



Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Verificação e Validação de Software - 206580

Proposta de Processo de Verificação no Projeto Falko

Autores

Fabíola Malta Fleury
Lucas de Araujo Martins
Marlon Mendes Ciriatico Guimaraes
Matheus de Sousa Bernardo
Matheus Vitor Costa Joranezon
Matheus Richard Torres Gomes de Melo
Thalisson Barreto de Melo Silva

Professor

Ricardo Ajax Dias Kosloski

Brasília, DF
2017



1. Tema	3
1.1. Pesquisa-Ação	3
2. Problema	3
3. Objetivo	5
3.1 Objetivos Gerais	5
3.2 Objetivos Específicos	5
4. Questões de pesquisa	5
5. Referencial teórico	6
6. Proposta de solução	7
7. Resultados Obtidos	10
7.1. Métricas coletadas	11
Sprint 4	12
Sprint 5	13
7.2. Entrevistas	14
8. Análise de Resultados	14
9. Conclusões	15
Referências bibliográficas	16
APÊNDICE	17
APÊNDICE A - GQM do Objetivo de Medição 03	18
Objetivo de Medição	18
Questões	18
Métricas	19

1. Tema

Segundo o SWEBOK v3.0 (2014), qualidade de software pode ser definida como o grau em que um produto de software atende aos requisitos estabelecidos. Entretanto, a qualidade depende do grau em que esses requisitos estabelecidos representam com precisão as necessidades, desejos e expectativas das partes interessadas.

Nesse contexto, Verificação e Validação se encaixa como o processo de aferição de que um software corresponde às suas especificações e de que atende seus propósitos. Estas disciplinas garantem não apenas que os desenvolvedores de software elaborem o produto corretamente, mas que elaboram o produto correto.

1.1. Pesquisa-Ação

Como apresentado por Tripp (2015), a pesquisa-ação é um tipo de investigação-ação, possuindo o intuito de aprimorar uma prática agindo em campo e a investigando. Para isso, são feitos planejamentos, implementações e avaliações sobre mudanças. Essas avaliações podem ser consideradas para melhorar tanto o que está se investigando quanto a investigação em si.

Este trabalho propõe a pesquisa-ação dentro do contexto de um projeto desenvolvido no ambiente de graduação, com o intuito de aplicar conhecimentos da área de verificação e validação de *software* para melhorar a qualidade do processo de desenvolvimento e, por consequência, do código resultante, de forma que este atenda às especificações anteriormente levantadas.

2. Problema

Durante o trabalho conjunto nas disciplinas de Métodos de Desenvolvimento de Software e Gestão de Portfólios e Projetos de Software (GPP/MDS) acontece o desenvolvimento de um produto de software, na primeira parte da disciplina utiliza-se o RUP e na segunda parte é utilizado o Scrum e o XP.

Para esta pesquisa ser desenvolvida, acompanhou-se o desenvolvimento do Falko (<https://github.com/fga-gpp-mds/Falko-2017.2-BackEnd/wiki>). Falko é um dos projetos em realização durante o segundo semestre de 2017 nas disciplinas de GPP/MDS com intuito de construir uma ferramenta para gestão de projetos de software desenvolvidos seguindo a metodologia ágil. O grupo é formado por cinco integrantes cursando a disciplina GPP e sete cursando MDS.

O projeto FALKO está sendo desenvolvido utilizando metodologias ágeis baseadas em scrum, XP e Kanban. Os papéis atribuídos dentro do projeto são: scrum master, product owner e time de desenvolvimento. O time trabalha em sprints com duração de uma semana, possuindo os eventos do scrum de: planejamento da sprint, desenvolvimento da sprint, revisão da sprint, retrospectiva da sprint, reunião diária. O fluxo de trabalho da equipe é

documentado em um quadro kanban presente no repositório do *github*. Além disso, métricas para qualidade foram definidas a partir do modelo GQM (*Goal Question Metric*) (Solingen, 1999). Um diagrama representativo das métricas escolhidas pode ser observado na Figura 1.

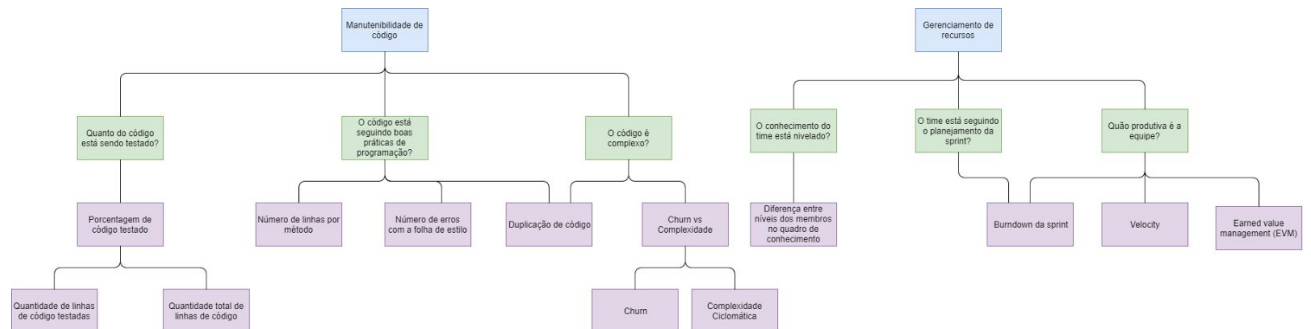


Figura 1 - Diagrama GQM Falko

Ao decorrer do desenvolvimento do projeto especialmente na segunda parte da disciplina foi detectado a possibilidade de introdução de erros em incrementos do produto. Portanto, foi levantado por meio de um gráfico de espinha de peixe que esses erros podem ser causados por três grandes grupos e todos eles estão relacionados à verificação como mostra a imagem a seguir.

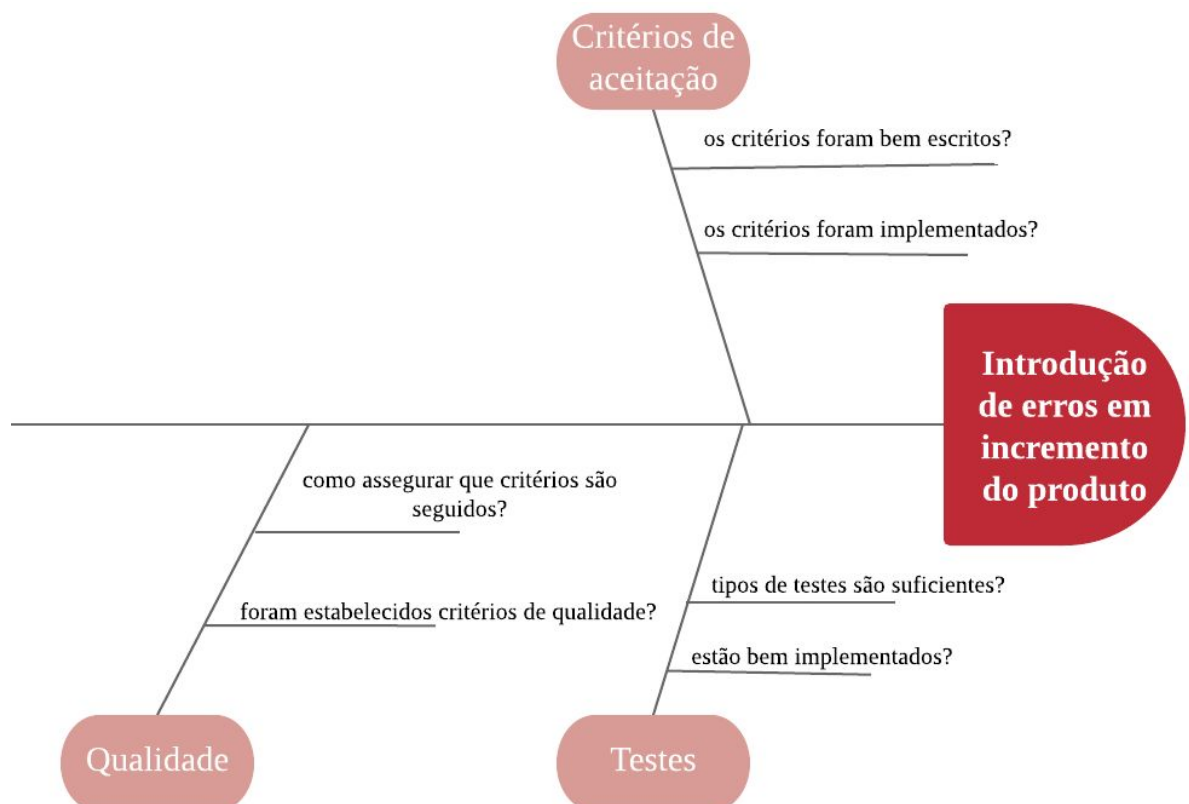


Figura 2 - Diagrama espinha de peixe

A equipe tem como propósito propor atividades de verificação relacionadas aos problemas identificados, com o foco em qualidade e em testes.

3. Objetivo

3.1 Objetivos Gerais

O objetivo dessa solução é estabelecer um processo que assegure a verificação dos incrementos do produto em uma equipe ágil para evitar a introdução de erros na aplicação.

3.2 Objetivos Específicos

Essa solução tem como objetivo criar um processo de verificação que permita auxiliar equipes ágeis a evitar introduzir erros em incrementos do produto. Esse objetivo será alcançado por meio de alguns objetivos específicos.

- Propor um processo de revisão de código para verificar os testes daquele incremento
- Estabelecer um processo para detectar erros causados por falta de critérios de qualidade do produto
- Estabelecer um processo para auxiliar o desenvolvimento de testes e qualidade de entregas

Será verificado se esses processos alcançam os resultados esperados comparando-se as métricas de qualidade do produto e também qualitativamente por meio de entrevistas com os envolvidos no projeto Falko.

4. Questões de pesquisa

- **Questão 1:** Como garantir que os testes feitos são significativos para o desenvolvimento?
- **Questão 2:** Como guiar o desenvolvimento de testes?
- **Questão 3:** Como assegurar que os critérios de qualidade escolhidos estão sendo seguidos e estão sendo significativos?

5. Referencial teórico

Para esta pesquisa, é importante destacar conceitos ligados a área de testes de *software* e também sobre a metodologia ágil *scrum*. Estes termos são utilizados como base dentro do estudo de caso estudado para implementação da solução proposta.

De acordo com Meyers (2011), testes são feitos para encontrar erros. Existem diversas técnicas de testes que podem ser divididas em realizadas por máquinas e realizados por humanos. Exemplos de técnicas realizadas por pessoas são: auditorias, revisões em pares, *walkthrough* e inspeções. Testes feitos em máquinas, dentre muitos tipos podem ser: testes de regressão, testes unitários, testes de sistema e testes funcionais.

O Guia Scrum (Sutherland, 2013) define um processo de desenvolvimento que tem como valores a transparência, inspeção e adaptação e como base papéis, artefatos e eventos. O time *scrum* é a equipe de projeto definida em três diferentes papéis: o *scrum master*, responsável por garantir a implementação da metodologia, o time de desenvolvimento, responsável pela entrega de artefatos e o *product owner* que possui a visão do cliente e gerencia o *product backlog*. Os eventos são: a *sprint*, tempo utilizado para criar um incremento de produto, o planejamento da *sprint*, reunião para definir o objetivo da *sprint*, reunião diária, para verificar o progresso da *sprint* e possíveis impedimentos, revisão da *sprint*, para verificar o incremento produzido e retrospectiva da *sprint* para verificar o processo utilizado. Os artefatos do *scrum* são: *product backlog*, requisitos do projeto, *sprint backlog*, requisitos da *sprint* e o incremento que é aquilo entregue ao final de cada *sprint*.

O estudo de caso conduzido no artigo “*Product Backlog Rating: A Case Study On Measuring Test Quality In Scrum*” (Kayes, 2016) apresenta a utilização de uma adaptação da metodologia *Scrum* utilizada na empresa *SoftwarePeople* pelos suas equipes de desenvolvimento e uma métrica relacionada a qualidade de teste.

No artigo, os artefatos gerados pelos times são: gráfico *burndown*, *sprint backlog* e *dashboard*. Os papéis: *scrum master*, *product owner*, time de desenvolvimento e engenheiro de garantia da qualidade. Observa-se que o papel engenheiro de garantia da qualidade não é proposto dentro do Guia Scrum (Sutherland, 2013) porém tem o objetivo de garantir a qualidade das entregas, participando ativamente do desenvolvimento, delimitando histórias e entendimentos dos requisitos, ajudando a construir os testes, apresentando para um cliente uma versão do *software* durante a *sprint*, chamado de *sneak peek*, para caso sejam necessárias mudanças nos requisitos, criando o plano de teste para realização de teste de regressão na entrega do incremento ao final da *sprint*.

Testes de aceitação de usuário e *deploy* são realizados por equipes separadas que não fazem parte do time *scrum*.

6. Proposta de solução

Com o intuito de prevenir a introdução de erros no incremento do produto entregue ao cliente do projeto Falko, propõe-se seguir um processo de verificação detalhadamente definido. Assim, são sugeridas mudanças no processo de desenvolvimento e no gerenciamento de qualidade do time, baseando-se no estudo de caso apresentado por Kayes (2016), adaptando o que foi observado para o contexto do projeto sendo desenvolvido e também no que é proposto por Myers (2011, p.21) no que diz respeito à técnicas de testes não baseada em computadores. Serão conduzidas então, ao final da pesquisa, entrevistas com os integrantes da equipe de desenvolvimento quanto às diferenças notadas com as implementações propostas e serão estudadas também as métricas coletadas.

A solução propõe três mudanças:

- Introdução do papel engenheiro de garantia de qualidade;
- Novo objetivo de medição;
- Mudança na revisão antes da entrega de uma história;

O processo de desenvolvimento utilizado atualmente pela equipe Falko é composto por três papéis: *Scrum master*, *product owner* e time de desenvolvimento. O acréscimo do papel engenheiro de garantia da qualidade tem como objetivo assegurar a qualidade dos artefatos entregues, trabalhando em paralelo à equipe de desenvolvimento. O engenheiro de qualidade é responsável por escrever uma versão concisa de casos de testes, que é então entregue para que os desenvolvedores possam aplicá-las, e acompanha o desenvolvimento das histórias de usuário e do incremento de cada sprint. Assim, a inclusão deste papel proporciona suporte para revisões de código e escrita de testes (KAYES, 2016). O que é importante dentro do contexto que está sendo estudado, tendo em vista que metade da equipe de desenvolvimento tem pouco ou nenhum contato com testes e pode então contar com o apoio deste participante.

Também é modificada como é feita a entrega de cada história. O scrum master é responsável por aceitar a história no incremento da sprint, porém não há descrição dessa atividade, então criou-se um processo formalizado de revisão que será realizado com apoio de *checklist*.

O novo processo pode ser observado na Figura 2 com detalhamento do subprocesso de desenvolvimento da sprint na Figura 3.

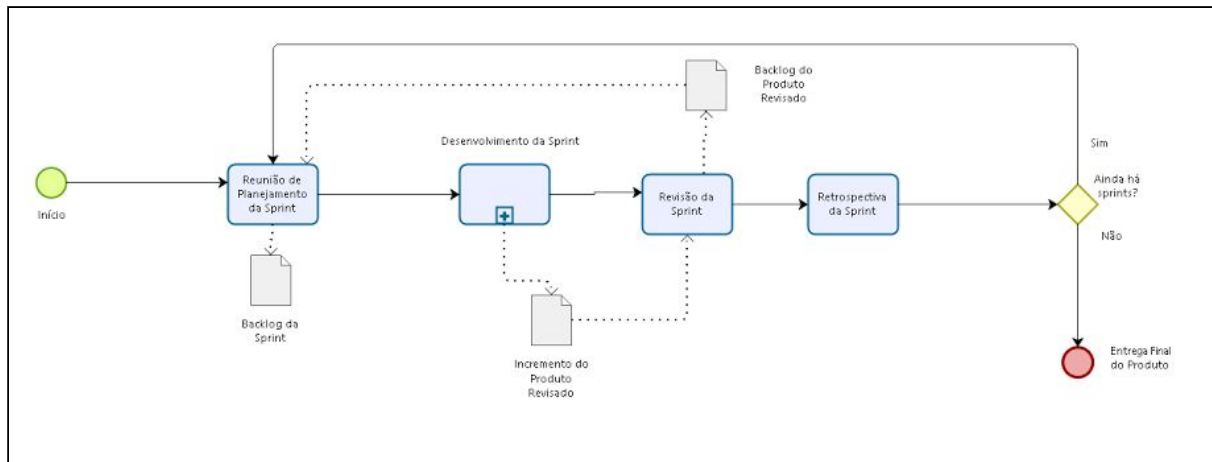


Figura 3 - Processo de desenvolvimento proposto

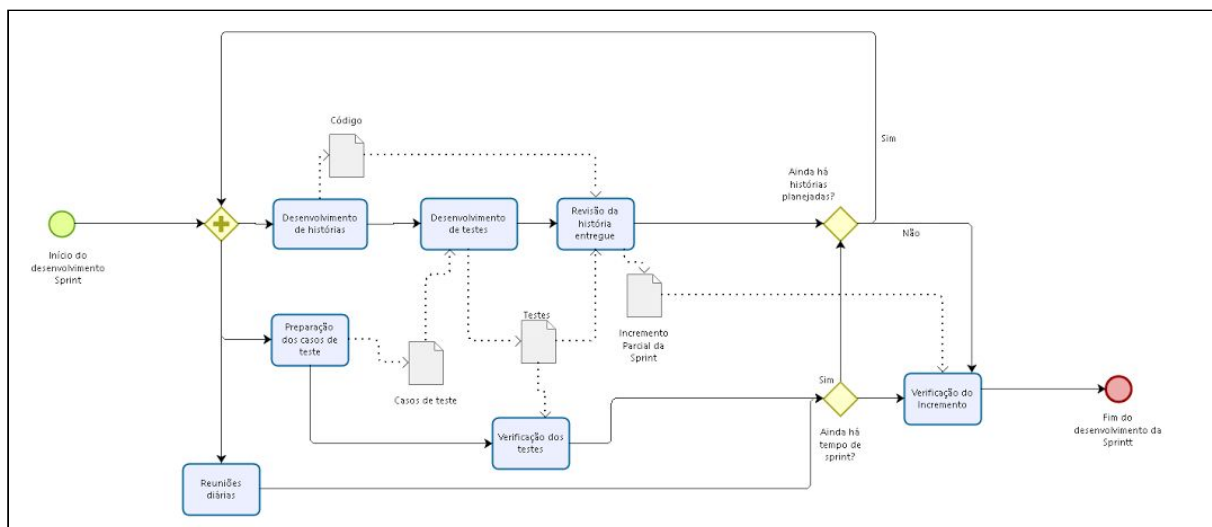


Figura 4 - Subprocesso de desenvolvimento da sprint

Das atividades propostas no processo, é importante destacar que a revisão da história entregue utiliza a funcionalidade de *Pull Request (PR)* da ferramenta *github* como auxílio. O *PR* permite que tanto ferramentas automatizadas quanto pessoas comentem no pedido e deem sugestões ou o recusem. Ao terminar de implementar a história e seus respectivos testes, o desenvolvedor submete um *PR* para poder acrescentar seu código no que será o incremento da sprint. As ferramentas automatizadas de teste de integração são ativadas e avisam se aquele pedido pode ou não ser integrado. A tabela 1 mostra a revisão realizada após as ferramentas automatizadas cumprirem seu papel.

Tabela 1 - Atividade: Revisão da história entregue

Descrição	Depois da submissão do código da história por meio do <i>PR</i> do github, caso seja aceito pelas ferramentas automatizadas, o scrum master revisa esse código utilizando a <i>checklist</i> da figura 5. Caso não esteja de acordo,
------------------	--

	ele comenta o tópico criado pelo PR solicitando ao desenvolvedor responsável que faça as alterações necessárias, podendo então, após correções, aceitá-lo para tornar-se parte do incremento da sprint.
Artefatos de Entrada	Código e testes da história
Artefatos Gerados	Incremento parcial da sprint
Responsável	Scrum Master

- ☐ A build foi feita com sucesso
- ☐ Cada critério de aceitação possui ao menos um teste de aceitação
- ☐ Foram implementados todos os casos de testes
- ☐ A cobertura de testes manteve-se ou aumentou
- ☐ Não existem erros do linter (folha de estilo)
- ☐ A complexidade ciclomática dos métodos não passa de 6
- ☐ Não existem métodos duplicados
- ☐ Os métodos possuem 30 linhas ou menos

Figura 5 - Checklist da Revisão da história

Além disso também foram revisadas as métricas coletadas pela equipe por meio do modelo GQM e proposto mais um objetivo ligado à qualidade seguindo o processo proposto por Van Solingen (1999). O documento desenvolvido para este objetivo encontra-se no [apêndice A](#). As métricas relacionadas à este objetivo estão ligadas a qualidade dos testes implementados, sendo avaliadas pelo engenheiro de garantia de qualidade.

No processo proposto, existem atividades que serão realizadas pelo novo papel, engenheiro de garantia da qualidade como estão destacados na Figura 6. Estas atividades são voltadas para coleta das métricas propostas pelo novo objetivo e também para apoio na realização dos testes. Os casos de testes serão descritos de forma resumida em formato de *checklist*, sendo entregue para que os desenvolvedores realizem os testes unitários. Depois de terminados os testes e submetidos para avaliação pelo *scrum master* em conjunto ao código, o engenheiro também estará responsável por verificar estes testes.

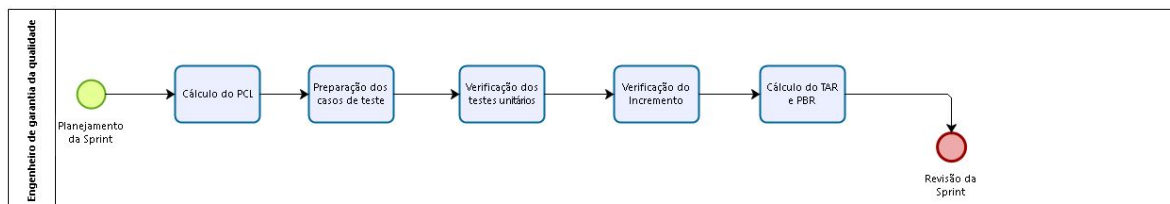


Figura 6 - Subprocesso do Engenheiro de garantia da qualidade

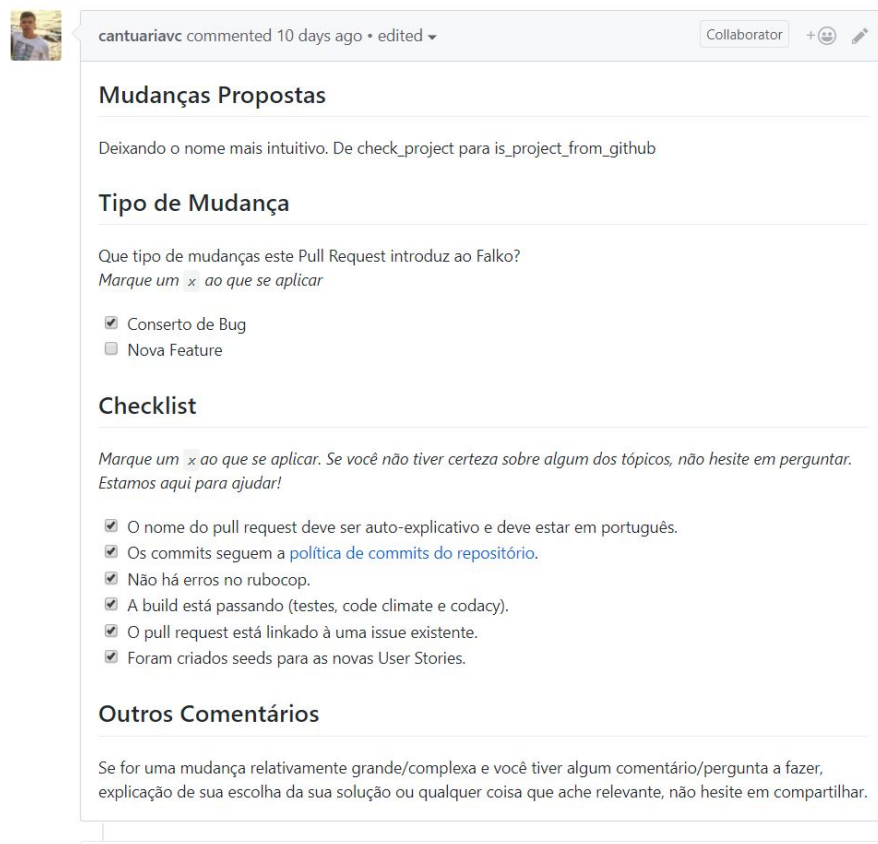
A equipe do projeto Falko trabalhou durante três sprints utilizando sua organização anterior, é proposto que trabalhem duas semanas com as alterações aqui sugeridas, sendo as métricas monitoradas e entrevistas feitas com os membros ao final da implementação para que se possa avaliar se a proposta de solução foi de fato efetiva para o seu contexto.

7. Resultados Obtidos

Antes da implementação do trabalho proposto na primeira entrega da disciplina de Verificação e Validação de *Software*, fez-se um treinamento do time GPP/MDS em relação à necessidade de testes, o que seria o engenheiro de garantia de qualidade e como as mudanças propostas seriam aplicadas.

Assim, durante as duas sprints que foram implementadas essas mudanças, houve o acompanhamento por esse papel, a coleta de métricas e ao final uma entrevista com a equipe de desenvolvimento para auxiliar na obtenção de resultados.

Na figura 7 observa-se a *checklist* utilizada durante a revisão de histórias de usuário enviadas por meio de *pull requests* do github. Ela permitiu uma orientação no diálogo entre revisor e a pessoa que submeteu o código.



The image shows a GitHub Pull Request form for user 'cantuariavc'. The form is titled 'Mudanças Propostas' and includes a section for 'Tipo de Mudança' with checkboxes for 'Conserto de Bug' (checked) and 'Nova Feature'. Below this is a 'Checklist' section with several items, all of which are checked: 'O nome do pull request deve ser auto-explicativo e deve estar em português.', 'Os commits seguem a política de commits do repositório.', 'Não há erros no rubocop.', 'A build está passando (testes, code climate e codacy).', 'O pull request está linkado à uma issue existente.', and 'Foram criados seeds para as novas User Stories.'. The form also includes a section for 'Outros Comentários'.

cantuariavc commented 10 days ago • edited ▾ Collaborator + 😊 ✎

Mudanças Propostas

Deixando o nome mais intuitivo. De `check_project` para `is_project_from_github`

Tipo de Mudança

Que tipo de mudanças este Pull Request introduz ao Falko?
Marque um ☒ ao que se aplicar

☒ Conserto de Bug
☐ Nova Feature

Checklist

Marque um ☒ ao que se aplicar. Se você não tiver certeza sobre algum dos tópicos, não hesite em perguntar. Estamos aqui para ajudar!

☒ O nome do pull request deve ser auto-explicativo e deve estar em português.
☒ Os commits seguem a [política de commits do repositório](#).
☒ Não há erros no rubocop.
☒ A build está passando (testes, code climate e codacy).
☒ O pull request está linkado à uma issue existente.
☒ Foram criados seeds para as novas User Stories.

Outros Comentários

Se for uma mudança relativamente grande/complexa e você tiver algum comentário/pergunta a fazer, explicação de sua escolha da sua solução ou qualquer coisa que ache relevante, não hesite em compartilhar.

Figura 7 - Utilização da checklist no envio de um *pull request*

Os casos de testes da US12 - Manter Retrospectiva de Sprint estão documentados na figura 8. Estes foram escritos de forma simplificada, tendo em vista que são apenas orientações para que os responsáveis pela história estejam satisfazendo os diferentes cenários de testes unitários para certificar-se que o que foi entregue foi realizado com qualidade. Esta estrutura foi replicada em todas as histórias de usuário desenvolvidas nas *sprints*. Os responsáveis

pela realização dos testes marcaram como feito cada item após sua conclusão e durante a entrega o revisor utilizou a lista para certificar-se que a implementação proposta foi realizada.

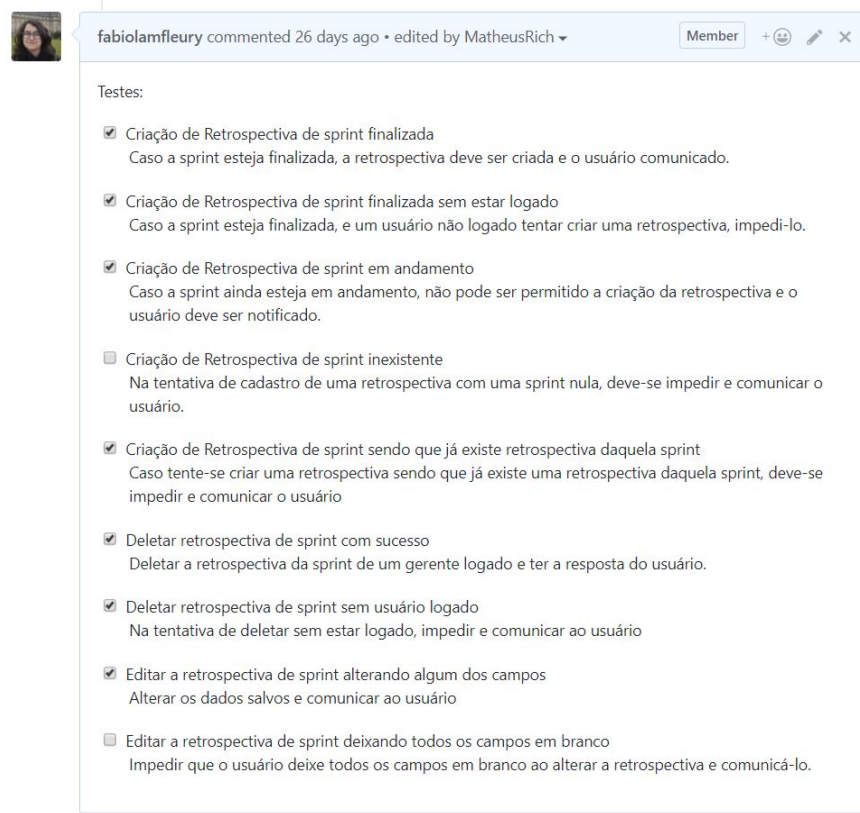


Figura 8 - Casos de testes de US 12 simplificados

7.1. Métricas coletadas

A cada sprint a pessoa designada ao papel de engenheiro de garantia de qualidade foi responsável por, inicialmente, avaliar a complexidade de cada item priorizado do *backlog* da sprint (PCL) e ao final, os testes entregues de cada história (TAR), sendo os dois utilizados para calcular a qualidade de testes entregues à cada sprint (PBR). Lembre-se que as três métricas utilizadas são índices que possuem valores que variam de 1 a 5. A primeira sprint avaliada (Sprint 4) possuiu como resultado o *Product Backlog Rating (PBR)* de 2,18, considerado ruim, e na segunda sprint (Sprint 5) o *PBR* teve o valor de 2,29, também considerado ruim pelo que foi estabelecido anteriormente pelo plano de análise do GQM. Observa-se porém, um pequeno aumento no índice de uma sprint para outra. Abaixo, estão descritas de forma detalhada as métricas coletadas.

Sprint 4

Na sprint 4 foram priorizadas as histórias que podem ser observadas na tabela 2, totalizando 15 pontos planejados.

Tabela 2 - Backlog da Sprint 4

Identificador	História
US 07	Visualizar GPA
US 12	Manter retrospectiva de sprint
US 14	Manter história
US 22	Manter issues

Nas tabelas 3 e 4 foram calculados respectivamente o PCL e o TAR de cada história de usuário e foram utilizados estes dados para o cálculo do PBR da sprint com resultado de 2,18. Observou-se grande dependência das histórias entre si e também a utilização de conexão com sistemas externos, a US 22 por exemplo utiliza o sistema Falko para criar issues no site do *github* fatores estes que contribuíram para o aumento da complexidade dos itens do *backlog*.

Tabela 3 - Product Complexity Level (PCL) das histórias da Sprint 4

Fator de complexidade	US 07	US 12	US 14	US 22
Necessidade de dados de teste	2	4	3	3
Estimação do teste	3	4	3	3
Dependência interna	5	4	3	2
Dependência externa	5	5	4	5
Complexidade de execução do teste	2	3	2	4
PCL	3,4	4	3	3,4

Tabela 4 - Test Assessment Rate (TAR) das histórias da Sprint 4

Fator de complexidade	US07	US 12	US 14	US 22
Contagem de bugs	N/A	N/A	N/A	N/A
Mudança de Requisitos	5	2	5	1
Confiança nos testes	1	1	2	1
TAR	3	1,5	3,5	1

Sprint 5

Durante a sprint 5 foram desenvolvidas as histórias explicitadas na tabela 5, que somam 12 pontos planejados.

Tabela 5 - Backlog da Sprint 5

Identificador	História
US 06	Designar membros
US 11	Manter revisão de sprint
US 15	Pontuar histórias
US 22	Manter issues

Os índices PCL e TAR das histórias da sprint podem ser visualizados nas tabelas 6 e 7 resultando em um PBR da sprint de 2,29.

Tabela 6 - Product Complexity Level (PCL) das histórias da Sprint 5

Fator de complexidade	US 06	US 11	US 15	US 22
Necessidade de dados de teste	2	3	1	3
Estimação do teste	2	3	2	3
Dependência interna	1	2	5	2
Dependência externa	5	5	5	5
Complexidade de execução do teste	2	2	2	4
PCL	2,4	3	3	3,4

Tabela 7 - Test Assessment Rate (TAR) das histórias da Sprint 5

Fator de complexidade	US 06	US 11	US 15	US 22
Contagem de bugs	N/A	N/A	N/A	N/A
Mudança de Requisitos	1	3	4	5
Confiança nos testes	2	2	1	2
TAR	1,5	2,5	2,5	2,5

7.2. Entrevistas

Ao final da realização das duas sprints com as alterações propostas pela equipe de Verificação e Validação de software, foram feitas entrevistas com a equipe de desenvolvimento do Falko, para entender melhor o impacto das mudanças. Foram feitas perguntas relacionadas a dificuldade em incorporar as mudanças em um projeto já em andamento, as vantagens e desvantagens percebidas bem como sugestões de melhoria para o processo proposto.

A maior vantagem percebida pelos membros foi a formalização das revisões por meio de uma *checklist* em *pull requests* do github. Anteriormente, não havia um retorno sobre a revisão realizada e isso além de assegurar que a revisão considerasse diversos pontos tidos como importantes para qualidade, permitiu aos membros perceberem erros realizados e poderem corrigi-los de acordo com o necessário. Também foi relatado um aumento de segurança ao utilizar os casos de testes simplificados e saber que existia um papel para auxílio no desenvolvimento de testes.

8. Análise de Resultados

A construção deste trabalho permitiu o acompanhamento do desenvolvimento de um projeto de *software*, por meio de uma perspectiva de qualidade. Assim, foi proporcionado um foco diferente daquele que os integrantes do grupo estão habituados, com objetivos que acabam se diferenciando dos objetivos traçados pela equipe de desenvolvimento.

Nota-se que a equipe do projeto Falko possui pouca experiência nas áreas de desenvolvimento de projetos e testes, o que acarretou na dificuldade de cumprimento do processo e também na realização de testes, evidenciados pelos resultados baixos coletados das métricas.

O trabalho sugeriu a alteração do processo utilizado pela equipe e dado o acompanhamento que foi realizado, percebeu-se que o processo documentado anteriormente não foi de fato seguido, o que pode impactar diretamente na qualidade do produto entregue. Problemas foram encontrados no cumprimento dos eventos *timed-boxed* do *Scrum*, a revisão da sprint não proporcionou de fato a revisão do incremento entregue na sprint e existiam membros utilizando tempo dessa reunião e da retrospectiva da sprint para desenvolvimento de código, na documentação dos requisitos, nem todas as histórias foram escritas em voz de usuário e as histórias técnicas nem sempre eram documentadas e priorizadas para as sprints com a alocação necessária, não participando assim na medição do PBR, já que não ficava claro a que sprint pertenciam, sendo introduzidas às vezes já no decorrer dela.

Apesar do retorno positivo dos membros em relação ao engenheiro de garantia de qualidade, houve certa resistência ao novo papel por se tratar de alguém que não participava anteriormente da equipe, então a interação com a equipe não foi tão eficiente. Ainda, as pessoas que realizaram este papel também encontraram dificuldade na coleta das métricas propostas no [apêndice](#) tendo em vista que elas requerem a análise de diversos fatores de qualidade e complexidade de requisitos.

9. Conclusões

A proposta deste trabalho foi adaptar o processo do *Scrum* de modo a assegurar a qualidade dos testes automatizados e evitar a introdução de erros no incremento do produto ao final de cada sprint.

Entende-se que este trabalho conseguiu propor um processo de verificação para equipes ágeis por meio da introdução do papel do Engenheiro de Garantia de Qualidade, da elaboração de *checklists* para a realização dos testes e submissão de *Pull Requests*, e a coleta de métricas de qualidade. A métrica do PBR foi trabalhada para avaliar a significância dos testes desenvolvimentos e permitir decisões para sua melhoria.

Entretanto, a aplicação das modificações propostas pelo projeto à equipe de GPP/MDS em questão apresentou alguns obstáculos, tais como a falta de experiência da equipe, o que dificultou a coleta das métricas propostas, a falta de compromisso com a qualidade e a introdução do processo de verificação durante o decorrer do projeto, o que fez com que o papel do Engenheiro de Garantia de Qualidade fosse subutilizado pela equipe. Por causa dos fatores indicados, o processo não foi aplicado de forma satisfatória.

Apesar disso, a aplicação das *checklists* foi considerada positiva pela equipe, pois proporcionou um *feedback* importante para a equipe de desenvolvimento sobre o incremento recém-entregue, além de proporcionar uma possibilidade de auto-avaliação do fragmento do produto entregue pela própria equipe.

Com base nesta avaliação, pode-se concluir que o processo de verificação apresentado não é indicado para equipes inexperientes com a tecnologia ou metodologia, e, para que tenha sucesso, não pode ser inicializado no decorrer do processo de desenvolvimento, pois estas práticas se mostraram ineficazes. Por outro lado, a verificação utilizando *checklists* se mostrou muito positiva para este tipo de equipe. Estas constatações não garantem que o mesmo processo não seria eficaz no contexto de uma equipe madura e sendo aplicado no início do ciclo de vida do projeto. Deixamos este caso como sugestão para pesquisas futuras.

Referências bibliográficas

Bourque, Pierre; Fairley, Dick. **SWEBOK 3.0 Guide to the Software Engineering Body of Knowledge**. [S.l.]: IEEE Computer, 2014

TRIPP, David. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, v. 31, n. 3, 2005.

SUTHERLAND, Jeff; SCHWABER, Ken. The scrum guide. **The Definitive Guide to Scrum: The Rules of the Game**. **Scrum. org**, 2013.

KAYES, Imrul; SARKER, Mithun; CHAKARESKI, Jacob. Product backlog rating: a case study on measuring test quality in scrum. **Innovations in Systems and Software Engineering**, v. 12, n. 4, p. 303-317, 2016.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. **The art of software testing**. John Wiley & Sons, 2011.

VAN SOLINGEN, Rini; BERGHOUT, Egon. **The Goal/Question/Metric Method: a practical guide for quality improvement of software development**. McGraw-Hill, 1999.

APÊNDICE

APÊNDICE A - GQM do Objetivo de Medição 03

Objetivo de Medição

Tabela 1 - Objetivo de Medição 03

Analisar	código
para o propósito de	melhorar
com respeito a	detecção de erros
do ponto de vista	do time de projeto
no contexto de	Falko (GPP/MDS)

Questões

A partir da construção do *Abstraction Sheet* foram estabelecidas as questões e métricas da Figura 2.

Objeto	Propósito	Foco de Qualidade	Perspectiva
Código	Melhorar	Detecção de erros	Time de desenvolvimento

Foco de Qualidade <ul style="list-style-type: none">- Qualidade dos testes- Cobertura de testes	Fatores de variação <ul style="list-style-type: none">- Conhecimento em testes da equipe- Subjetividade de critérios de qualidade de testes
Hipótese de Baseline <ul style="list-style-type: none">- Cobertura de testes em ao menos 50%- Qualidade não muito boa de testes, tendo em vista que os membros não estão acostumados a testar.	Impacto dos fatores de variação <p>O conhecimento em testes da equipe, que possui membros não acostumados a testar, pode acarretar na diminuição da qualidade e cobertura, enquanto a qualidade pode ter um resultado diferente a depender do avaliador.</p>

Figura 1 - Abstraction Sheet do Objetivo de Medição 03

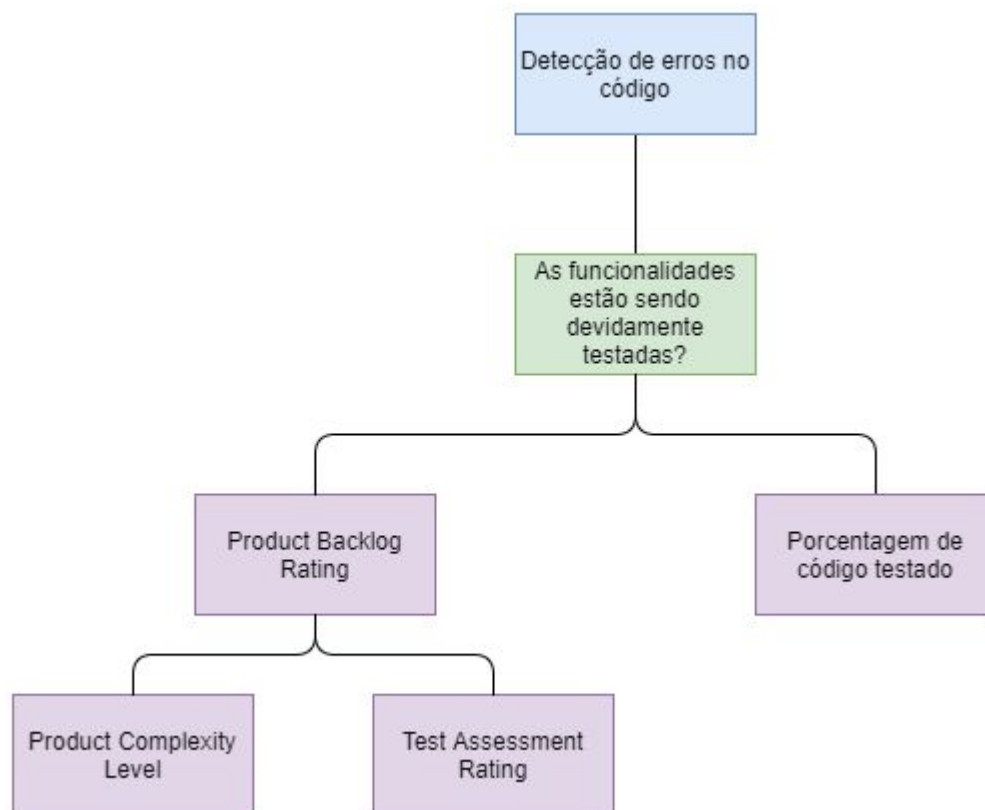


Figura 2 - Diagrama GQM do Objetivo de Medição 03

Métricas

As métricas das tabelas 2 e 3 são utilizadas para cálculo da métrica na tabela 4. As tabelas são detalhamentos das métricas, bem como suas coletas e respectivas análises.

Tabela 2 - Métrica PCL

Métrica	<i>Product Complexity Level (PCL)</i>
Objetivo de Medição	Detecção de erros no código
Descrição	É uma nota dada à dificuldade de se realizar testes em um item do product backlog, levando em consideração os seguintes fatores: Necessidade de dados de teste, estimação do teste, dependência interna, dependência externa e complexidade de execução do teste. Estes fatores são avaliados pelo engenheiro de garantia de qualidade.

Fórmula	$PCL_k = \frac{\sum_{i \in P} FP_i * W_i}{\sum_{i=1}^{i \in P} W_i}$, sendo FP os fatores de complexidade (descritos acima) e W o peso de cada um deles.
Escala	Racional
Coleta	Engenheiro de garantia de qualidade Periodicidade do Evento: Ao início de cada sprint.
Procedimento	O cálculo é feito manualmente pelo responsável da coleta.
Análise	Métrica auxiliar
Providência	Ele deve ser usado, juntamente com o <i>Test Assessment Rating</i> para o cálculo da métrica <i>Product Backlog Rating</i> .

Tabela 3 - Métrica TAR

Métrica	<i>Test Assessment Rate (TAR)</i>
Objetivo de Medição	Detecção de erros no código
Descrição	É uma nota dada ao sucesso dos testes em relação aos seguintes fatores: Contagem de bugs, mudança de requisitos e confiança nos testes.
Fórmula	$TAR_k = \sum_{i=1}^{i \in T} \frac{FT_i}{T}$, sendo FP os fatores de complexidade (descritos acima) e W o peso de cada um deles.
Escala	Racional
Coleta	Engenheiro de garantia de qualidade

	Periodicidade do Evento: Ao final de cada sprint.
Procedimento	O cálculo é feito manualmente pelo responsável da coleta.
Análise	Métrica auxiliar
Providência	Ele deve ser usado, juntamente com o <i>Product Complexity</i> para o cálculo da métrica <i>Product Backlog Rating</i> .

Tabela 4 - Métrica PBR

Métrica	<i>Product Backlog Rating (PBR)</i>
Objetivo de Medição	Detecção de erros no código
Descrição	É uma nota dada ao item do Product Backlog após sua entrega.
Fórmula	$PBR = \frac{\sum_{i=1}^{i \in N} PCL_i * TAR_i}{\sum_{i=1}^{i \in N} PCL_i}$, sendo PCL advinda da métrica <i>Product Complexity Level</i> , e TAR da métrica <i>Test Assessement Rate</i> .
Escala	Racional
Coleta	Engenheiro de garantia de qualidade Periodicidade do Evento: Ao final de cada sprint.
Procedimento	O cálculo é feito manualmente pelo responsável da coleta.
Análise	O PBR possui como resultado um número de 0 a 5 que pode ser interpretado: 5 - Excelente 4 - Bom 3 - Moderado

	2 - Ruim 1 - Péssimo
Providência	Itens do product backlog com notas ruins e péssimos possuirão histórias técnicas criadas para melhoria na qualidade de seus testes.