UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE EMERSON DE OLIVEIRA CALIXTO

INVESTIGANDO MUDANÇAS EM RELATÓRIOS DE BUGS DO BUGZILLA

CAMPINA GRANDE - PB

2022

FELIPE EMERSON DE OLIVEIRA CALIXTO

INVESTIGANDO MUDANÇAS EM RELATÓRIOS DE BUGS DO BUGZILLA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Professor Dr. Franklin de Souza Ramalho

CAMPINA GRANDE - PB

2022

FELIPE EMERSON DE OLIVEIRA CALIXTO

INVESTIGANDO MUDANÇAS EM RELATÓRIOS DE BUGS DO BUGZILLA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Franklin de Souza Ramalho

Orientador – UASC/CEEI/UFCG

Professor Dr. Everton Leandro Galdino Alves

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de Setembro de 2022.

CAMPINA GRANDE - PB

RESUMO

O comportamento da ocorrência de mudanças, em sistemas de rastreamento de bugs, pode identificar problemas relacionados à dificuldade do autor de um relatório de bug em preencher informações corretamente e/ou a falta delas. Além disso, estudos com relatórios de bugs pouco exploram a utilização de features de mudanças na construção de modelos/ferramentas. Este estudo investiga as mudanças para identificar quais campos de um relatório mais se alteram, qual o perfil dos bugs que mais são alterados e a existência de relações entre mudanças. Descobrimos que os campos mais alterados são *flagtypes.name* e *cc*. Relatórios tendem a terem mais modificações quando são relatórios de bugs válidos, com prioridade média/alta e severidade média/alta. E, existem correlações de grau moderado a alto, entre as mudanças dos pares de campos: *product*/*component*, *priority*/*severity* e *platform*/*op_sys*.

Investigando Mudanças em Relatórios de Bugs no Bugzilla

Felipe Emerson de Oliveira Calixto
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
felipe.calixto@ccc.ufcg.edu.br

Orientador: Franklin de Souza Ramalho
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
franklin@computacao.ufcg.edu.br

Coorientador: Tiago Massoni
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
massoni@computacao.ufcg.edu.br

RESUMO

O comportamento da ocorrência de mudanças, em sistemas de rastreamento de bugs, pode identificar problemas relacionados à dificuldade do autor de um relatório de bug em preencher informações corretamente e/ou a falta delas. Além disso, estudos com relatórios de bugs pouco exploram a utilização de *features* de mudanças na construção de modelos/ferramentas. Este estudo investiga as mudanças para identificar quais campos de um relatório mais se alteram, qual o perfil dos bugs que mais são alterados e a existência de relações entre mudanças. Descobrimos que os campos mais alterados são *flagtypes.name* e *cc*. Relatórios tendem a terem mais modificações quando são relatórios de bugs válidos, com prioridade média/alta e severidade média/alta. E, existem correlações de grau moderado a alto, entre as mudanças dos pares de campos: *product*/*component*, *priority*/*severity* e *platform*/*op_sys*.[1]

Palavras-chave

Relatório de bug, Sistemas de rastreamento de bugs, Bugzilla

1. INTRODUÇÃO

No desenvolvimento de software, é essencial contar com a ajuda dos usuários do sistema para identificar bugs, pois dificilmente um sistema está livre de falhas. Além disso, sistemas com uma base de usuários volumosa recebem diariamente um grande número de relatórios de bugs. Por exemplo, a Mozilla recebe diariamente cerca de 307 novos relatórios [1]. Assim, são utilizados sistemas para identificação e monitoramento de bugs, como Bugzilla[2], JIRA[3], entre outros. Tais sistemas permitem ao usuário, seja este usuário final ou desenvolvedor, criar e acompanhar relatórios. Estes relatórios em sua maioria são de bugs, mas também podem descrever solicitações de novos recursos/melhorias.

Um relatório de bug pode conter diversas informações importantes como: produto afetado, componente afetado, prioridade, severidade, classificação do bug, tipo do bug, entre outros. Desses campos, vários podem não ser reportados pelo usuário, seja por falta de conhecimento ou atenção, ou então serem informados de maneira errada ou imprecisa. Zimmerman et al. [2] identificaram indícios de uma incompatibilidade entre o que os desenvolvedores consideram mais útil e o que os relatores fornecem. Para o desenvolvedor, relatórios com tais problemas podem tomar mais tempo para serem entendidos e resolvidos.

Campos de um relatório de bug podem não ser preenchidos ou informados corretamente, devido à quem preenche um relatório muitas vezes ser um usuário final do sistema, que pode não ter conhecimento técnico a respeito das várias características de um bug. Como consequência, a compreensão do relatório por parte dos desenvolvedores e o tempo até o bug ser resolvido podem ser afetados.

Ao longo do ciclo de vida de um relatório de bug, este passa por inúmeras atualizações, desde mudanças de status (quando o bug é confirmado ou corrigido, por exemplo) até mudanças para correção de informações, como por exemplo, o produto afetado ser outro. Outras mudanças comuns acontecem para identificar certos aspectos de um bug: quando um bug está bloqueando (bug bloqueante) a correção de outros (bug bloqueado), quando um bug é identificado e validado (bug válido) ou o bug descrito no relatório é inválido (bug inválido) e também quando um relatório de bug aborda sobre um bug já relatado anteriormente (bug duplicado).

Para entender o comportamento das mudanças nos relatórios, levantamos as seguintes questões de pesquisa: (QP1) Quais são os campos que mais se alteram? (QP2) Qual o perfil dos bugs que mais se alteram? (QP3) Existem relações entre as mudanças de campos em relatórios de bugs?

A investigação revelou que as mudanças ocorrem em dois tipos de campos: *custom*[4] (campos cujo nome começa por "cf_" e são criados pelos administradores do Bugzilla) ou não *custom* (campos que existem por padrão), sendo este último tipo o que mais passa por alterações, com média de 12,24 (vs. 2,04) de modificações durante o ciclo de vida. Considerando a média, os campos que mais se alteram são *cc* (usuários inscritos para acompanhar um dado relatório) e *flagtypes.name*[5] (usado para requisitar informações a um usuário). Os relatórios de bugs com mais mudanças são relacionados a bugs válidos (com resolução *FIXED*), com prioridade média ou alta (P1, P2 ou P3) e severidade média ou alta (S1, S2, S3, *blocker*, *critical* ou *major*). Há produtos, como "Infrastructure & Operations", que possuem uma quantidade média de mudanças abaixo em relação aos demais. E, quanto às relações entre mudanças, foram encontradas correlações significativas entre a ocorrência simultânea de alguns pares de campos e suas quantidades de modificações individuais: entre *platform* (0,94) e *op_sys* (0,86), entre *product* (0,64) e *component* (0,69), e entre *priority* (0,28) e *severity* (0,61);

[2] https://bugzilla.mozilla.org/

[3] https://www.atlassian.com/software/jira

[4] https://www.bugzilla.org/docs/3.0/html/custom-fields.html

[5] https://wiki.mozilla.org/BMO/UserGuide#Needinfo_Flag

Este documento está organizado como segue. A seção 2 discorre sobre alguns conceitos relacionados ao Bugzilla. A seção 3 trata sobre outros trabalhos envolvendo relatórios de bugs. A seção 4 apresenta a metodologia deste estudo. A seção 5 expõe os resultados alcançados. A seção 6 expõe possíveis ameaças à validade da pesquisa. E, por fim, a seção 7 apresenta as conclusões e possíveis linhas de pesquisa para futuros trabalhos.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção, entraremos em mais detalhes sobre a definição dos campos que foram abordados neste trabalho e também sobre como o Bugzilla estrutura o histórico de mudanças de um relatório de bug.

2.1 Definição dos campos

A seguir, temos o nome, a descrição e valores possíveis dos principais campos abordados neste estudo:

- *id*. É um campo numérico usado para identificar unicamente um relatório de bug;
- *history*. Contém todos os registros de modificações. É apresentado em mais detalhes na subseção 2.2;
- *resolution*. Indica a resolução atual de um relatório. A resolução pode ser uma das seguintes: *FIXED*, *INVALID*, *INCOMPLETE*, *DUPLICATE*, *WORKSFORME*, *WONTFIX*, *INACTIVE* ou *MOVED*. Quando um relatório foi identificado como, de fato, sendo um bug e foi corrigido, sua resolução será *FIXED*. Quando um relatório descreve um bug que já foi reportado anteriormente, o relatório é resolvido como *DUPLICATE*. Enquanto que os restantes dos valores se referem a bugs que não são reproduzíveis. Mais detalhes sobre cada valor podem ser obtidos através da documentação[6] do Bugzilla;
- *product.* O produto afetado pelo bug. Na Mozilla tem, por exemplo, o Firefox e o Thunderbird;
- *component*. O componente afetado pelo bug. Cada componente pertence a um produto específico e um produto pode ter vários componentes;
- *priority*. Define a importância que um bug tem para ser corrigido comparado a outros. A prioridade possui valores num intervalo que vai de P1 a P5, onde P1 diz respeito a um bug com prioridade máxima e P5 a um bug com prioridade muito baixa.
- *severity*. Descreve a severidade de um bug. Atualmente, o Bugzilla da Mozilla possui duas escalas de severidade. A primeira escala[7] possui valores de S1 a S4, que vai de uma severidade catastrófica para uma severidade trivial, respectivamente. Além disso, a severidade pode possuir o valor N/A para relatórios em que não se aplica uma classificação de severidade, por exemplo, quando um relatório é de sugestão de melhoria. A segunda escala[8] é mais auto-explicativa, pois cada valor já indica semanticamente o grau de severidade (*blocker*, *critical*, *major*, *normal*, *minor*, *trivial*, *enhancement*).

2.2 Estrutura do histórico de um relatório de bug

A Figura 1 mostra a estrutura do histórico utilizada pelo Bugzilla para registrar as modificações de um relatório de bug. Em suma, cada relatório de bug possui um atributo *history* que é um *array*, este pode ser vazio (quando nenhuma mudança foi registrada ainda) ou conter vários objetos do tipo *ChangeSet*. *ChangeSet* é composto por três atributos: *who* contém o email do usuário que fez a mudança, *when* é a data na qual a mudança foi feita e *changes*, um *array* de objetos do tipo *Change*. um *ChangeSet* obrigatoriamente inclui ao menos uma mudança (*Change*). Por último, um objeto *Change* é constituído pelos atributos: *field_name* (nome do campo que foi alterado), *added* (o valor que foi adicionado) e *removed* (o valor que foi removido).

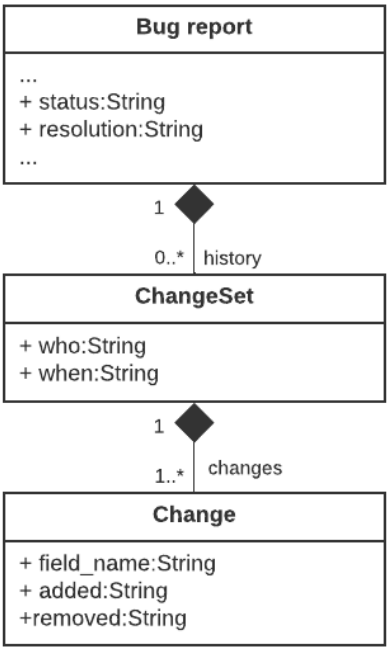

Figura 1: Estrutura[9] do histórico.

3. TRABALHOS RELACIONADOS

Zimmerman et al. [2] desenvolveram uma ferramenta para medir a qualidade de relatórios de bugs e sugerir melhorias ao autor do relatório. Já David Lo et al. [1], desenvolveram um modelo para predizer se um relatório de bug é válido ou não.

Hooimeijer e Westley [4] construíram um modelo para predizer quando um bug será triado dado uma determinada quantidade de tempo, para isso utilizaram três *features* relacionadas a mudanças: quantidade de mudanças de *severity*, quantidade de comentários e quantidade de anexos. Shihab e Garcia [3] também trabalharam com modelos de predição, mas para predizer se um bug é bloqueante ou não e usaram a *feature priority has increased*, isto é, se a prioridade subiu após o relatório inicial.

Joorabchi et al. [5] fizeram uma caracterização de relatórios de bugs não reproduzíveis e utilizaram o histórico de mudanças de *status* e *resolution*. Enquanto que Rocha et al. [6] realizaram uma caracterização, mas focada em fluxos de trabalho no Mozilla Firefox. Por fim, Gupta [7] também explorou fluxos de mudanças de status.

4. METODOLOGIA

Este trabalho tem como objetivo analisar o histórico de mudanças de relatórios de bugs reportados na base de dados do Bugzilla. Por meio de uma análise exploratória sobre esta base de dados, tem-se como objetivos: (i) identificar quais campos são mais alterados; (ii) o perfil dos relatórios que mais apresentam mudanças e (iii) se há relações entre mudanças.

O estudo foi realizado tendo em vista responder às seguintes questões de pesquisa:

[6]https://wiki.documentfoundation.org/QA/Bugzilla/Fields/Status/RESOLVED

[7] https://wiki.mozilla.org/BMO/UserGuide/BugFields

[8]https://wiki.documentfoundation.org/QA/Bugzilla/Fields/Severity

[9] https://wiki.mozilla.org/Bugzilla:BzAPI:Objects

QP1. Quais são os campos que mais se alteram?

QP2. Qual o perfil dos bugs que mais se alteram?

QP3. Existem relações entre as mudanças de campos em relatórios de bugs?

A subseção 4.1 descreve a base de dados utilizada, enquanto a subseção 4.2 indica as métricas usadas no trabalho e a subseção 4.3 detalha o procedimento utilizado. A subseção 4.4 apresenta os campos utilizados. Todo o material do estudo está disponível em: https://github.com/felipeemerson/Bugzilla-mozilla-investigation.

4.1 Base de Dados

A base de dados escolhida foi a do Bugzilla da Mozilla, devido à sua popularidade, disponibilidade e por vários estudos prévios trabalharem com ela [1], [2], [3], [4], [5]. Foram definidos os seguintes filtros para a sua coleta:

- Intervalo da data de criação do relatório: entre 01/01/2013 e 01/01/2022. Abrange um período de 9 anos garantindo um bom volume de relatórios de bugs;
- Status: *RESOLVED*. Limita a base a apenas relatórios de bugs que encontram-se resolvidos. Desse modo, diminuindo as chances dos relatórios apresentarem novas mudanças.

No total, a base de dados utilizada possui 690.817 relatórios de bugs.

4.2 Métricas

As métricas utilizadas no trabalho foram:

- Total de modificações por relatório de bug. Permite verificar quais relatórios de bugs mais se alteram.
- Total de modificações em um campo por relatório de bug. Dado um campo específico, verificamos quantas vezes ocorreram mudanças nele.
- Percentual de presença de modificações por campo. É o percentual dos relatórios de bugs que registraram ao menos uma modificação num dado campo. Fornece indícios sobre quais modificações são mais frequentes de acontecer.

A métrica total de modificações em um campo foi utilizada, no campo *severity*, como feature no trabalho de Hooimeijer e Westley [4]. O restante das métricas foram propostas neste estudo.

4.3 Procedimento

Os dados utilizados neste trabalho foram obtidos em duas etapas: 1) coleta dos dados e 2) preparação dos dados.

4.3.1 Coleta de dados

A coleta dos dados foi feita através da API REST[10] do Bugzilla, utilizando scripts em Python. Os dados são retornados em formato JSON. Devido à quantidade massiva de dados, estes foram armazenados no banco de dados MongoDB[11] e, foram acessados utilizando a biblioteca mongoengine[12].

4.3.2 Preparação dos dados

Com o propósito de obter os valores para as métricas descritas na seção 2.2, os dados foram processados por meio de scripts Python e os resultados foram gravados em arquivos JSON. Os arquivos são utilizados em documentos do tipo notebook para produzir os gráficos e estatísticas a serem analisadas. Para tanto, foram utilizadas as bibliotecas pandas[13], matplotlib[14] e seaborn[15].

4.4 Campos

Os campos explorados neste estudo são:

1. ***id***: identificador único do relatório de bug;
2. ***history***: o histórico de modificações;
3. ***resolution***: indica como o relatório de bug foi resolvido;
4. ***product***: produto afetado;
5. ***severity***: nível de severidade do bug;
6. ***priority***: nível de prioridade do bug.

Desses, *id*, *history* e *resolution* foram baixados entre 03/06/2022 e 04/06/2022; enquanto que os campos *product*, *severity* e *priority* foram acrescentados entre 19/07/2022 e 31/07/2022. A segunda parte de downloads foi necessária devido a identificação posterior da exigência dos mesmos com seus valores finais para responder a QP2. Ademais, foi detectado que um relatório de bug, de *id* 1604167, passou a exigir autorização de acesso, o que levou o mesmo a ficar de fora das análises específicas envolvendo os campos adicionados posteriormente.

5. RESULTADOS

Os resultados obtidos para as questões de pesquisa propostas na seção 4 são descritos e discutidos nesta seção.

5.1 QP1. Quais são os campos que mais se alteram?

Para responder a essa questão, foi necessário identificar quais campos registraram ao menos uma modificação em toda a base de dados (subseção 5.1.1). Após verificarmos que boa parte dos campos são *custom*, investigamos o percentual de presença destes (subseção 5.1.2) para entender se são muito ou pouco utilizados. Na subseção 5.1.3, apresentamos as estatísticas gerais de mudanças, que foram coletadas para ter uma melhor compreensão dos resultados obtidos. Na subseção 5.1.4, apresentamos a resposta para a questão de pesquisa QP1.

5.1.1 Total de campos que registraram ao menos uma modificação

Como mostrado na Tabela 1, identificamos um total de 617 campos que apresentaram registro de pelo menos uma mudança na base. Dando uma olhada nos nomes dos campos, percebeu-se que 578 campos começam com o prefixo "cf_", usado para identificar campos *custom*, estes que são criados pelos administradores do Bugzilla para atender alguma demanda que campos já existentes não atendem.

Dentre os campos *custom*, a maioria são campos relacionados a versões de um produto específico, e começam com os prefixos "cf_status" ou "cf_tracking".

No total, existem 39 campos não *custom*, apenas 6% do total.

Tabela 1: Quantidade de campos que registraram mudanças por tipo

Campos		Quantidade	
Campos *custom*	Relacionados à produto	578 (93,68%)	544 (88,17%)
	Não relacionados		34 (5,51%)

[10] https://bmo.readthedocs.io/en/latest/api/index.html
[11] https://www.mongodb.com/
[12] http://mongoengine.org/
[13] https://pandas.pydata.org/
[14] https://matplotlib.org/
[15] https://seaborn.pydata.org/

à produto	
Campos não *custom*	39 (6,32%)
Total	617 (100%)

5.1.2 Quais campos custom tem um percentual de presença maior?

Considerando apenas os campos *custom* não relacionados à versões de produtos, temos como destacado na Tabela 2 que, o campo *cf_last_resolved* (última data em que o relatório foi considerado como resolvido) tem 100% de presença, ou seja, é atualizado ao menos uma vez em todos os relatórios de bugs da base. Campos como *cf_has_regression_range* (diz se um relatório possui intervalo de regressão), *cf_crash_signature* (guarda a assinatura da falha), entre outros, possuem presença abaixo de 2,48%, isto é, modificações envolvendo estes campos ocorrem em pouquíssimos relatórios.

Tabela 2: Os cinco campos *custom* mais presentes

Campo	Percentual de presença
cf_last_resolved	100%
cf_blocking_b2g	2,48%
cf_qa_whiteboard	1,68%
cf_has_regression_range	1,61%
cf_crash_signature	0,96%

5.1.3 Total de modificações de relatórios de bugs

A Tabela 3 apresenta as estatísticas sobre o total de modificações por relatórios de bugs, tanto para campos *custom* como também campos não *custom* e os dois tipos juntos. Em média, um relatório de bug apresenta 14,29 modificações ao longo do seu ciclo de vida, sendo a maioria das modificações em campos não *custom*. Campos não *custom* apresentam um desvio padrão alto em comparação com campos *custom*, o que indica ter uma maior dispersão nos valores. Quanto aos valores máximos registrados, campos não *custom* chegaram a ter até 1.793 modificações em um único relatório de bug, um valor muito fora de curva se comparado com a média e a mediana.

Tabela 3: Estatísticas sobre o total de modificações

	Média	Mediana	DP	Máximo
Campos *custom*	2,04	1	2,42	94
Campos não *custom*	12,24	8	15,20	1.793
Todos os campos	14,29	10	16,25	1.796

5.1.4 Total de modificações por campo não custom

Como visto na subseção anterior, os campos não *custom* representam a maior parte do total de modificações. A Tabela 4 mostra que só 03 (três) campos atingiram mediana diferente de zero: *cc*, *resolution* e *status*; sendo que destes *resolution* e *status* são campos com 100% de presença em toda a base devido à restrição de todos os relatórios de bugs da base estarem com o *status RESOLVED* e, por consequência do *status* mudar para *RESOLVED* é necessário informar o campo *resolution*. Assim, fora *status* e *resolution*, os campos com maiores percentuais são *cc* (usado para usuários se inscreverem e receberem notificações acerca do relatório), *flagtypes.name* (usado para pedir informações à um usuário) e *assigned_to* (é preenchido quando um desenvolvedor é atribuído para consertar um bug).

Quanto aos valores máximos de modificações, dois campos que indicam relações entre bugs aparecem entre os com maiores valores máximos: *depends_on* (lista dos bugs que bloqueiam o atual) com 1.770 e *blocks* (lista de bugs que estão bloqueados pelo atual) com 353. Já, entre os campos com menores valores máximos, podemos destacar *regressed_by* (lista de bugs que introduziram o atual) com apenas 5 que é outro campo que indica relações entre bugs.

Em resumo, os campos *cc* e *flagtypes.name* se destacaram nos vários cenários, mostrando que estão entre os mais modificados e mais presentes. Para *cc*, a razão provável dos resultados é que a inscrição de usuários para acompanhar relatórios de bug seja muito comum. Já *flagtype.name*, a solicitação de informações de um usuário a outro é comum por várias razões: um desenvolvedor pedindo novas informações sobre o bug descrito, um usuário pedindo a análise de um desenvolvedor, entre outras situações. E provavelmente, em ambos os casos, quanto mais complexo e/ou urgente for o bug, mais usuários tendem a participar do bug, por consequência, ocorrendo mais modificações nos ditos campos.

Tabela 4: Resumo das estatísticas sobre o total de modificações em campos não *custom*

5 campos com maiores médias	5 campos com maiores medianas	5 campos com maiores valores máximos	5 campos com menores valores máximos	5 campos mais presentes
flagtypes.name (2,70)	*cc* (1)	*depends_on* (1.770)	*attachments.isprivate* (4)	*resolution* (100%)
cc (2,34)	*resolution* (1)	*cc* (773)	*restrict_comments* (4)	*status* (100%)
status (1,34)	*status* (1)	*flagtypes.name* (547)	*regressed_by* (5)	*cc* (77,43%)
resolution (1,16)	*product* (0)	*comment_tag* (519)	*bug_mentor* (5)	*flagtypes.name* (48,30%)
comment_tag (0,69)	*component* (0)	*blocks* (353)	*op_sys* (5)	*assigned_to* (34,90%)

5.2 QP2. Qual o perfil dos bugs que mais se alteram?

Agrupar as modificações por valor de um campo pode dar indícios de quais valores tendem a estar relacionados com mais modificações. Esta subseção considera o total de modificações

dos campos não *custom*. Usamos os campos *resolution*, *priority* e *severity* para agrupar os dados. Não utilizamos *status* devido à restrição da base, onde todos os bugs tem status final *RESOLVED*. Além disso, realizamos agrupamento pelo campo *product*, considerando os produtos mais populares dentro da base.

5.2.1 Agrupamento por resolução final

Devido ao intervalo de tempo de download descrito na seção 4.4, 06 (seis) relatórios de bugs ficaram de fora da análise pois foram reabertos dentro do intervalo, e tiveram suas resoluções finais removidas.

A Tabela 5 mostra que, a resolução com mais modificações é *FIXED* com 15,13 modificações em média, enquanto *INACTIVE* e *WONTFIX* possuem em torno de 10 modificações em média, já *DUPLICATE*, *INCOMPLETE* e *INVALID* possuem em torno de 8. Há uma diferença de até cerca de 50% entre a média de modificações de *FIXED* para as outras. Este resultado pode indicar que, relatórios inválidos tendem a serem identificados como tal com um menor grau de mudanças ou que relatórios de bugs válidos possui um ciclo de vida maior, dado que após um bug ser identificado, o seu relatório ainda pode apresentar mudanças que ajudem na correção do mesmo.

Já os maiores valores de modificações foram registrados em *WONTFIX* com 1.793, *INVALID* com 1.692 e *FIXED* com 1.369, estes valores foram registrados nos relatórios de bugs de *id* 838081[16], 950073[17] e 1243581[18], respectivamente. Examinando-os, descobrimos que os mesmos foram usados como um centralizador de relatórios de bugs relacionados a algum projeto em específico. O relatório de *id* 838081 centralizava um *Product Backlog* (lista de requisitos) da interface Metro do Firefox, que nunca chegou a ser lançada. O relatório de *id* 950073 centralizava relatórios do Firefox Desktop. E, o relatório de *id* 1243581 estava relacionado ao projeto Stylo. Dessa maneira, concentravam os relatórios relacionados através do uso do campo *depends_on*, novos relatórios de bugs eram adicionados, e removidos caso fossem resolvidos. Em resumo, nenhum dos três relatórios de bugs estavam ligados a algum bug específico, e possivelmente existam mais relatórios de bugs na mesma situação que estes.

Tabela 5: Qnt. de modificações agrupado por resolução final

Valor	Média	Mediana	DP	Máximo
FIXED	15,13	10	17,88	1.369
INVALID	7,86	6	10,38	1.692
INCOMPLETE	8,24	6	8,74	860
DUPLICATE	8,18	6	7,32	362
WORKSFORME	9,97	8	9,87	580
WONTFIX	10,39	7	14,86	1.793
INACTIVE	10,42	8	12,62	247
MOVED	8,74	6	8,50	99

[16] https://bugzilla.mozilla.org/show_bug.cgi?id=838081
[17] https://bugzilla.mozilla.org/show_bug.cgi?id=950073
[18] https://bugzilla.mozilla.org/show_bug.cgi?id=1243581

5.2.2 Agrupamento por prioridade final

Na Tabela 6 podemos observar que, as maiores médias de modificações por prioridade são de P1 com 19,11, P2 com 17,69 e P3 com 15,40, mas os 3 valores possuem um desvio padrão de até 20,71 modificações. Assim, a mediana pode ser uma estatística comparativa melhor, onde as prioridades P1, P2 e P3 possuem mediana entre 11 e 13 modificações, enquanto que P4 e P5 possuem 7 e 6, respectivamente. Há uma diferença de até cerca do dobro de modificações entre relatórios de bugs com prioridade baixa e bugs com prioridade alta. Além disso, como o campo prioridade não é obrigatório, há uma quantidade de relatórios de bugs alta (478.542) que não possuem prioridade definida (representada pelo valor "--"), estes possuem mediana de 8 modificações.

Os resultados sugerem que relatórios com prioridade média ou alta tendem a apresentar mais modificações que relatórios com baixa prioridade.

Tabela 6: Qnt. de modificações agrupado por prioridade final

Valor	Média	Mediana	DP	Máximo
–	11,41	8	14,28	1.793
P1	19,11	13	20,71	544
P2	17,69	13	19,61	595
P3	15,40	11	17,21	1.003
P4	8,44	7	9,06	860
P5	8,73	6	10,50	797

5.2.3 Agrupamento por severidade final

Diferentemente de *priority*, cujo os valores são numa escala de P1 a P5, *severity* possui duas escalas diferentes, uma que vai de S1 a S4 e outra cuja classificação é feita por palavras (*critical*, *normal*, *enhancement*, etc). Aparentemente, a escala S1 a S4 foi adicionada depois, e ambas coexistem, ficando a critério do usuário qual escolher.

Analisando as estatísticas da Tabela 7, temos que para a primeira escala, a severidade S4 possui a menor média de modificações com 8,39, enquanto que S1, S2 e S3 possuem respectivamente 16,68, 19,09 e 15,32 modificações. Já considerando a outra escala, *blocker*, *critical* e *major* que são os maiores níveis de severidade também possuem as maiores médias de modificações com 17,19, 15,38 e 15,87, respectivamente.

De modo parecido com o identificado em prioridade, relatórios com severidade de média a alta possuem uma tendência a terem mais modificações que relatórios com baixa severidade.

Tabela 7: Qnt. de modificações agrupado por severidade final

Valor	Média	Mediana	DP	Máximo
–	8,81	7	6,98	190
N/A	12,12	9	11,48	300
S1	16,68	11	14,05	83
S2	19,09	15	14,67	129

S3	15,32	13	11,32	348
S4	8,39	6	7,64	157
blocker	17,19	11	20,03	530
critical	15,38	11	13,97	254
enhancement	11,53	11	3,32	23
major	15,87	11	16,64	244
minor	11,72	9	9,31	137
normal	12,30	8	15,81	1.793
trivial	12,05	9	21,50	842

5.2.4 Agrupamento por produto final

Na base de dados utilizada existem 163 produtos distintos, assim optamos por agrupar os 10 (dez) mais populares, isto é, os produtos com maiores quantidades de relatórios identificados na base de dados. Observando a Tabela 8, temos que o produto "Infrastructure & Operations" é o produto com a menor média entre todos, sua média chega a ser cerca de 2 (duas) vezes menor em comparação com os outros produtos. Logo, pode ser possível que dependendo do contexto de um produto, relatórios de bugs que o afetam apresentem menor número de mudanças do que relatórios de outros produtos.

Tabela 8: Qnt. de modificações agrupado por produto final (os 10 mais populares)

Valor	Média	Mediana	DP	Máximo
Core	14,53	10	18,01	1.369
Firefox	11,17	8	13,55	767
Firefox OS Graveyard	13,90	9	16,40	576
Testing	10,80	7	13,21	326
DevTools	14,18	10	14,94	423
Infrastructure & Operations	6,15	4	6,27	335
Toolkit	14,34	10	16,43	530
Firefox for Android Graveyard	13,95	10	14,82	433
Thunderbird	11,53	9	11,39	280
Firefox Build System	14,08	10	15,80	545

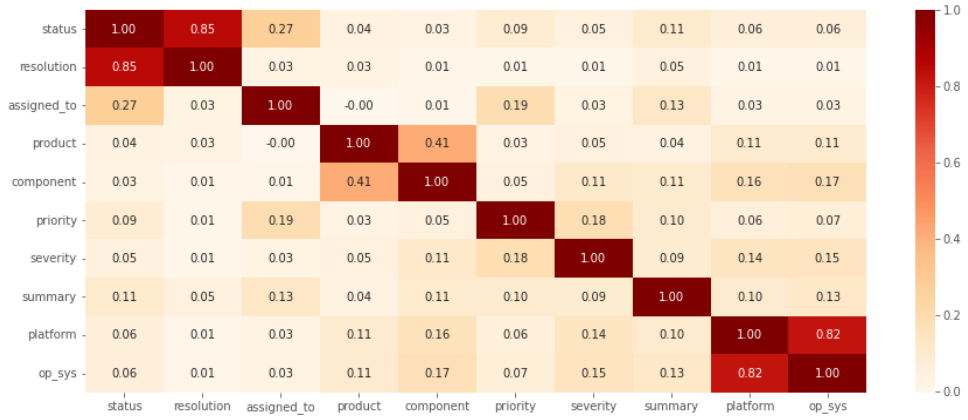

Figura 2: Correlações entre as quantidades de modificações de campos.

5.3 QP3. Existem relações entre mudanças de campos em relatórios de bugs?

Devido a ser possível que várias mudanças sejam registradas de uma só vez, é factível que a presença de uma modificação tenha a ver com a presença de outra modificação, ou que a ocorrência de uma modificação tenha relação com outra modificação que possa vir a acontecer. Assim, é possível que haja correlações entre seus totais de modificações e a ocorrência simultânea entre pares de campos. Para esta análise, foram considerados os campos: *status*, *resolution*, *assigned_to*, *product*, *component*, *priority*, *severity*, *summary*, *platform* e *op_sys*. A escolha se deu por serem campos explorados em outros estudos [1], [2], [3], [4], [5], [6], [7].

Correlação entre *status* e *resolution*. A Figura 2 mostra que existe uma correlação muito alta de 0.85 entre a quantidade de modificações de *resolution* e *status*, o que seria esperado, dado que sempre que o campo resolution muda, o campo *status* também mudará junto, porém nem sempre que o status muda *resolution* mudará junto.

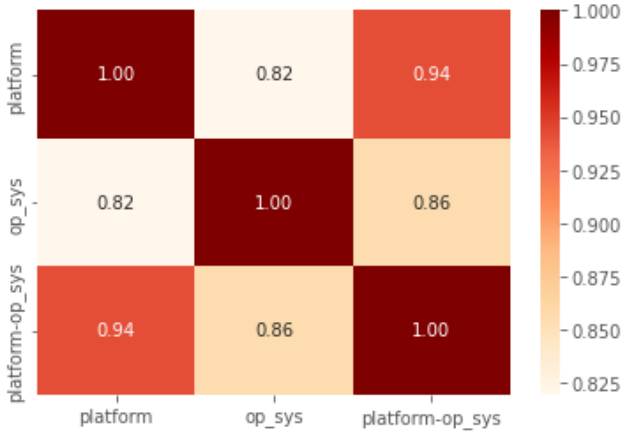

Figura 3: Correlação entre *platform*, *op_sys* e suas ocorrências simultâneas.

Correlação entre *platform* e *op_sys*. Há uma correlação muito alta de 0,82 entre *platform* e *op_sys*, onde *platform* se refere a arquitetura do dispositivo (x86, ARM, etc) enquanto que *op_sys* ao sistema operacional. Ainda é possível explorar a correlação de ocorrências simultâneas desses dois campos com seus totais de modificações. A Figura 3 nos mostra que a correlação entre o total de modificações de *platform* e a ocorrência simultânea entre os dois é de 0,94, ainda mais alto que a correlação anterior. A correlação entre *op_sys* e a ocorrência simultânea é de 0,86 ainda alta, porém um pouco menor. Assim, modificações em *platform* dependem um pouco mais de *op_sys* do que o contrário.

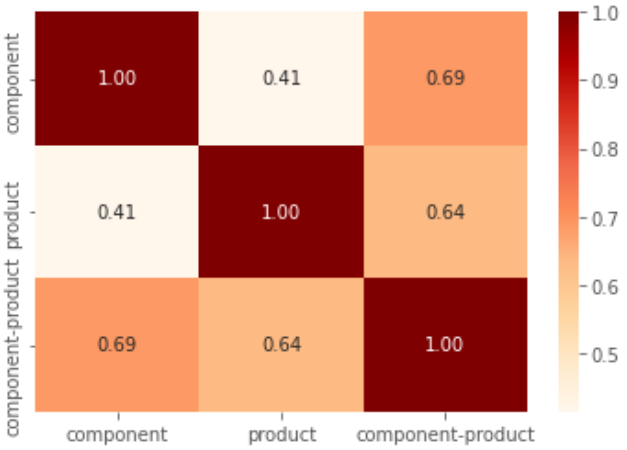

Figura 4: Correlação entre *product*, *component* e suas ocorrências simultâneas.

Correlação entre *product* e *component*. A correlação entre os totais de modificações de *product* e *component* é de 0,41, que é considerado uma correlação moderada. Porém, a Figura 4 nos apresenta que, a correlação entre o número de modificações em *component* e a ocorrência simultânea dos dois é de 0,69, e de *product* com a ocorrência simultânea de 0,64. Isto é, há uma correlação alta de uma modificação no campo *component* ocorrer simultaneamente com *product* e vice-versa.

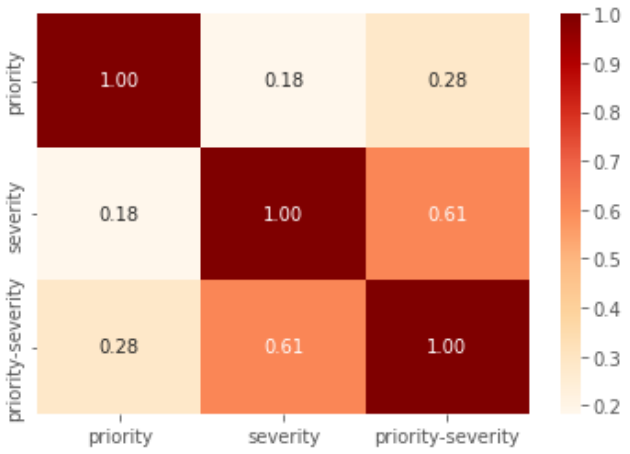

Figura 5: Correlação entre *priority*, *severity* e suas ocorrências simultâneas.

Correlação entre *priority* e *severity*. A correlação entre a quantidade de modificações de *priority* e *severity* individualmente é de 0,18, um valor muito baixo. Mas, como visto na Figura 5, quando se leva em consideração a ocorrência simultânea, há uma correlação alta de 0,61 entre *severity* e suas ocorrências simultâneas e uma correlação baixa entre priority e suas ocorrências simultâneas. Ou seja, *severity* ocorre com mais frequência acompanhado de *priority* do que o inverso.

6. AMEAÇAS À VALIDADE

Ameaças internas à validade. Os procedimentos realizados na fase de coleta e/ou processamento podem ser possíveis fontes de erros. Por isso, verificamos cada procedimento mais de uma vez. Outro fator é que os dados estão em constante atualização, existindo a possibilidade de relatórios de bugs serem reabertos e, consequentemente, terem novas atualizações. Levando isso em conta, fizemos o download dos históricos de mudanças dos relatórios em um período curto (3 dias). Embora, posteriormente foi necessário o download de mais campos, o que levou a 6 bugs serem reabertos entre o intervalo.

Ameaças externas à validade. Este estudo focou apenas na base de dados do Bugzilla da Mozilla, portanto os resultados podem não ser válidos para outras bases de dados, sejam elas abertas ou fechadas.

7. CONCLUSÕES

O presente estudo identificou que cerca de 85% das modificações ocorrem em campos não *custom* (campos *custom* são aqueles iniciados com prefixo "cf_"). Com exceção de *cf_last_resolved*, campos *custom* têm menos de 3% de presença de modificações em relatórios de bugs. Portanto, podem não ser promissores como *features*.

Os campos *cc* e *flagtypes.name* são os mais modificados. Destes, apenas *cc* foi utilizado nos trabalhos [3] e [5]. Futuros trabalhos podem avaliar o uso do campo *flagtypes.name* como feature nos seus modelos ou ferramentas.

Descobrimos que relatórios de bugs no Bugzilla da Mozilla tendem a apresentar uma média maior de modificações quando são bugs válidos, com prioridade média-alta e/ou severidade média-alta. Essas informações podem ajudar na estimativa de esforço necessário no rastreamento e correção de bugs. Outros estudos poderiam investigar quais outros campos, e seus respectivos valores, afetam a média de modificações em relatórios de bugs.

Quanto às relações entre mudanças, identificamos que a correlação entre modificações de pares de campo pode ser promissora. Por exemplo, os campos *platform* e *op_sys* apresentam uma correlação muito forte (0,94 e 0,86, respectivamente) entre a ocorrência simultânea de mudanças dos mesmos. Ou seja, a maioria das mudanças em *platform* ocorreram em conjunto com mudanças de *op_sys* e vice-versa. Já com os campos *product* e *component* há uma correlação moderada nos dois casos. Os próximos trabalhos podem verificar mais campos para encontrar novos pares promissores.

Um estudo comparativo envolvendo várias bases de dados é interessante para generalizar mais os resultados obtidos.

8. REFERÊNCIAS

[1] Fan, Y., Xia, X., Lo, D., & Hassan, A. E. Chaff from the wheat: characterizing and determining valid bug reports. IEEE Transactions on Software Engineering, 46(5), 495-525, 2020.

[2] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss. What makes a good bug report? IEEE Transactions on Software Engineering, 36(5), 618–643, 2010.

[3] VALDIVIA GARCIA, Harold; SHIHAB, Emad. Characterizing and predicting blocking bugs in open source projects. In: Proceedings of the 11th Working Conference on Mining Software Repositories, 72-81, 2014.

[4] HOOIMEIJER, Pieter; WEIMER, Westley. Modeling bug report quality. In: Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, 34-43, 2007.

[5] ERFANI JOORABCHI, Mona; MIRZAAGHAEI, Mehdi; MESBAH, Ali. Works for me! characterizing non-reproducible bug reports. In: Proceedings of the 11th Working Conference on Mining Software Repositories, 62-71, 2014.

[6] Henrique Rocha, Guilherme de Oliveira, Marco Tulio Valente, and Humberto Marques-Neto. Characterizing Bug Workflows in Mozilla Firefox. In Proceedings of the 30th Brazilian Symposium on Software Engineering (SBES '16). Association for Computing Machinery, New York, NY, USA, 43–52, 2016.

[7] Monika Gupta and Ashish Sureka. Nirikshan: mining bug report history for discovering process maps, inefficiencies and inconsistencies. In Proceedings of the 7th India Software Engineering Conference (ISEC '14). Association for Computing Machinery, New York, NY, USA, Article 1, 1–10, 2014.