	ID	Intervalo de Releases
Fase1	Eclipse	Portabilidade	290182	1(4.6.1-4.6.2)
		Portabilidade	187062	6(4.4.0-4.6.0)
		Portabilidade	498196	1(4.6-4.6.1)
		Compatibilidade	228109	2(3.3.2-3.4)
		Compatibilidade	222885	5(3.4-3.5M2)
		Portabilidade	234307	2(3.4-3.4RC2)
		Portabilidade	232641	2(3.4-3.4RC2)
		Funcionalidade	250946	5(3.4-3.5M2)
		Eficiência	234718	2(3.4-3.4RC2)
	Mozilla Firefox	Portabilidade	1242901	9(43-52)
		Eficiência	1260850	6(45-51)
		Portabilidade	1287823	4(48-52)
		Manutenibilidade	1308840	3(49-52)
		Manutenibilidade	959567	1(42-43)
	LibreOffice	Funcionalidade	74104	1(4.2.0-4.2.1)
		Funcionalidade	78801	2(4.2.4-4.2.6)
		Usabilidade	95709	13(5.0.3-5.2.0)
		Usabilidade	41169	34(3.4.3-4.2.0)
		Funcionalidade	62038	19(4.0.0-4.2.5)
		Usabilidade	66924	1(4.1.0-4.1.1)
		Usabilidade	96742	0(5.2.0-5.2.0)

Quadro 11 - Súmario de Demandas Resolvidas na fase 3

		Atributo de Qualidade	ID	Intervalo de Release
Fase3	Eclipse	Compatibilidade	308364	5(1.0-1.0M5)
		Compatibilidade	363334	6(4.2-4.2M6)
		Compatibilidade	74341	2(3.0-3.1)
		Compatibilidade	438505	5(4.4-4.5RC2)
		Compatibilidade	28727	4(2.1-2.1M4)
		Funcionalidade	479435	12(4.5-4.6M4)
		Funcionalidade	294650	20(3.5.1-3.7M7)
		Funcionalidade	64540	1(3.0-3.0RC1)
		Funcionalidade	63296	1(3.0-3.0RC1)
		Funcionalidade	14291	5(2.0-2.0M5)
		Confiabilidade	335263	1(3.6.1-3.6.2)
		Confiabilidade	97148	2(3.1-3.1RC2)
		Confiabilidade	16208	6(2.0-2.0M6)
		Portabilidade	184433	1(3.3-3.3RC1)
		Eficiência	129979	6(3.2-3.2M6)
		Eficiência	52982	7(3.0-3.0M7)
		Eficiência	27264	4(2.1-2.1M4)
		Manutenibilidade	270744	7(3.5-3.5M7)
		Manutenibilidade	179210	2(3.3-3.3RC2)
	Mozilla Firefox	Manutenibilidade	1261651	0(48-48)
		Manutenibilidade	1211016	0(42-42)
		Eficiência	1310554	0(51-51)
		Eficiência	1303611	1(51-52)
		Segurança	1215885	0(44-44)
	LibreOffice	Usabilidade	84752	3(4.3.0-4.3.3)
		Usabilidade	72647	0(4.2.0-4.2.0)
		Usabilidade	73464	1(4.2.0-4.2.1)
		Usabilidade	89873	0(4.5.0-4.5.0)
		Usabilidade	47368	6(3.5.1-3.6.0)
		Usabilidade	73243	7(4.2.0-4.3.0)
		Usabilidade	72741	13(4.1.0-4.2.5)
		Funcionalidade	43868	0(3.4.5-3.4.5)
		Funcionalidade	55493	4(3.6.0-3.6.4)
		Funcionalidade	76949	1(4.2.3-4.2.4)
		Funcionalidade	36982	6(3.4.0-3.4.6)
		Compatibilidade	38452	7(3.4.0-3.5.0)
		Compatibilidade	64490	30(3.5.4-4.4)
		Portabilidade	69517	0(4.2.0-4.2.0)
		Portabilidade	92041	0(5.0.0-5.0.0)

5.3.5 Comparação dos resultados

Para o Eclipse, *Compatibilidade* foi o atributo de qualidade que mais se destacou. Especialmente na fase 2 do estudo, uma justificativa possível é devido a ser uma ferramenta que integra várias outras ferramentas, e roda em várias versões de vários sistemas operacionais, o Eclipse e os projetos criados em torno dele devem ser compatíveis e capazes de se adaptar a infra-estrutura heterogênea (hardware, sistemas operacionais, versões de sistemas, etc.). Além disso, *Eficiência* apresentou resultados semelhantes nas 3 fases do estudo.

No Mozilla Firefox, os atributos com a maioria das demandas levantados foram *Compatibilidade* e *Eficiência*. No entanto, enquanto *Compatibilidade* teve um número significativo de demandas na fase 2, *Eficiência* apresentou resultados mais homogêneos nas três fases. Isso pode ser justificado pelo fato de que o Firefox é um navegador Web, deve ser compatível com diferentes tecnologias usadas em portais e ser capaz de ser instalado em vários sistemas operacionais (*Compatibilidade*). Tudo isso deve ocorrer de forma transparente para o usuário, sem problemas de lentidão ou inoperabilidade. O desempenho do programa é fundamental, porque a concorrência no mercado é feroz no que diz respeito ao desempenho (*Eficiência*).

Para o LibreOffice, *Funcionalidade* foi o atributo indicado com mais freqüência. Está presente em todas as três fases. *Usabilidade* também se destaca, embora nenhuma demanda tenha sido encontrado na segunda fase. Uma justificativa que pode explicar a proeminência relativa desses atributos é que o Libre Office é fortemente orientado para o usuário, além de ser um conjunto de escritórios, que exige características intrínsecas ao seu domínio de aplicação. Portanto, ele precisa fornecer um número significativo de funcionalidades, de forma a satisfazer as expectativas (exigentes) de seus usuários. Além disso, deve ser facilmente abordado por usuários não-técnicos e ter uma interface gráfica atraente e moderna, de modo que seu uso torna-se intuitivo.

5.4 CONCLUSÃO

Esse estudo exploratório serviu para investigar a influência de atributos de qualidade de produto alvo em práticas de release de software de Projetos OSS. Nós alvejamos quatro

questões de pesquisa para encontrar provas que mostrem as relações entre os atributos de qualidade, as prioridades atribuídas as demandas registradas e como eles são entregues por release de produtos. Com esta iniciativa pretendemos entender como os projetos OSS atuam para alinhar suas metas com as necessidades de seus usuários. Foi possível constatar que a depender do perfil do projeto, um atributo de qualidade tem mais importância do que um outro determinado atributo. Um exemplo foi a constatação do atributo de usabilidade no Libre Office, que foi apontado praticamente em todas as fases do estudo e é justificavel por ser um projeto que tem uma forte concorrência de outros produtos de suite, inclusive de projetos proprietários, e por isso precisa fornecer uma interface amigável, que proporcione um nível de satisfação para o usuário.

5.4.1 Ameaças de Validação

Neste estudo foram estabelecidas três limitações. A primeira pode ser atribuído ao número reduzido de usuários que participaram do estudo. Os resultados da amostra podem ter sido influenciados, pois apenas sete voluntários participaram do estudo. Foi apresentado um tutorial com um projeto de teste, com vista a reduzir este risco. A segunda ameaça era uma possível má interpretação da atividade solicitada. Para minimizar esta ameaça, o autor estive disponível para responder a quaisquer dúvidas que possam ter surgidos durante a execução desta atividade. A última ameaça diz respeito à identificação de atributos e problemas de qualidade de produto do Eclipse, Mozilla Firefox e do LibreOffice. Para mitigar esse risco, todas as demandas encontradas foram discutidas para tornar mais confiáveis os atributos de qualidade caracterizados para os projetos selecionados.

Este capítulo apresentou o planejamento, elaboração, execução e resultados de um estudo exploratório realizado nesta dissertação. A seguir será apresentada a conclusão deste trabalho.

6 CONCLUSÃO E PERSPECTIVAS FUTURAS

Este capítulo apresenta as considerações finais, contribuições, e as perspectivas de trabalhos futuros.

6.1 CONSIDERAÇÕES FINAIS

Essa dissertação apresentou um estudo com o objetivo de investigar a influência de atributos de qualidade de produto de software na priorização das atividades em projetos que adotaram práticas de releases rápidas. Para esta finalidade, foi realizada uma Revisão Sistemática de Literatura e um estudo exploratório com o objetivo de responder as questões de pesquisa estabelecidas nesta dissertação. Na Revisão Sistemática de Literatura (Capítulo 4) foi possível identificar, classificar e comparar evidências existentes de escalas de tempos utilizadas, estratégias, motivações, vantagens e desafios na adoção de práticas de releases rápidas em projetos open source. Já no estudo exploratório foi possível verificar a prevalência de determinados atributos de qualidade de produto em relação aos projetos Eclipse, Mozilla Firefox e Libre Office.

No total foram selecionados 30 estudos que foram evidências utilizadas para responder QPD1 (Quais os estudos mais relevantes publicados na literatura que abordam planejamento rápido de releases em projetos open source?).

Para responder QPD2 (Quais as principais estratégias e motivações que levam projetos open source adotarem releases frequentes?) foi possível observar os projetos adotaram principalmente as releases baseada em tempo em detrimento as releases baseada em funcionalidades, além disso foi necessário a adoção de ferramentas que automatizem o processo e foco em testes e em relação a motivação foi possível identificar a necessidade de aumentar o número de participantes, a atratividade do projeto e o aumento da participação no mercado.

Para responder QPD3 (Quais as principais vantagens e dificuldades na adoção de releases frequentes em projeto open source?) As principais vantagens observadas na adoção de releases rápidas foram: um *Time-to-Market* mais curto, maior qualidade, eficiência, feedback e satisfação do cliente, testes mais efetivos, entrada de novos colaboradores nos projetos, inovação, monitoramento mais efetivo e planejado, inclusão de novas

funcionalidades, concerto de bugs e atualizações de segurança. Os desafios foram caracterizados como: pressão de tempo, aumento da divida técnica, uma dependência maior da comunidade que suporta o projeto e a confiabilidade do produto.

Para responder QPD4 (Quais atributos de qualidade são priorizados em projetos que adotaram releases frequentes?) Foram utilizadas 3 fases em um estudo exploratório com o objetivo de identificar em 3 projetos open source conhecidos e que utilizam releases rápidas, quais atributos de qualidade foram priorizados. Dessa forma foi possível constatar que essa priorização é bastante influenciada pelo perfil e objetivo do projeto.

Para responder QPD5 (Quais as influências que determinados atributos de qualidade exercem, na priorização das atividades no planejamento das releases frequentes, em projetos open source?). Foram analisadas as demandas que possuissem um alto grau de severidade e urgência e estivessem concluídas, a fim de identificar se houve implantanção desses ajustes/correções em releases proximas a data de registro da demanda. No entanto, não foi possível concluir se um determinado atributo de qualidade possui prioridade no planejamento das releases desses projetos.

6.2 CONTRIBUIÇÕES

A principal contribuição deste trabalho foi o compartilhamento das informações apresentadas nos resultados da revisão sistemática e do estudo exploratório para apoiar a tomada de decisão no planejamento de estudos de caso relacionados à planejamento de releases e atributos de qualidade de produto.

Foi possível perceber uma tendência dos projetos open source em utilizar releases baseada em tempo, uma vez que essa estratégia beneficia o modelo de desenvolvimento utilizado por esses projetos, no que diz respeito a interação com a comunidade, concorrência com projetos proprietários e a satisfação do cliente.

Também percebeu-se um tendência no planejamento da releases de alguns projetos open source na adoção de releases rápidas, uma que vez é possível obter diversas vantagens, que foram apresentadas na revisão sistemática. Entre os projetos que podem ser destacados que utilizam essa abordagem estão: Google Chrome, Mozilla Firefox, Apache Coocoon, Apache Tomcat, Eclipse e o Libre Office.

Atráves das pesquisas nos repositórios de dados de determinados projetos open source foi possível observar o grau de maturidade e qualidade que esses produtos conseguem atingir, levando em consideração que a grande maioria dos colaboradores são voluntários das comunidades desses projetos.

6.3 TRABALHOS EM ANDAMENTO E FUTUROS

Neste momento, está sendo planejado o desenvolvimento de uma nova atividade exploratória, com o objetivo de obter demandas de forma automatizada a partir de determinados critérios. O objetivo é disponibilizar uma amostragem maior de dados, possibilitando uma melhor acurácia dos resultados.

Também estão sendo planejadas próximas etapas para a revisão sistemática. Estas etapas são relacionadas a seguir:

a) O mesmo modelo de pesquisa pode ser repetido, a fim de ampliar o escopo do estudo com a adição de artigos publicados após janeiro de 2016 e encontrar novas motivações, estratégias, vantagens e desvantagens.

b) Buscar novas contribuições a partir da utilização de snowballing.

REFERÊNCIAS

ADAMS, B.et al. An empirical study of integration nactivities indistributions of open source software. **Empirical Software Engineering**, Springer, v.21, n.3, p.960 – 1001, 2016.

BIJLSMA, D. et al. Faster issue resolution with higher technical quality of software. **Software Quality Journal**, Springer, v.20, n.2, p.265 – 285, 2012.

BRERETON, P.etal. Lessons from applying the systematic literature review process with in the software engineering domain. **Journal of systems and software**, Elsevier, v. 80, n.4, p.571–583,2007.

CHEN, L. ; BABAR, M. A. A systematic review of evaluation of variability management approaches in software product lines. **Information and Software Technology**, Elsevier, v.53, n.4, p.344–362, 2011.

COSTA, D.A.da et al. An empirical study of delays in the integration of addressed issues. In: ICSME, [S.l.]: [s.n.], 2014.p.281–290.

CROWSTON, K.; ANNABI,H.; HOWISON, J. Defining open source software project success. In: ICIS2003, 2003. **Proceedings…** 2003.

DYBA,T.;DINGSOYR,T. Empirical studies of agile software development: A systematic review. **Information and SoftwareTechnology,** v.50, n.9-10, p.833–859, 2008. ISSN 09505849.

FITZGERALD, B. The transformation of open source software. **MisQuarterly, JSTOR**, p.587–598, 2006.

FOGEL, K. **Producing open source software:** how to run a successful free software project. [S.l.]: O'ReillyMedia,Inc., 2005.

FUNDATION, E. **The open source developer report.** [S.l.]: Eclipse Community Survey, 2009.

GAMALIELSSON, J. ;LUNDELL, B. Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved? **Journal of Systemsand Software**, Elsevier, v.89, p.128–145, 2014.

GONZALEZ-BARAHONA, J. M .et al. Understanding how companies interact with free software communities. **IEEE software,** n.5, p.38–45, 2013.

GONZALEZ-BARAHONA, J. M. ; ROBLES, G. Trends in free, libre,open source software communities: From volunteers to companies. **It–InformationTechnologyit–Information Technology,** v.55, n.5, p.173–180, 2013.

HENNINGSSON, K.; WOHLIN, C. Understanding the relations between software quality attributes – a survey approach. In: CITESEER. INTERNATIONALCONFERENCE, 12., 2002. **Proceedings…** 2002.

ISO, I. Iec25010:2011. **Systems and software engineering:** Systems and Software Quality Requirements and Evaluation (SQuaRE). System and Software Quality Models. [S.l.], 2011.

ISO/IEC.ISO/IEC9126. **Software engineering:** product quality. [S.l.]: ISO/IEC, 2001.

JONSSON, L.et al. Automated bug assignment: Ensemble-based machine learning in large scale in dustrial contexts. **Empirical Software Engineering**, Springer, p.1–46, 2015.

KHOMH, F. et al. Do faster releases improve software quality? An empirical case study of mozilla firefox. In: IEEE. MINING SOFTWARE REPOSITORIES (MSR), IEEE WORKING CONFERENCE ON. 2012, 9., 2012. **Proceedings…** 2012. p.179–188.

KITCHENHAM,B.;CHARTERS,S. Guidelines for performing systematic literature reviews in software engineering. In: TECHNICAL report, Ver. 2.3. EBSE Technical Report. EBSE. [S.l.] : [s.n.], 2007.

MATOS, J. P. **Avaliação de atributos de qualidade em ambiente de software livre e aberto.** [S.l.] : [s.n.], 2012.

MICHLMAYR, M. ; FITZGERALD, B. Time-based release management in free and open source(foss) projects. **International Journalof Open Source Software and Processes (IJOSSP),** IGIGlobal, v.4, n.1, p.1–19, 2012.

MICHLMAYR, M.; FITZGERALD, B.; STOL, K.-J. Why and how should open source projects adopt time-based releases? **IEEE Software**, IEEE, v.32, n.2 ,p.55–63,2015.

OFFUTT, J. Quality attributes of web software applications. **IEEE software,** IEEE Computer Society, v.19, n.2, p.25, 2002.

OREG, S. ; NOV, O. Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. **Computers inhuman behavior**, Elsevier, v.24, n.5, p.2055–2073, 2008.

PERENS, B.et al. **The open source definition.** Open sources: voices from the open source revolution. Sebastopol, CA: O'Reilly,v.1,1999. p.171–188.

PEREPLETCHIKOV, M.; RYAN, C.;TARI, Z. The impact of software development strategies on project and structural software attributes in soa. In: SPRINGER .OTM CONFEDERATED INTERNATIONAL CONFERENCES "ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS, 2005. **Proceedings...** 2005. p.442–451.

PFLEEGER,S. L. Experimental design and analysis in software engineering. **Annals of Software Engineering,** Springer, v.1, n.1, p.219–253, 1995.

RIGBY, P. C.et al. Contemporary peer review in action: Lessons from open source development. **Software IEEE**, v.29, n.6, p.56–61, 2012.

RUHE, G. **Product release planning:** methods, tools and applications. [S.l.]: CRCPress,

2010.

STOL, K.-J.; FITZGERALD, B. Inner source – adopting open source development practices in organizations: A tutorial. **IEEESoftware,** n.4, p.60–67, 2015.

THOMAS, L.et al. Impact of release intervals on empirical research into software evolution, with application to the maintainability of linux. **IET software,** v.3, n.1, p.58–66, 2009.

WOHLIN, C.et al. **Experimentation ins oftware engineering**. [S.l.]: Springer Science & Business Media, 2012.

YE, Y.; KISHIDA, K. Toward an understanding of the motivation of open source software developers. In: IEEE SOFTWARE ENGINEERING, INTERNATIONAL CONFERENCE, 25., 2003. **Proceedings...** 2003. p.419–429.

YIN,R.K.;CAMPBELL, D. **Case study research:** design and methods. Applied Social Science Research Methods Series. [S.l.]:SagePublications,ThousandOaks, 2003. v.5.

ZHANG, H.; BABAR, M. A.; TELL, P. Identifying relevant studies in software engineering. **Information and Software Technology**, Elsevier, v.53, n.6, -p.625–637, 2011.

APÊNDICE A – LISTA DE ARTIGOS SELECIONADOS NA REVISÃO SISTEMÁTICA

ID	Autor, Título	Local de Publicação	Ano
S1	Olga Baysal, Ian Davisand, Michael Godfrey, A tale of two browsers.	ICSE	2011
S2	Puneet Agarwal, Continuous SCRUM: agile management Of SAAS products.	ISEC	2011
S3	Khomh,F., Dhaliwal T.,Ying Zouand Adams, B., Do, Faster Release Improves of tware Quality? An Empirical Case Study of Mozilla Firefox.	MSR	2012
S4	Nou red dine Kerzaziand Foutse Khomh, Factors impacting rapid releases:an industrial case study.	ESEM	2014
S5	Thierry Lavoieand Ettore Merlo.How much really changes? A case study of fire fox version evolution using a clone detector.	ICSE	2013
S6	SM Didar Al Alam, SM Shahnewaz, Dietmar Pfahl, Guenther Ruhe.Introduction of continuous delivery in multi-customer project courses	ICSE	2014
S7	Sandy Clark, Michael Collis, Matt Blazeand Jonathan M.Smith, Moving Targets: Security and Rapid-Release in Firefox.	CCS	2014
S8	JoeF.,Shobe, Md Yasser Karim, Motahareh Bahrami and Huzefa Kagdi Onmapping releases to commitsin open source systems.	ICSE	2014
S9	Anil Shankar, Honray Lin, Hans-Frederick Brownand Colson Rice, Rapid Usability Assessment of an Enterprise Application inan Agile Environmentwith Cog Tool.	CHI	2015

--*

ID	Autor, Título	Local de Publicação	Ano
S10	Johannes Wettinger, Uwe Breitenbucherand Frank Leymann Standards-Based Dev Ops Automation and integration Using TOSCA.	UCC	2014
S11	Mehrdad Nurolahzade, Seyed Mehdi Nasehi,Shahedul Huq Khandkar and,Shreya Rawal,The role of patch review in software evolution:ananalysis of the mozilla firefox.	FSE	2009
S12	Daniel Alencar da Costa, Surafel Lemma Abebe, Shane Mcintosh, Uira Kuleszaand Ahmed E.Hassan, Na Empirical Study of Delay sin the Integration of Addressed Issues.	ICSME	2014
S13	Souza,R.,Chavez,C.and Bittencourt, R.A.Do Rapid Releases Affect Bug Reopening? A Case Study of Firefox.	SBES	2014
S14	Conley, C.A.and Sproull,L.Easier,Said than Done: Na Empirical Investigation of Software Design and Quality in OpenSource Software Development.	HICSS	2009
S15	Thomas, L.G., Schach,S.R.,Heller, G.Z.and Offutt, J. Impact of release intervals on empirical research in to software evolution, with, application to the main trainability Of Linux.	IET	2009
S16	Hemmati, H. ,Zhihan Fangand Mantyla, M.V. Prioritizing Manual Test Cases in Traditional and Rapid Release Environments.	ICST	2015
S17	Souza, R., Chavez, C.and Bittencourt, R.A.Rapid Releases and Patch Backouts: A Software Analytics Approach.	IEEE	2015
S18	Md Tajmilur, Rahmanand Peter C.Rigby Release Stabilization on Linux and Chrome.	IEEE	2015
S19	Biffl, S. , Sunindyo,W.D.andMoser,T.Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release, Software Projects.	CISIS	2010
S20	Cohan, S.Success ful Integration of Agile Development Techniques with in DISA.	AGILE	2007
S21	Baysal,O.,Kononenko,O.,Holmes,R.and Godfrey, M.W.,The Secret Life of Patches: A Firefox Case Study.	WCRE	2012

ID	Autor, Título	Local de Publicação	Ano
S22	Mantyla M, Adams B, Khomh F, Engstrom E and Petersen K On rapid releases and software testing: A case study and a semi-systematic, literature review.	ESSE	2015
S23	Souza,R,Chavez Cand Bittencourt R.Patch rejection in Firefox: negative reviews, backouts, and is suere opening.	SBES	2015
S24	Adams B, Khomh F, Dhaliwal Tand ZouYing Understanding the impact of rapid releases on software quality.	ESSE	2015
S25	Xiang Li, Yan Fu Li, Min Xie, Szu Hui Ng. Reliability analysis and optimal version-up dating for opensource software.	GSE	2011
S26	P Rodriguez, e tal. Continuous deployment of software intensive products and services: Asystematic mapping, study.	JSS	2016
S27	M V, Mantyla, K Petersen and, TOA Lehtinen, Time Pressure: A Controlled Experiment of Test Case Development and Requirements, Review.	ICSE	2014
S28	M, Michlmayr, B Fitzgerald and K J Stol, Why and How Should Open Source Projects Adopt Time-Based Releases?	IEEE	2015
S29	B Adams, M Michlmayr Modern Release Management Ina Nutshell: Why Researchers should Care.	SANER	2016
S30	B Fitzgerald, S McIntosh Time- Based Release Managementin Free/Open Source (FOSS) Projects.	LERO	2011