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

THAÍS NICOLY ARAÚJO TOSCANO

ABORDAGENS PARA ESCRITA DE RELATÓRIOS DE BUGS: UM ESTUDO QUALITATIVO

CAMPINA GRANDE - PB

2022

THAÍS NICOLY ARAÚJO TOSCANO

ABORDAGENS PARA ESCRITA DE RELATÓRIOS DE BUGS: UM ESTUDO QUALITATIVO

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 ou Bacharela em Ciência da Computação.

Orientador: Professor Tiago Lima Massoni.

CAMPINA GRANDE - PB

2022

THAÍS NICOLY ARAÚJO TOSCANO

ABORDAGENS PARA ESCRITA DE RELATÓRIOS DE BUGS: UM ESTUDO QUALITATIVO

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 ou Bacharela em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Tiago Lima Massoni

Orientador – UASC/CEEI/UFCG

Professor Dr. Maxwell Guimarães de Oliveira

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 06 de Abril de 2022.

CAMPINA GRANDE - PB

Abordagens para Escrita de Relatórios de Bugs: Um Estudo Qualitativo

Thaís Nicoly Araújo Toscano
thais.toscano@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil

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

RESUMO
O sucesso de um software condiz com a atenção em todo processo de desenvolvimento que envolve uma série de etapas até que o produto esteja em produção. No contexto da qualidade deste software, a utilização de relatórios de bugs providos por usuários finais contribui em maior escala para um melhor entendimento para desenvolvedores, equipes de suporte e sustentação, por fornecerem informações cruciais para a execução de correções ou melhorias no sistema. Entretanto, a falta de informações dificultam a compreensão dos bugs, justamente por existir casos onde o usuário relata falsos bugs, com isso, para reconhecer esses casos e alcançar a proficiência para atender às necessidades , é essencial obter detalhamento, seja ele com título, descrições de cenários, passo a passo para execução, grau de severidade, anexos como planilhas, capturas de tela. Dessa forma, torna-se a leitura mais clara, objetiva e concisa. Dado este cenário, através de uma análise qualitativa, avaliamos a qualidade das informações contidas nesses relatórios, se dentre eles existem ou não padronização, procurar entender também a dificuldade dos usuários ao realizar o repasse dessas informações, e, a partir disso, indicar quais os formatos das abordagens que satisfazem as partes envolvidas para o correto funcionamento do software.

Palavras-Chave
Bug, Relatório de bugs, Comunicação Informal, Relatório inválido.

1. INTRODUÇÃO

O desenvolvimento de um software pode ser considerado difícil, por envolver equipes, processos e consequentemente, manter este software atualizado e em funcionamento sem defeitos, se torna ainda mais difícil. Uma das ferramentas consideradas importantes para a criação e manutenção de um software de qualidade, com o mínimo de defeitos são os relatórios de defeitos, ou *bug reports*. Relatórios de bugs permitem aos usuários informarem aos desenvolvedores sobre os problemas encontrados ao usar um software, eles normalmente contêm uma descrição detalhada de uma falha e ocasionalmente sugerem a localização da falha no código. No entanto, os relatórios variam na qualidade do conteúdo, isso porque muitas vezes trazem informações incompletas [1].

Ao deixar de enviar informações relevantes, fornecendo-as de forma incompleta, faz surgir problemas que categorizam o que seria um relatório ruim. Relatórios ruins não contém as informações necessárias para reproduzir e corrigir os problemas, por não conter informações específicas, se torna um meio de comunicação ineficiente para ambos os envolvidos e por conseguinte acaba não sendo resolvido e resulta no status ‘rejeitado’.

Este trabalho tem como objetivo analisar e discutir as dificuldades que desenvolvedores e usuários possuem na hora do relato de um bug em um software, investigando a qualidade desses relatórios através de uma análise qualitativa, em que o público alvo são desenvolvedores que possuem em seu tempo de carreira este tipo de atividade. Foram selecionadas dez pessoas de diferentes empresas em que seu produto é software a fim de obter resultados em diversas realidades. O estudo foi elaborado a partir de um roteiro com dez perguntas discursivas, com os seguintes temas: *interação entre desenvolvedores e usuários* e *relatórios rejeitados,* na qual a duração da entrevista varia de acordo com a quantidade de informações que cada um se propôs a compartilhar, com no máximo quinze minutos.

A partir disso, dois questionamentos principais foram elaborados : *Como as interações entre desenvolvedores e usuários,*

que utilizam ferramentas de relatórios de bugs, influencia no entendimento e posteriormente na resolução de um bug ? e *Como relatórios de bugs incompletos e/ou mal-escritos podem gerar impactos negativos na correção de uma falha?*, a metodologia aplicada consiste em análises que juntas resultaram em proposições que culminam na resolução dessas perguntas no intuito de apontar os melhores cenários para a criação de bons relatórios de bugs.

Acreditamos que os resultados desta pesquisa podem ajudar a esclarecer dúvidas a respeito de como as empresas que produzem software para o mercado adotam em seu cotidiano a relação desenvolvedor e usuário ao lidarem com relatórios de bugs. Além disso, os motivadores podem servir como base para o planejamento e ações para contribuir aos usuários as melhores maneiras de repassar as informações para os desenvolvedores de software. As principais contribuições deste artigo são:

- Identificação de uma boa relação entre desenvolvedor e usuário, com base nos dados coletados. Nesse caso, todas as empresas vêm adotando formatos de interação em que todos os entrevistados são satisfatórios.
- Uma discussão sobre os campos presentes nos relatórios na qual trabalham atualmente e quais eles acham fundamentais para que se tenha o contato com quem reportou, pois vem muitas vezes em branco ou incompleto.
- O conhecimento dos principais campos e sua importância, que podem servir para pesquisas adicionais onde o experimento pode ser adotar formas de instruir aos usuários como preenchê-los corretamente.

2. FUNDAMENTAÇÃO TEÓRICA

Antes de iniciarmos as discussões a respeito do que seria considerado um bom relatório de bug, faz-se necessário discorrer a respeito de alguns conceitos que estão entrelaçados, que seria o que é um bug, o que é o ciclo de vida de um bug e trazer a diferença entre relatórios bons e ruins através da perspectiva dos desenvolvedores.

2.1 O que é um bug ?

O Macquarie Dictionary[6] define um bug de software como um erro em um programa ou na própria máquina, muitas vezes não detectado pelos testes mais rigorosos, entre outras palavras, um bug de software pode ser conceituado como um erro ou falha em um programa de computador ou sistema que faz com ele produza um resultado incorreto ou inesperado. Não importando quem o escreveu, todo e qualquer bug merece um relatório para que se tenha uma investigação formal sobre e o quanto antes ele for exposto, melhor, pois bugs tendem a ficar cada vez mais caros, conforme o tempo passa. Estima-se que um bug descoberto em produção custa 10 vezes mais o valor do que se ele tivesse sido encontrado durante o desenvolvimento, por exemplo[7].

Um bug pode ser introduzido em qualquer ponto do Ciclo de Vida de Desenvolvimento de Software. Além disso, quanto mais cedo o bug for detectado e removido, menor será o custo total da qualidade. Por fim, o custo da qualidade é minimizado quando o bug é removido na mesma fase em que foi introduzido[2].

2.2 O que é um ciclo de vida de um bug ?

O ciclo de vida do bug é a sequência de fases pelas quais o bug passa desde sua identificação até o fechamento [2]. Este ciclo de vida garante que a correção resolva o bug e também o rastreie até o encerramento, verifica que o sistema fica livre do defeito identificado. Principalmente, esse processo de jornada do bug do início ao fim depende do projeto, mas logo a seguir trazemos uma visão genérica do ciclo de vida de um bug típico.

Um bug passa de um estado para outro em seu tempo de vida de acordo com seu status. Os diferentes estados de um ciclo de vida típico do bug são mostrados na Figura 1. Cada estado pode ser entendido da seguinte forma:

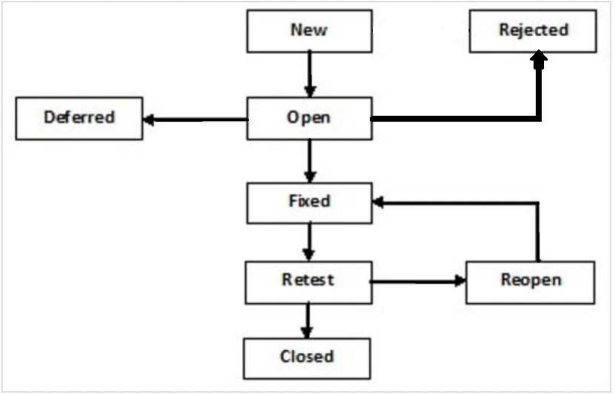

Figura 1: Estados do ciclo de vida de um bug

1) **New**: Este é o primeiro estado de um bug no Ciclo de Vida do Bug. Quando qualquer novo bug é encontrado, ele cai em um estado 'Novo' e as validações e testes são realizados nele nos estados posteriores.
2) **Open**: Aqui, o desenvolvedor inicia o processo de análise do bug e trabalha para corrigi-lo, se necessário. Se o desenvolvedor constatar que o problema que foi passado não tem impacto na funcionalidade do aplicativo, o seu status será alterado para **"Não é um Bug"** ou se achar que o bug não é apropriado, ele pode ser transferido para os estados Rejeitado ou Deferido:
    - **Rejected:** Se o bug não for considerado um defeito genuíno pelo desenvolvedor, ele será marcado como "Rejeitado" pelo desenvolvedor.
    - **Deferred:** Se o desenvolvedor sentir que o bug não possui prioridade muito importante e pode ser corrigido nos próximos lançamentos, pode-se alterar o seu status como 'Diferido'.
3) **Retest**: Neste ponto, o usuário inicia a tarefa de reteste do defeito para verificar se o defeito foi corrigido com precisão pelo desenvolvedor de acordo com os requisitos ou não.
4) **Reopen**: Se algum problema persistir no defeito, ele será atribuído ao usuário novamente para teste e o status do defeito será alterado para 'Reabrir'.
5) **Closed**: Quando o defeito não existe mais, o desenvolvedor altera o status do defeito para “Fechado”.

2.3 O que é um bom relatório de bug ?

Para auxiliar no entendimento desta questão, primeiramente vamos levantar pontos na qual [3] ressalva, após coletar depoimentos onde descrevem a diferença entre um bom e um ruim relatório de bug, seriam eles:

RELATÓRIO DE BUG BOM

- Contém as informações necessárias para reproduzir e corrigir problemas
- É uma forma eficiente de comunicação tanto para o relator de erros quanto para o receptor de erros
- Pode ser e é resolvido o mais rápido possível
- É enviado ao responsável
- É arquivado de forma definida
- Estabelece um terreno comum para a colaboração.

RELATÓRIO DE BUG RUIM

- Não contém as informações necessárias para reproduzir e corrigir problemas
- É uma forma de comunicação demorada e ineficiente para todos os envolvidos
- Nunca se resolve
- Não contém informações específicas
- É arquivado em qualquer meio disponível, mas não da maneira definida
- Não permite a colaboração em equipe ou equipe/cliente.

Em resumo, um bom relatório de bug deve descrever o problema, de forma que o desenvolvedor familiarizado com o projeto possa entender e resolvê-lo, sem falar com a pessoa que o escreveu. Um dos artigos do *BugHerd* [4], define que um relatório de bug é um relatório específico que descreve informações sobre o que está errado e precisa ser corrigido com software, ele pode listar motivos ou erros detectados, para apontar exatamente o que é visto como errado e detalhes de como lidar com cada problema.

Um bom exemplo de um procedimento bem escrito é dado abaixo:

Passos

- Selecione o produto Abc01.
- Clique em Adicionar ao carrinho.
- Clique em Remover para remover o produto do carrinho.

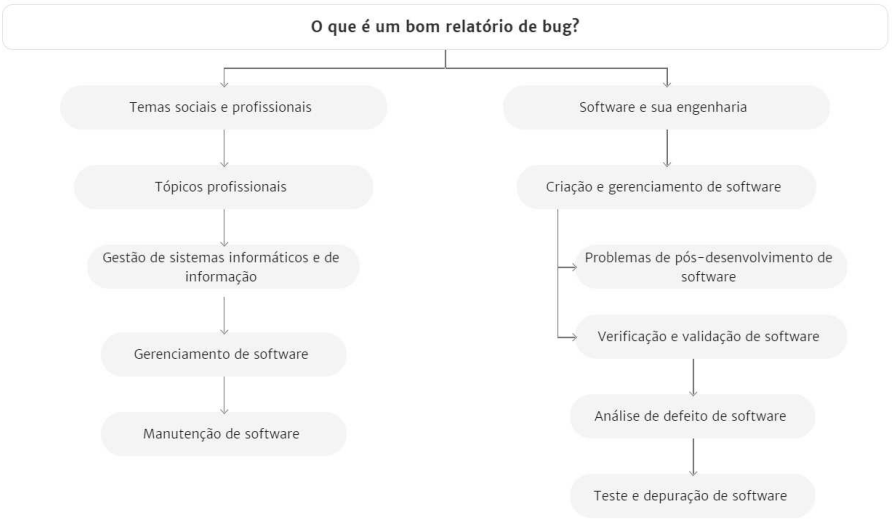

Figura 2: Conceitos relacionados a um bom relatório de bug

INFORMAÇÕES ADICIONAIS SOBRE RELATÓRIOS DE BUG

A Figura 2 mostra a concepção de assuntos envolvidos quando o assunto é um bom relatório de bug, onde deve ser inclusas informações precisas sobre o problema ocorrido. Para que isso aconteça, os seguintes itens precisam estar presentes. O Devrix [5], traz um modelo para relatórios de bugs em que está provado que fornecem informações suficientes sobre os problemas encontrados, são eles:

- **Título**: Deve servir como um breve resumo de qual é o problema. É importante que o título seja específico sobre a natureza do problema. É a primeira coisa que um desenvolvedor vê.
- **Status**: Este é um indicador do status do relatório de bug. Alguns exemplos:
    - **A ser atribuído**: quando ainda não foi determinado quem investigará os problemas do relatório.
    - **Em andamento**: Quando um desenvolvedor está trabalhando para resolver o bug relatado. Este status também é usado quando uma atualização está atualmente em teste.
    - **Completo** - Um sinal de que o bug relatado foi resolvido.
- **Prioridade**: indica a urgência de um bug relatado, em comparação com outras tarefas e problemas. A prioridade de um bug está ligada à severidade de um bug, pois mede o impacto no sistema testado. As prioridades mais comuns são:
    - **Crítico**: Usado para problemas em que todas as outras atividades devem ser interrompidas e todos os esforços são concentrados para resolver a situação. Isso se deve ao fato de que tais problemas têm um grande impacto no projeto para o qual são reportados e podem resultar em perdas para o negócio.
    - **Alto**: Usado para problemas que devem ser resolvidos mais cedo ou mais tarde. Eles não têm um impacto tão grande, mas se não houver problemas críticos, eles devem ser resolvidos o mais rápido possível.
- **Detalhes(Cenário) do bug**: os detalhes contém informações importantes sobre onde o problema relatado foi encontrado. As seguintes informações podem ser incluídas:
    - **O navegador** onde o bug foi encontrado.
    - **O SO**: Pode ser um problema, observado apenas no Windows.
- **Descrição(Passo a passo) do bug**: Deve-se fornecer uma explicação precisa do problema observado, junto com as etapas necessárias para reproduzir o problema. O usuário também poderia descrever qual deveria ser o resultado esperado. Tudo isso é importante para ajudar a equipe de desenvolvimento a entender qual é o problema, quais ações precisam ser

executadas para reproduzi-lo e qual deve ser o comportamento correto.

- **Anexos(Capturas de tela):**Um anexo que muitas vezes vem em forma de captura de tela permite que o desenvolvedor veja qual é e frequentemente onde está o problema.

- **Comentários**: a seção de comentários geralmente é importante para acrescentar informações que o usuário queira compartilhar livremente dos demais campos.

3. METODOLOGIA

Com o objetivo de explorar a comunicação entre desenvolvedores de software e usuários finais, realizamos uma avaliação qualitativa juntamente com o estudo de entrevistas estruturadas[10], com desenvolvedores que em decorrer de sua carreira lidaram com relatos de defeitos para softwares.

3.1 Questões de Pesquisa

- **Objetivo 1:** Como as interações entre desenvolvedores e usuários, que utilizam ferramentas de relatórios de bugs, influencia no entendimento e posteriormente na resolução de um bug ?
- **Objetivo 2:** *Como relatórios de bugs incompletos e/ou mal-escritos podem gerar impactos negativos na correção de uma falha?*

3.2 Seleção dos participantes

A metodologia consiste num roteiro com 10 perguntas, a fim de instigar os entrevistados a compartilhar suas perspectivas em relação a qualidade dos relatórios de bugs que recebem em sua rotina de trabalho.

Os entrevistados são desenvolvedores que atuam em diferentes empresas e em cada relato poderemos discutir a respeito de informações consideradas importantes em bug report e os problemas que enfrentaram com eles, com o objetivo de obter dados para entender como esses profissionais lidam com essa interação com os usuários,eles podem relatar o grau de dificuldade ao deparar com as informações que os usuários fornecem e de suas perspectivas o que consideram mais útil.

Os dados obtidos servirão para entender e mensurar o quão importante é o incentivo às práticas de melhores escritas quando se refere a relatórios de bugs.

ID	GÊNERO	CARGO NA EMPRESA	TEMPO DE CARREIRA
1	Feminino	Analista de Sistemas	4 anos
2	Feminino	Desenvolvedora de Software	2 anos e 9 meses
3	Feminino	Analista Pleno	4 anos
4	Feminino	Desenvolvedora Front-end	3 anos e 6 meses
5	Masculino	Desenvolvedor Autônomo	2 anos
6	Masculino	Analista de Testes	2 anos e 4 meses
7	Masculino	Analista de Testes	4 anos
8	Feminino	Engenheira de Testes Pleno	6 anos
9	Feminino	Desenvolvedora Back-end	2 anos e 5 meses
10	Masculino	Desenvolvedor de Software Pleno	3 anos

Tabela 1: Dados dos entrevistados

3.3 Seleção de estudos primários

Para este estudo, realizamos um roteiro de entrevistas com 10 perguntas. Entrevistas são o método mais utilizado em pesquisas qualitativas. dentro desse cenário, o pesquisador busca respostas ricas e detalhadas, que poderiam ser fornecidas em mais de um e às vezes até em várias ocasiões[9].

Para o primeiro objetivo da pesquisa, foram selecionadas quatro perguntas do roteiro de entrevistas para compor a análise, são elas:

- Como você classificaria a experiência de interagir com usuários que reportam bugs ?Pode exemplificar casos que ilustram sua opinião?
- Quais os relatórios que são mais difíceis de serem resolvidos e por qual motivo? Você acredita que bugs que se enquadram nessa categoria de dificuldade, podem ser aqueles em que o tempo de resolução é maior?
- Quais os principais motivos para entrar em contato com quem reportou o bug?
- Existe alguma tecnologia que você utiliza que te auxilia nessa interação? Quando você precisa entrar em contato com quem reportou, você o faz na mesma ferramenta onde o bug foi reportado?

As demais perguntas do roteiro de entrevistas complementam a investigação, tomando como base o tema relatórios invalidados que seria o segundo objetivo da pesquisa, são elas:

- Quais os principais motivos que já te fizeram invalidar um bug?
- Você entra em contato com quem reporta o bug antes de invalidá-lo?
- Levando em consideração que bugs mais antigos são colocados em prioridade, o indicador de tempo, neste caso, mais te auxilia ou atrapalha na resolução do mesmo ?
- Você considera que relatórios com mais dados ajudam na melhor compreensão? Quais campos estão presentes nos relatórios de bugs que você costuma resolver ?
- Existe algum campo que muitas vezes vem em branco ou incompleto que você considera importante para a resolução do bug?
- A falta do preenchimento de algum campo no relatório, mesmo após interagir com quem reportou, já te fez rejeitar algum bug?

Todas as entrevistas foram gravadas e posteriormente transcritas. No artigo, discutimos as transcrições na qual corrigimos erros gramaticais, bem como intervenções com interjeição entre as perguntas e respostas. O Youtube[8] foi utilizado para tornar a transcrição mais ágil e eficiente.

3.4 Extração de dados

No primeiro ciclo, geramos códigos abertos rotulando as declarações, relacionadas às tarefas de report de bugs feitas pelos participantes. Vários *memos* também foram adicionados para registrar induções potenciais. No método qualitativo, os entrevistados afirmaram que a interação com os usuários não possuem dificuldade em si, mas que se torna constante quando o relatório não descreve bem o passo a passo para se chegar no bug, enxergamos esse padrão ao decorrer das entrevistas e então constatamos que o número total de códigos não mudou, a partir disso concluímos que este ponto havia sido alcançado.

Para a codificação, usamos o MAXQDA Analytics Pro 2022, importando arquivos de texto com transcrições de entrevistas. A ferramenta possibilita a criação, categorização e organização do código. Além disso, os memorandos são anexados aos fragmentos e códigos, contendo insights iniciais sobre a discussão.

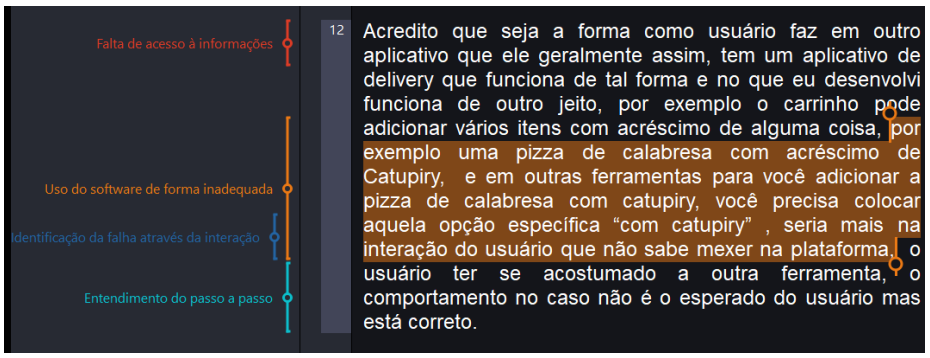

Figura 3: Trecho das anotações utilizando o MaxQDA

4. RESULTADOS

Após a análise, à medida que os códigos foram identificados e abstraídos em conceitos, é possível observar algumas relações entre esses conceitos.

A Figura 4 ilustra algumas associações encontradas neste estudo. Estavam presentes **Importância da comunicação informal**; **Entendimento do passo a passo** e **Identificação da falha através da interação** como conceitos centrais do estudo qualitativo.

Figura 4: Mapa de códigos importado do MaxQDA

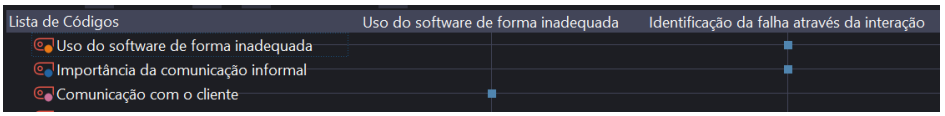

Figura 5: Conexão entre os códigos na ferramenta MaxQDA

Como apresentado na Figura 5, esses foram os códigos principais utilizados no tratamento qualitativo, merecendo destaque para os códigos **Uso de software de forma inadequada**, **Identificação da falha através da interação** e **Importância da comunicação informal**, que implica na conexão direta com o código **Comunicação com o cliente** e isso significa que a maioria dos usuários que fazem uso relatórios na qual os desenvolvedores entrevistados são usuários finais.

Extraindo do MaxQda mais a fundo essa conexão, como apresentado na Figura 6 abaixo, nota-se os trechos de cada entrevista onde ressalta a forma como é a interação entre os desenvolvedores e clientes, eles reforçam que quando o usuário que está reportando é um cliente, muitas vezes os clientes que não sabem mexer na plataforma ou querem inserir um valor indevido em algum campo, como por exemplo inserir letras no campo para informar o cpf, que aceita apenas números, alguns relatos podem ser observados na figura abaixo:

Entrevista 1	Importância da comunicação informal	uma coisa assim em que você encontrou uma vez e de repente sumiu
Entrevista 5	Uso do software de forma inadequada	por exemplo uma pizza de calabresa com acréscimo de Catupiry, e em outras ferramentas para você adicionar a pizza de calabresa com catupiry, você precisa colocar aquela opção específica "com catupiry" , seria mais na interação do usuário que não sabe mexer na plataforma,
Entrevista 5	Identificação da falha através da interação	mais na interação do usuário que não sabe mexer na plataforma, o
Entrevista 6	Uso do software de forma inadequada	A gente estava com um teste no sonar de um determinado serviço, o usuário estava querendo passar um CPF que era "bananinha" e estava dando erro no serviço porque ele não tinha como consultar o Serasa para saber qual o score do "bananinha", mas sendo que nenhum usuário final no caso vai conseguir colocar na ferramenta o CPF dele como "bananinha" porque ele tem filtros para barrar isso lá no próprio sistema.
Entrevista 7	Uso do software de forma inadequada	que vai ter mais influência é o impacto que esse que esse bug causa para o usuário final.

Figura 6: Exemplo da relação entre trechos das entrevistas e os códigos mais citados na ferramenta MaxQDA

Para cada questão de pesquisa, apresentamos os resultados, juntamente com observações e análises de entrevistas. Temos nas transcrições trechos anônimos e ofuscado seu gênero para manter a privacidade dos participantes.

4.1 Como as interações entre desenvolvedores e usuários, que utilizam ferramentas de relatórios de bugs, influencia no entendimento e posteriormente na resolução de um bug ?

Um relatório que não descreve bem o cenário em que o bug está acaba dificultando o trabalho do desenvolvedor; este seria um exemplo para que se tenha a comunicação adicional com quem reportou, ao questionar o principal motivo de se entrar em contato o P2 ressalta *"Saber o que de fato aconteceu porque se a gente precisa reproduzir o cenário , a pessoa que viu o bug, que viu ele acontecendo é quem mais pode descrever, detalhar o que é que tá acontecendo para gente mesmo(...)".*

Pela experiência de como é trabalhoso investigar esses bugs através de logs e, muitas vezes, ser um problema pontual onde na máquina do desenvolvedor não é possível simular o erro. A maioria prefere que os usuários reportem o máximo de dados que puderem, entretanto há também a consciência de que nem sempre muitos dados vão trazer de fato a uma síntese do que se trata o bug relatado, como diz o P4 quando questionamos se quanto maior for o report de dados, melhor: "*...depende sempre tem um nível que às vezes dados demais atrapalha, eu acho que tem que ter um equilíbrio e saber reportar o que é de real interesse(...)*"

4.2 Como relatórios de bugs incompletos e/ou mal-escritos podem gerar impactos negativos na correção de uma falha?

Os relatórios mais difíceis de analisar, e por conseguinte resolver, seriam aqueles que vêm incompletos, principalmente aqueles em que o campo 'cenário' está em branco. falta do detalhamento, que muitas vezes é retratada apenas por uma captura de tela, é fundamental para que o desenvolvedor não tenha um progresso bem sucedido na análise deste bug, sendo assim entrando em contato com quem o reportou como saída para um bom entendimento do problema, quando questionado a respeito dos relatórios incompletos. P7 ressalta que o passo a passo é um dos campos que considera importante e que muitas vezes vem em branco ou incompleto, *"a busca por ela(passo a passo) vai depender da quantidade de informação que eu tenho(...), quando não encontro essas informações eu procuro quem reportou ou alguém para poder compreendê-lo."*

Ou então aqueles em que se tratam de um erro com cenário específico, na qual não possuem dados e acesso suficientes e com isso não conseguem analisar, ou então essa análise se torna um pouco confusa demandando mais tempo que o habitual das demais resoluções. P10 confirma quando diz que *"...uma descrição geral do problema é o que eu acho que auxilia muito no momento da correção porque ele já dá um direcionamento a mais do que você precisa para procurar no bug, então quer dizer se ele vem eles não vem preenchido esse campo e muitas vezes vêm branco, então você tem que fazer uma análise completa do log que mandam e é o que demanda muito mais tempo(...)."*

A dificuldade ao receber o relatório pode estar diretamente relacionada ao status final dele ser inválido, isso porque quando o desenvolvedor constata que as informações presentes nesse report não são suficientes e entra em contato com quem o reportou, o entendimento muitas vezes faz com que o desenvolvedor aponte que a execução daquele fluxo não está conforme descrito na estória por

exemplo, um trecho da fala do P3 sobre esse assunto ressalta"*...a gente entra em contato e diz: olha, esse bug aqui não é bug porque na estória a gente não tava vendo esse cenário e geralmente quando acontece isso o bug é invalidado transformamos isso em uma outra estória(...)".* Outro exemplo também seria quando o desenvolvedor entende que o usuário não estava inserindo credenciais válidos, não colocava nos campos que já tinha um tratamento de validação as informações coerentes para cada caso, o P6 traz em sua fala exatamente isso "*... o bug aparece nesse serviço porque na verdade é da integração com o sistema, há uma camada superior que filtra a entrada como, por exemplo a questão do valor ser uma string e não numeral(...)*".

5. DISCUSSÃO

OBSERVAÇÕES GERAIS

Numa perspectiva geral, os desenvolvedores consideram uma experiência boa e importante, boa no sentido de que cada um tem a sua maneira em específico em entrar em contato com quem reportou e essa tratativa não causa desconforto para ambas as partes. Quando necessário esse contato sempre ocorre de forma efetiva, isso não significa que a partir dessa interação o bug não será rejeitado ou invalidado e é justamente por isso que eles consideram importante; esse contato se faz necessário para entender o que o usuário tentou reproduzir para chegar no bug que ele relatou, pois muitas vezes esse relatório vem incompleto. Quando há o entendimento do problema e a verificação de que não é um bug de fato, o mesmo é invalidado após a resolução pontual com o usuário.

Relatórios não necessariamente precisam estar completos para que o desenvolvedor consiga chegar ao entendimento do problema reportado, vai depender bastante de quais campos estão preenchidos, muitas vezes com o *título* e a *descrição* do problema conseguem se fazer por completo, com isso notamos que há um grau de importância em relação a determinados campos, podemos citar que os campos mais relevantes seriam, o campo *cenário* por exemplo, que também se refere aos detalhes do bug contém informações importantes sobre onde o problema relatado foi encontrado, o campo *passo a passo* que se refere à descrição do bug fornece uma explicação precisa do problema observado, junto com as etapas necessárias para reproduzir o problema. Todas as informações presentes são importantes para ajudar a equipe de desenvolvimento a entender qual é o problema, quais ações precisam ser executadas para reproduzi-lo e qual deve ser o comportamento correto.

Com um cenário bem descrito, um passo a passo bem estruturado faz com que os bugs sejam solucionados no tempo esperado, para se chegar a este ponto faz-se necessário que se tenha um acordo de dados que devem estar de certa forma obrigatórios ou pelo menos de acordo com ambas as partes, acredito que estamos a um passo de cessar dores, essa investigação traz como conclusão principal que quanto mais comunicação e conhecimento do software, mais fácil será a experiência do usuário ao utilizá-lo e ao desenvolvedor aprimorá-lo.

O principal motivo para que um bug seja invalidado seria, quando o usuário não utiliza o software da forma correta, seja pela inserção de dados inválidos, até o não conhecimento prévio de como funciona o fluxo de execução , trazendo de sua perspectiva o fato do software está com o bug reportado, ao ser constatado que não é um bug o desenvolvedor pode vir a resolver pontualmente com o usuário e antes ou depois fazer a invalidação do relatório.

6. LIMITAÇÕES E AMEAÇAS À VALIDADE

Primeiramente, para reduzir o viés de amostragem, a pesquisa foi feita com alunos e ex-alunos da Universidade Federal de Campina Grande, que além de oferecer grande diversidade regional e cultural, todos entrevistados são empregados de empresas distintas, implicando assim trabalhos diferenciados de relacionamentos. Ainda sobre a diversidade da amostra, outra característica seria que a predominância da amostra é feminina, tendo minoria masculino, a escolha de enfatizar o gênero da amostra é proposital quando temos cada vez mais o aumento do número de mulheres no trabalho de T.I.*(Tecnico de Informatica)*[13]. Além disso, é impossível dizer que toda uma população ou a maioria de seus indivíduos pensam de uma determinada maneira com base nas respostas oferecidas pelos indivíduos entrevistados neste estudo. Em geral, todas as pesquisas representam uma ameaça à validade dos resultados, então precisamos encontrar maneiras de minimizar esses riscos.

Para isso, realizamos os seguintes procedimentos:

Em relação ao processo de entrevista, o roteiro não foi previamente disponibilizado aos participantes para evitar ideias. Antes de iniciar as entrevistas, precisamos explicar os conceitos envolvidos e os objetivos da pesquisa.

A transcrição, o processo de codificação e análise foram acompanhados semanalmente com outros pesquisadores e professores orientadores que buscavam complementar em outras pesquisas os resultados desta, os resultados de cada um expostos para discussão, e quando havia divergência de entendimento ou sugestões de melhorias, discutia-se até chegar em um consenso.

Foi realizada também uma revisão da literatura antes da fase de codificação e criação dos *memos*. Embora possa ter facilitado a execução das entrevistas e a identificação dos achados, pode também introduzir um viés de interpretação, com isso os resultados e suas discussões foram revisadas para minimizar esse tipo de viés. Para garantir a consistência dos resultados, a literatura foi revisada novamente após a definição inicial dos achados e quando necessários, os códigos foram revisados e aprimorados.

7. CONCLUSÃO

Devido a facilidade da interação entre os desenvolvedores e os usuários é possível notar que as empresas têm adotado maneiras para que essa comunicação seja fácil para ambas as partes, entretanto há ainda a necessidade de melhorar no quesito da obrigação e concordância dos dados que são transitados.

O foco da pesquisa era unificar as análises pela maioria dos autores na questão das abordagens adotadas e trazer da perspectiva dos desenvolvedores como eles lidam ao recorrer às interações com os usuários. Conseguir alcançar as melhores abordagens ainda é um desafio devido à maneira de como de quem está no papel de quem usa o relatório para reportar o que ele considera naquele momento um bug, já que não existe ainda um método unificado que auxilie esses usuários.

Para firmar os resultados obtidos, uma proposta para trabalhos futuros seria realizar uma pesquisa investigativa de como esses relatórios são projetados de acordo com a dinâmica proposta em cada empresa a fim de propor como experimento a obrigatoriedade de indicadores como o campo cenário, cuja é aquele mais apontado na pesquisa como o principal, pois seria o indicador do fluxo executado pelo usuário. Outra ideia também seria como melhorar o contato do usuário como foco o preenchimento correto desses relatórios, a partir de como eles são exibidos para os mesmos, utilizar de *placeholders*[11] que são atributos onde nos campos de qualquer formulário, podem trazer o entendimento de como inserir as informações de cada campo de forma correta.[12]

REFERÊNCIAS

[1] Thomas Zimmermann, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schro¨ ter, and Cathrin Weiss What Makes a Good Bug Report? IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36, NO. 5, SEPTEMBER/OCTOBER 2010

[2] Software Testing Help, *What Is Defect/Bug Life Cycle In Software Testing?* https://www.softwaretestinghelp.com/bug-life-cycle/

[3] Thomas Peham, "*What is a bug report? The ins and outs of bug reports* https://usersnap.com/blog/what-is-a-bug-report/

[4] Merrin Hughes, Bug Reporting: What is a bug report and how to create good bug reports https://bugherd.com/blog/bug-reporting/

[5] Affde, A anatomia de um bom relatório de bug, https://www.affde.com/pt/good-bug-report.html

[6] Susan Butler, Colin Yallop. Macquarie Dictionary, macquariedictionary.com.au

[7] George Silva, Como escrever bons relatórios de bugs, https://geoadmin.com.br/como-escrever-bons-relatorios-de-bugs/

[8] Youtube, plataforma de compartilhamento de vídeos, https://www.youtube.com

[9] Hugo rocha, O que é Pesquisa Qualitativa, tipos, vantagens, como fazer e exemplos https://blog.klickpages.com.br/o-que-e-pesquisa-qualitativa/

[10] Gupy, O que é entrevista estruturada, vantagens e como fazer uma? https://www.gupy.io/blog/entrevista-estruturada#:~:text=Entrevista%20estruturada%20%C3%A9%20um%20m%C3%A9todo,de%20forma%20criteriosa%20e%20justa.

[11] Rodrigo Hahn, 10 ótimos sites de placeholder para usar em seus projetos web https://king.host/blog/2021/04/placeholder/

[12] SapoUx, Regras de usabilidade para Formularios,https://ux.sapo.pt/usabilidade/web/formularios/

[13]ti.inside,https://tiinside.com.br/16/12/2021/aumenta-o-numero-de-mulheres-no-mercado-de-ti