Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Curso de Engenharia de Software

Plano de Projeto

Autores
Gabriel Viana
Indiara Duarte
Matheus Henrique
Matheus Oliveira
Mateus Andrade
Ramon Silva
Stefânia Bezerra
Thiago Félix

Brasília, DF
2017

Gabriel Viana
Indiara Duarte
Matheus Henrique
Matheus Oliveira
Mateus Andrade
Ramon Silva
Stefânia Bezerra
Thiago Felix

Plano de Projeto

Trabalho elaborado na disciplina de Verificação e Validação de *Softw*are na Universidade de Brasília.

Brasília, DF
2017

1. INTRODUÇÃO

De acordo com (SOMMERVILLE, 2011), revisões e inspeções são atividades de controle de qualidade que verificam a qualidade que o projeto entrega. Estes envolvem examinar o software, sua documentação e o processo para descobrir erros e omissões, a fim de verificar se os padrões de qualidade propostos foram seguidos.

Com o intuito de diminuir o retrabalho e aumentar a qualidade de produtos e artefatos, a abordagem de revisões e inspeções têm se mostrado bastante eficiente em encontrar defeitos ao longo do processo de desenvolvimento de um software, e impedindo que esses erros se propague durante o projeto.

Dentro da universidade, muitos produtos são desenvolvidos pensando mais no tempo em que eles serão entregues e, por muitas vezes a qualidade desses desses softwares, tanto código quanto documentação, não são levados tanto a sério. Visando esse cenário, técnicas de verificação e validação, mais especificamente revisões e inspeções, serão usadas para garantir um produto de qualidade.

2. METODOLOGIA

2.1. Proposta

Este trabalho propõe a aplicação de revisões em pares de código e inspeção no SchollApp, um projeto em desenvolvimento por alunos do curso de Engenharia de Software da Universidade de Brasília na disciplina de Desenho de Software, cujo possui o objetivo dar suporte ao aluno que deseja organizar compromisso acadêmicos de maneira mais fácil.

É esperado que com as técnicas aplicadas propostas que haja uma melhora da qualidade do código, da qualidade dos documentos inerentes ao desenvolvimento, prevenção de defeitos e falhas e identificação destas.

Será utilizado o método da pesquisa-ação de acordo com a definição de Thiollent (1998) para o desenvolvimento do projeto,

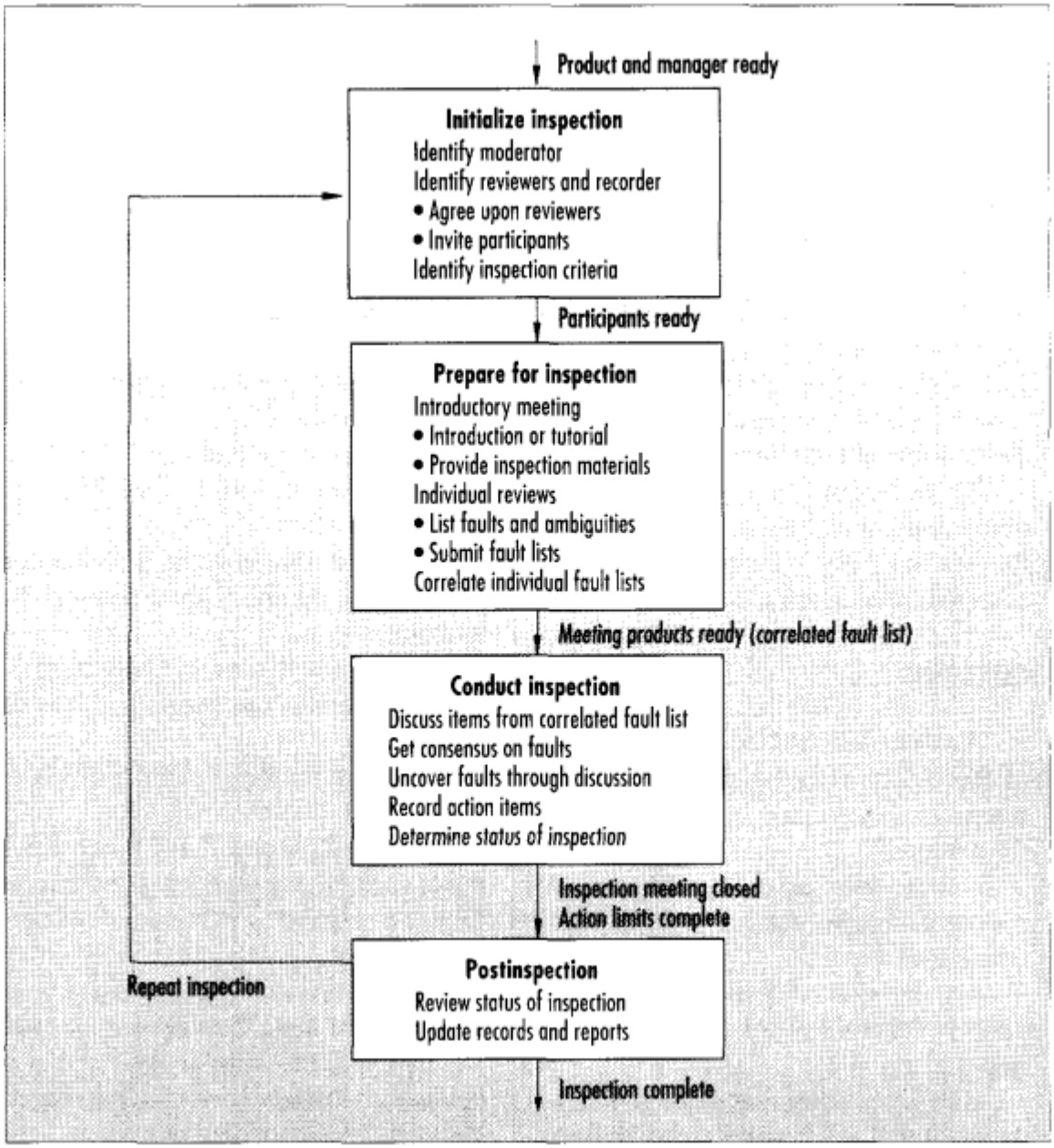
abordando a participação colaborativa de todos os envolvidos na aplicação das técnicas assim como a extração dos resultados gerados no projeto com o objetivo de solucionar o problema abordado.

> *A pesquisa ação é um tipo de investigação social com base empírica que é concebida e realizada em estreita associação com uma ação ou com a resolução de um problema coletivo no qual os pesquisadores e os participantes representativos da situação ou do problema estão envolvidos de modo cooperativo ou participativo.*
>
> (THIOLLENT, 1998)

A inspeção é utilizada para pequenas revisões de código e artefatos, é uma maneira eficiente de detectar falhas e evitar que elas se espalhem para as próximas etapas (MASHAYEKHI, et. al, 1993). As inspeções utilizaram como modelo o CSI (Inspeção Colaborativa de oftware, tradução livre), é uma ferramenta que propõe um modelo de inspeções voltadas a software.

Pelo fato do CSI se tratar de uma ferramenta antiga, a mesma não possui um software que gerencia as etapas e os artefatos gerados, e acabou servindo apenas como um modelo/metodologia a ser seguida, pois se encaixa-se bem nas necessidades da equipe em fazer inspeções por ser uma metodologia com objetivos muito bem definidos.

De acordo com MASHAYEKHI, et. al, 1993, O CSI pode ser utilizado de duas formas diferentes, uma delas consiste nos revisores levantarem falhas e preocupações informalmente, e também são encorajados a levantar aspectos positivos sobre o material alvo da revisão. A outra forma, cada revisor cria uma lista de falhas e entrega ao responsável pelo material antes de se encontrarem. Em ambos os cenários, cada integrante do time tem papeis especificos, se preparam individualmente para a inspeção e encontram falhas que resultarão em itens de ação.



Representação da inspeção proposta pelo CSI. (MASHAYEKHI, et. al, 1993)

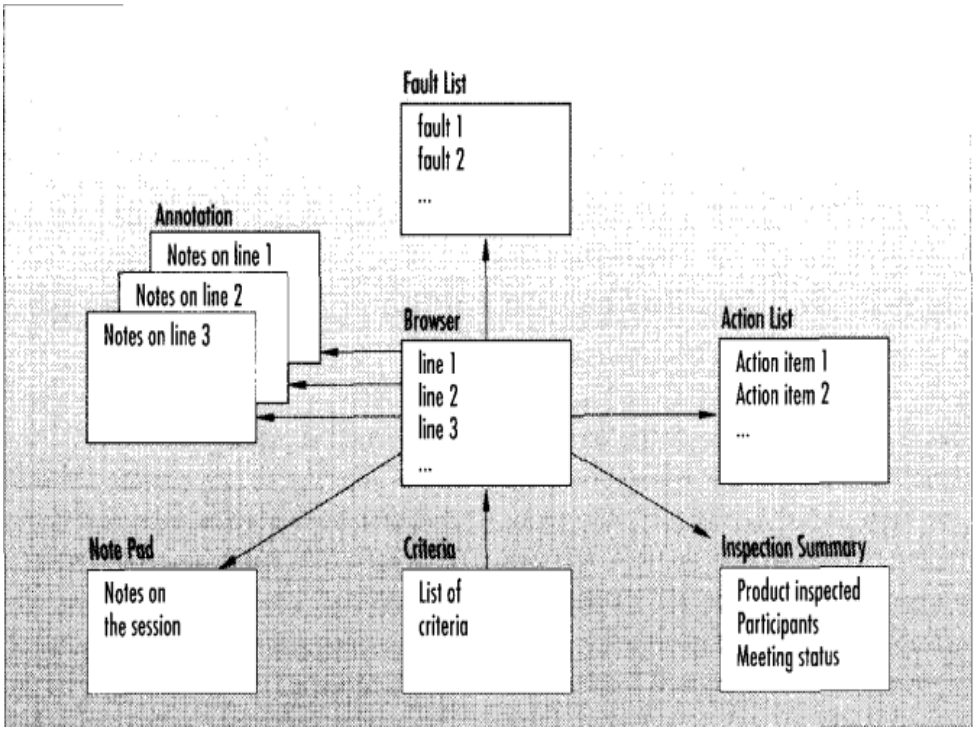
Os participantes do processo, seguidos de suas respectivas funções já adaptadas à nossa metodologia, são:

- O Produtor do conteúdo: Trata-se do desenvolvedor, que submeterá o trabalho à avaliação através de Pull Request na ferramenta GitHub.
- O Moderador: Será o Scrum Master do Time de Desenho de Software, que definirá quem será o revisor e está responsável por aprovar ou não a revisão.

- O Gravador: A própria ferramenta GitHub será encarregada de gravar os dados da revisão, o Pull Request tem como entrada o conteúdo a ser avaliado, comentários e quem são os participantes de determinada revisão.
- Os Revisores: Membros da equipe de Verificação e Validação que estarão vinculados ao repositório da disciplina de Desenho de Software e serão escolhidos pelo moderador.

As tarefas realizadas no CSI, de acordo com MASHAYEKHI, et. al, 1993, e já adaptadas às nossas necessidades, são:

- Distribuição do matérial: Dar inicio ao pull request, definir os papéis pra que cada um tenha ciência do material que será revisado.
- Revisões individuais do material: Assim que definidos os papéis, cada participante irá revisar individualmente.
- Correlacionar as falhas levantadas durante a revisão individual: Os participantes irão relacionar as falhas levantadas por cada um para uma melhor análise das falhas.
- Discutir falhas durante o encontro formal
- Gravar os itens de ação: Após um consenso, serão definidos os itens à serem corrigidos e as ações tomadas, caso sejam necessárias.
- Gravar os resultados do encontro: Gerar um backlog com tudo o que foi gerado naquela inspeção, como pontos discutidos, erros levantados, soluções a serem implementadas, etc.

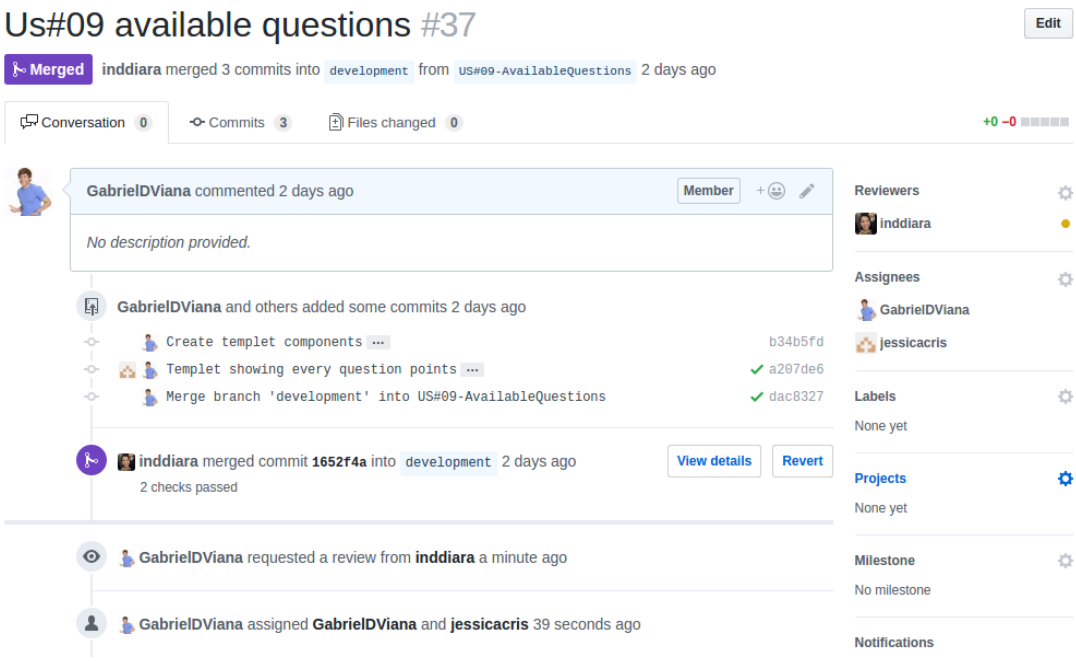


Representação das atividades e artefatos do CSI. (MASHAYEKHI, et. al, 1993)

2.2. Ferramentas

2.2.1. Github

Utilizando o Github onde se encontra o repositório da equipe de desenho e seguindo a metodologia adotada por esta, serão feitas as revisões, aproveitando o ambiente do próprio GitHub, sendo feitas através de pull request, onde o revisor pode comentar sobre o que foi feito (BELLER, BACHELLI, ZAIDMAN 2014). Como é mostrado no exemplo abaixo, onde existe um revisor e dois responsáveis pelo código.



Exemplo de Pull Request. Fonte: GitHub

2.2.2. Code Climate

A partir das métricas definidas pelo time de Verificação e Validação, identificou-se a necessidade da utilização de uma ferramenta de análise estática de código. Segundo Terra, R. e Bigonha, S. R. uso de ferramentas de análise estática fazem parte das técnicas de Verificação e Validação de Software, como sugerem:

> *Popularmente, o termo Análise Estática de Código se refere à análise automatizada, que é uma das técnicas estáticas de inspeção de software no processo de V&V. Logo, o termo verificador ou analisador estático de código é usualmente aplicado à verificação realizada por uma ferramenta automatizada seguida de uma análise humana sobre os resultados.* (Terra, R. et al, s.d)

Com isso, o Code Climate é uma ferramenta que se encaixa nas necessidades de ambos os grupos, tanto no de Verificação e Validação, quanto no de Desenho de Software, pois trata-se de uma ferramenta automatizada que analisa complexidade, cobertura de testes, duplicação de testes, etc.

2.2.3. Checklist

Será utilizado o modelo de checklist proposto por Cherchak (1996), cujo segue uma abordagem estatística formal para a melhoria dos checklists,

baseada em uma análise casual e modelagem de defeitos. Este modelo foi escolhido por ser viável ao projeto segundo o estudo desenvolvido em cima de um produto com interface, além da eficácia comprovada por um modelo estatístico de detecção de defeitos. Outro fator importante na escolha são as vantagens de adotar uma abordagem estatística segue ao fato de não necessitar de forma geral um conjunto de regras para a prática da inspeção, como é utilizada uma metodologia ágil no processo de desenvolvimento uma abordagem que menos presa a formalidade se adequa melhor ao projeto em geral levando em conta os princípios do manifesto ágil.

> Indivíduos e interações mais que processos e ferramentas
> Software em funcionamento mais que documentação abrangente
> Colaboração com o cliente mais que negociação de contratos
> Responder a mudanças mais que seguir um plano.
>
> (AGILE MANIFESTO, 2001)

Outras vantagens da abordagem escolhida é abordada por Cherchak é a abordagem da inspeção de código e não somente de outros documentos diversos, no caso é possível adotar tanto a revisão em pares de código proposta no trabalho como a inspeção utilizando as duas técnicas juntas, no caso um pull request apenas seria aprovado caso o que for submetido seguisse os itens do checklist proposto.

3. PROBLEMA

A disciplina de Desenho de Software é considerada por muitos, uma disciplina que exige esforços constantes para o sucesso do projeto final. Isso inclui uma documentação correta e concisa além de um software testável que aplique os padrões de design estudados, e que esse tenha uma qualidade aceitável para os níveis da disciplina.

Com as constantes criações de documentos e diagramas, é comum verificar que vários destes possuem alguma informação equivocada ou errada. Ausências de artefatos também podem ser percebidas em alguns casos.

Já em relação ao código, como existem várias pessoas trabalhando paralelamente, os mesmos problemas vistos na documentação podem ocorrer aqui, em forma de códigos com erros e a não implementação de funcionalidades previstas no projeto. A parte do código é um pouco mais crítica que a documentação, pois ainda devem ser verificados se os padrões estudados na disciplina estão sendo utilizados, se o código está sendo testado corretamente e se o código possui qualidade em questões como complexidade ciclomática, número de erros, nível de acoplamento e outras.

Logo, é necessário alguma forma de que todos os artefatos do projeto sejam verificados. Como dito por SOMMERVILLE, as técnicas de revisão e inspeção examinam o software em busca destas falhas e omissões, para que a qualidade seja mantida. Assim, utilizar essas técnicas no projeto SchoolApp irá implementar um controle de qualidade dos artefatos que não existia anteriormente e logo, o projeto terá mais chances de obter sucesso na disciplina.

4. OBJETIVOS GERAIS

- Definir e implementar como será feito o controle de qualidade utilizando técnicas de revisão e inspeção em relação aos artefatos do software – código fonte e documentação;
- Melhorar, por meio das técnicas que serão utilizadas, o desempenho do projeto em relação à matéria.

5. OBJETIVOS ESPECÍFICOS

- Reduzir o número de erros e omissões na documentação do projeto, o que inclui verificar se a documentação está de acordo com o esperado pela disciplina;
- Melhorar os artefatos, tanto a documentação quanto o código produzido através da aplicação das revisões técnicas e inspeções, com o intuito de garantir uma maior conformidade entre estes.
- Utilizar o GQM(*Goal Question Metric*) como uma abordagem para definir as métricas a serem coletadas dos produtos e processos de software. Uma etapa do processo do GQM é a coleta de métricas, sendo que estas serão obtidas a partir da definição de objetivos e questões. Dessa forma, as métricas coletadas irão dar suporte à análise e levantamento de indicadores de qualidade.
- Elaborar um checklist de verificação contendo os principais erros e/ou inconsistências encontrados com frequência em trabalhos acadêmicos, para que dessa forma haja uma espécie de filtro, capaz de remover as informações equivocadas mais comuns.

6. GOAL-QUESTION-METRICS

6.1. Objetivos Estratégicos

O objetivo estratégico da entidade a ser medida é de indispensável importância para a elaboração de uma abordagem seguindo o processo *Goal-Question-Metric* (GQM). São dos objetivos estratégicos que são geradas as questões a serem respondidas pelas métricas que serão coletadas posteriormente. Logo, vemos que os objetivos diretamente ligados com as métricas que serão estabelecidas para alcançar os objetivos.

No contexto em que nos encontramos – definindo um processo de medição para um produto de *software* que está sendo elaborado em uma disciplina da Universidade de Brasília – foram definidos três objetivos estratégicos para o processo de medição: todos no âmbito do produto.

6.1.1. Objetivos de Produto

O produto de *software* desenvolvido na disciplina de Desenho de Software tem uma grande exigência por uma alta qualidade de código, uma vez que as técnicas de padrão de desenho de *software* têm que ser aplicados. Além disso, a documentação deve estar de acordo com o que o grupo propôs a partir dos artefatos apresentados pela professora durante as aulas da disciplina. Assim, foram definidos os seguintes objetivos:

Código	O1
Descrição	Acompanhar a qualidade de código do produto desenvolvido pela equipe da disciplina de Desenho de *Software*.
Analisar	O código fonte
Com o propósito de	Manter um código limpo e com o mínimo de erros
Com respeito às	Técnicas de programação exigidas pela equipe
Do ponto de vista	Dos desenvolvedores
No contexto da	elaboração de um produto de *software* na disciplina de Desenho de Software

Código	O2
Descrição	Acompanhar a aplicação dos padrões de desenho de *software* no produto desenvolvido pela equipe da disciplina de Desenho de *Software*.
Analisar	O código fonte
Com o propósito de	Manter o código de acordo com o esperado
Com respeito aos	Padrões de desenho de software lecionados em sala de aula
Do ponto de vista	Dos desenvolvedores
No contexto da	elaboração de um produto de *software* na disciplina de Desenho de Software

Código	O3
Descrição	Verificar a documentação desenvolvida pela equipe da disciplina de Desenho de *Software*.
Analisar	A documentação
Com o propósito de	Garantir que a documentação esteja como planejada
Com respeito aos	Documentos que foram definidos a partir dos artefatos apresentados em sala de aula
Do ponto de vista	Dos desenvolvedores
No contexto da	elaboração de um produto de *software* na disciplina de Desenho de Software

6.1.2. *Abtraction Sheets*

O1. Acompanhar a qualidade de código do produto desenvolvido pela equipe da disciplina de Desenho de *Software*.	
Foco de Qualidade	**Fatores variantes**
Q01. O código produzido possui boa qualidade? Q02. A qualidade do código é afetada pela	- O desenvolvimento é feito em dupla, que são dinâmicas. - Refatorações para melhoria da

falta de conhecimento da tecnologia utilizada?	qualidade do código - Entendimento acerca da metodologia
Hipótese de linha-base	**Impacto na hipótese de linha-base**
- O código atual está aceitável.	- Espera-se um maior esforço dos desenvolvedores para ficarem atentos à qualidade do código. - No decorrer do desenvolvimento é esperado que os integrantes tenham maior conhecimento em relação à tecnologia, facilitando o desenvolvimento de um bom código.

O2. Acompanhar a aplicação dos padrões de desenho de software no produto desenvolvido pela equipe da disciplina de Desenho de Software.	
Foco de Qualidade	**Fatores variantes**
Q01. Os padrões apresentados em sala de aula estão sendo aplicados no projeto? Q02. Os padrões estão sendo aplicados de maneira correta?	- Entendimento acerca de padrões de desenho de *software* - Refatorações para a aplicação dos padrões no projeto de *software*
Hipótese de linha-base	**Impacto na hipótese de linha-base**
- Apenas alguns padrões GRASP estão sendo aplicados.	- Espera-se um maior esforço dos desenvolvedores para que ocorra aplicação dos padrões de desenho de software. - No decorrer do desenvolvimento é esperado que os integrantes tenham maior conhecimento em relação aos padrões que serão aplicados no projeto.

O3. Verificar a documentação desenvolvida pela equipe da disciplina de Desenho de *Software*.	
Foco de Qualidade	**Fatores variantes**
Q01. Os documentos pré determinados já foram ou estão sendo desenvolvidos? Q02. Os documentos estão de acordo com o produto em desenvolvimento?	- Documentos estão em plena maturação durante o desenvolvimento do *software*

Hipótese de linha-base	Impacto na hipótese de linha-base
- A documentação está aceitável até que seja feita a primeira inspeção.	- Os desenvolvedores dedicar-se-ão mais para que os documentos estejam atualizados de acordo com o código fonte.

6.2. Questões

O1Q01. O código produzido possui boa qualidade?
O1Q02. A qualidade do código é afetada pela falta de conhecimento da tecnologia utilizada?
O2Q01. Os padrões apresentados em sala de aula estão sendo aplicados no projeto?
O2Q02. Os padrões estão sendo aplicados de maneira correta?
O3Q01. Os documentos pré determinados já foram ou estão sendo desenvolvidos?
O3Q02. Os documentos estão de acordo com o produto em desenvolvimento?

6.3. Métricas

Código	M01
Questões a serem respondidas	O1Q01
Objetivo	Verificar a complexidade lógica dos métodos desenvolvidos
Tipo	Indireta
Derivada do	- Número de decisões que um determinado bloco de código deve fazer.
Como será mensurada	O código será analisado estaticamente pelo Codacy
Forma de Coleta	O Codacy fará a análise de complexidade quando o desenvolvedor efetua um pull request no repositório de desenvolvimento
Escala de medição	Ordinal
Indicador	Indicador de declarações máximas e de complexidade ciclomática. O de declarações máximas será um número inteiro, indicando o número de declarações do método. O indicador de complexidade ciclomática também será um inteiro, que indica caminhos linearmente independentes dentro de um método.
Análise	- **Valor aceitável** : Métodos com até 30 declarações e até 6 caminhos linearmente independentes. - Medida a ser tomada: Manter as práticas que foram

	utilizadas até o momento para a elaboração dos métodos. - **Valor preocupante** : Métodos com mais de 30 declarações ou mais de 6 caminhos linearmente independentes. - Medida a ser tomada: Repensar a lógica que está sendo executada no código para que um valor aceitado seja alcançado.

Código	M02
Questões a serem respondidas	O1Q01
Objetivo	Verificar a quantidade de linhas de código que um determinado método possui
Tipo	Direta
Derivada do	- Quantidade de linhas que um método possui
Como será mensurada	O código será analisado estaticamente pelao Codacy
Forma de Coleta	O Codacy fará a análise dos métodos quando o desenvolvedor efetua um pull request no repositório de desenvolvimento
Escala de medição	Ordinal
Indicador	Indicador de declarações máximas e de complexidade ciclomática. O de declarações máximas será um número inteiro, indicando o número de declarações do método. O indicador de complexidade ciclomática também será um inteiro, que indica caminhos linearmente independentes dentro de um método.

Análise	- **Valor aceitável** : Métodos com até 30 declarações e até 6 caminhos linearmente independentes.
	  - Medida a ser tomada: Manter as práticas que foram utilizadas até o momento para a elaboração dos métodos.
	- **Valor preocupante** : Métodos com mais de 30 declarações ou mais de 6 caminhos linearmente independentes.
	  - Medida a ser tomada: Repensar a lógica que está sendo executada no código para que um valor aceitado seja alcançado.

Código	M03
Questões a serem respondidas	O1Q02
Objetivo	Verificar o impacto do conhecimento sobre a tecnologia no desenvolvimento das *sprints*.
Tipo	Indireta
Derivada do	- Quadro de conhecimento da equipe - Velocity
Como será mensurada	Serão coletados o velocity e o quadro de conhecimento da equipe de desenvolvimento. Os membros terão de 1 a 5 sobre cada tecnologia e a soma total de conhecimento.
Forma de Coleta	O velocity será analisado e, então, comparado com o conhecimento atual da equipe. As coletas serão feitas por sprint.
Escala de medição	Racional
Indicador	IDK = Velocity / Média do nível de conhecimento dos membros. Onde o IDK será um valor racional.

Análise	- **Valor aceitável** : Espera-se que o IDK suba ou mantenha-se constante durante todo o desenvolvimento do projeto, uma vez que esperamos que o velocity aumente juntamente com a média do nível de conhecimento dos membros.
	  - Medida a ser tomada: Manter as práticas que foram utilizadas até o momento.
	- **Valor preocupante** : Caso o IDK atual seja menor que o IDK da sprint anterior.
	  - Medida a ser tomada: Reavaliar o desempenho do time de desenvolvimento.

Código	M04
Questões a serem respondidas	O2Q01, O2Q02
Objetivo	Verificar o uso dos padrões de desenho de software apresentados em sala.
Tipo	Direta
Derivada do	- Código gerado
Como será mensurada	Será analisado os padrões que estão sendo utilizados no software.

Forma de Coleta	Manual.
Escala de medição	Racional
Indicador	PTT = Padrões usados corretamente / Padrões adotados.

Análise	- **Valor aceitável** : 100%, pois é esperado que todos os padrões sejam aplicados corretamente.
	  - Medida a ser tomada: Manter as práticas que foram utilizadas até o momento.
	- **Valor preocupante** : Valores abaixo de 100%.
	  - Medida a ser tomada: Corrigir os padrões que não estão sendo aplicados corretamente.

Código	M065
Questões a serem respondidas	O3Q01
Objetivo	Verificar se os documentos estão sendo ou foram elaborados.
Tipo	Direta
Derivada dos	- Documentos escritos - Documentos previstos para serem desenvolvidos
Como será mensurada	Será feita a verificação dos documentos em desenvolvimento e já desenvolvidos através de *checklist*.
Forma de Coleta	Manual.
Escala de medição	Ordinal
Indicador	DC = Número de documentos esperados não desenvolvidos. Onde DC é um número inteiro.

Análise	- **Valor aceitável** : Espera-se que o valor de DC seja igual a 0.
	  - Medida a ser tomada: Manter as práticas que foram utilizadas até o momento.
	- **Valor preocupante** : Qualquer número maior que 0.
	  - Medida a ser tomada: Analisar os documentos que ainda têm que ser desenvolvido.

Código	M06

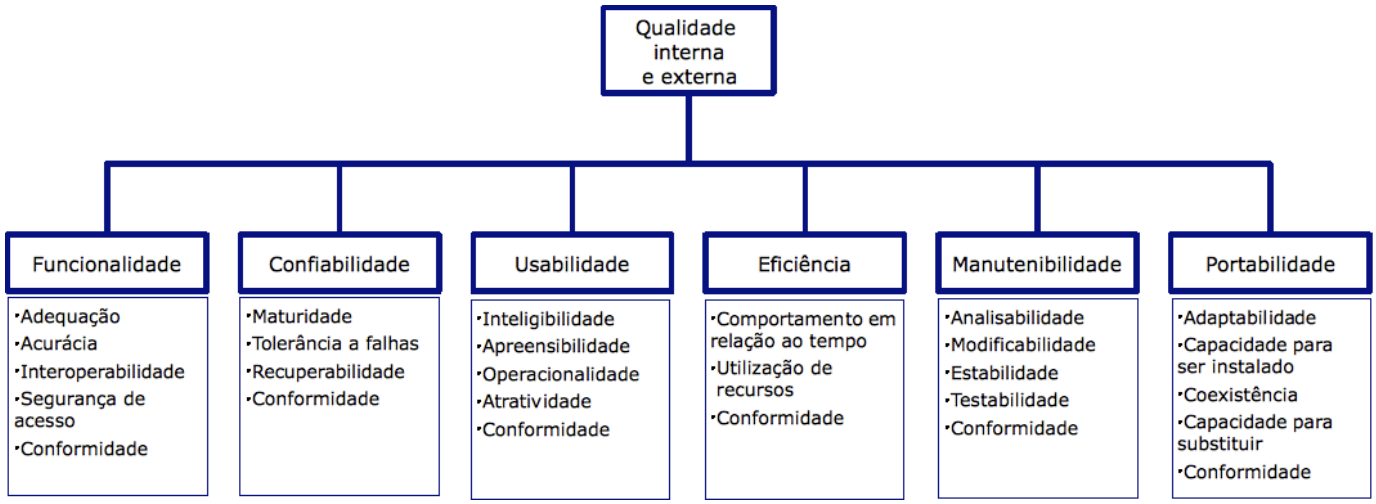
Questões a serem respondidas	O3Q02
Objetivo	Verificar se os documentos desenvolvidos estão de acordo com o código que está sendo gerado.
Tipo	Direta
Derivada dos	- Documentos escritos - Código gerado
Como será mensurada	Os possíveis erros serão encontrados através do checklist que a equipe praticará.
Forma de Coleta	Manual.
Escala de medição	Ordinal
Indicador	DC = Número de documentos desatualizados. Onde DC é um número inteiro.

Análise	- **Valor aceitável** : Espera-se que o valor de DC seja igual a 0.
	  - Medida a ser tomada: Manter as práticas que foram utilizadas até o momento.
	- **Valor preocupante** : Qualquer número maior que 0.
	  - Medida a ser tomada: Analisar as distinções entre o código e os documentos e corrigi-los.

7. RESULTADOS ESPERADOS

Os resultados terão base no padrão de qualidade ISO/IEC 2500 (SQuaRE), na antiga ISO/IEC 9126. A norma ISO/IEC 9126 tem foco na qualidade do produto de software, ele propõe atributos de qualidade que pode ser visto na imagem abaixo:

Figura 1 - Atributos de qualidade.



Pode-se notar que possui uma subcategoria com o nome Conformidade em cada atributo, o que é usada para avaliar o quanto o software obedece os requisitos de qualidade conforme a norma.

Para isso, será feito:

- Verificação de conformidade;
- Verificação em tempo real (usando o peer review) e através da utilização do pull request (feita pelo scrum master) se está cumprindo com o que foi definido como qualidade.

Seguindo essas etapas, será possível perceber se os resultados esperados serão contemplados conforme o planejado.

REFERÊNCIAS BIBLIOGRÁFICAS

- Ian Sommerville. **Engenharia de Software, 9ª Edição. Pearson Education, 2011.**
- Silvana M. Melo, **Inspeção de software.**
- Moritz Beller, Alberto Bacchelli, Andy Zaidman. **Modern Code Reviews in Open-Source Projects: Which Problems Do They Fix?**
- F. Macdonald, J. Miller, A. Brooks, M. Roper and M. Wood. **A Review of Tool Support for Software Inspection.**
- MASHAYEKHI V., DRAKE J.M., TSAl W., RIEDL J. **Distributed, collaborative software inspection.** University of Minnesota at Minneapolis.
- CHERNAK Y. **A statistical approach to the inspection checklist formal synthesis and improvement.** IEEE society computer.

- **Manifesto para desenvolvimento ágil de software.** <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em 17/04/2017 às 20:13
- THIOLLENT, M. **Metodologia da pesquisa-ação.** São Paulo: Cortez & Autores Associados, 1988.
- Terra, R. e Bigonha, R. S., Ferramentas para Análise Estática de Códigos Java. <https://www.academia.edu/4071023/Ferramentas_para_Analise_Estatica_de_Codigos_Java>. Acesso em 27 de abril de 2017.