Universidade de Brasília Faculdade UnB Gama

Universidade de Brasília - FGA

Disciplina: Verificação e Validação de Software

Professor: Ricardo Ajax

Grupo 6

Integrantes/Matrícula: **Dandara Aranha - 11/0010256**

Isaac Borges - 11/0121996

Leonardo Arthur - 14/0025171

Matheus Figueiredo - 14/0047743

Matheus Mello - 11/0017692

Victor Hugo Lopes - 13/0136581

Proposta de um processo de verificação e validação de software para os projetos da disciplina de GPP da faculdade do Gama que estejam utilizando metodologias ágeis de desenvolvimento de software

1. Problema

A falta de definição de um processo de verificação e validação de software na disciplina de GPP é um problema recorrente ao longo dos semestres. A disciplina de GPP pede que sejam retiradas métricas de qualidade do software produzido em conjunto com MDS. Contudo ainda não foram verificadas quais técnicas de verificação e validação realmente impactam na qualidade do software e quais métricas trazem a certeza que os incrementos de software que estão sendo criados atendem as necessidades do cliente.

Atualmente cada grupo da disciplina decide quais atividades de verificação e validação vão realizar, como e quando vão desenvolver tais atividades. Muitos alunos da disciplina por falta de experiência e orientação acabam por desenvolver um trabalho onde o processo de verificação e validação de software não existe, ou se existe, não é aplicado da maneira correta fazendo com que essas atividades não reflitam na qualidade do produto desenvolvido.

- Falta de um processo de verificação, validação e testes definido na disciplina de GPP que reflita na qualidade do produto que está sendo desenvolvido;
- Métricas que não indicam se a qualidade do software e artefatos está satisfatória aos objetivos da disciplina.

2. Objetivos

- Modelar e propor um processo para otimizar a verificação e validação no contexto da disciplina de GPP no contexto ágil
- Validar o processo junto aos alunos da disciplina
- Contribuir para a melhoria da qualidade dos artefatos produzidos na disciplina

3. Questões de Pesquisa

(Q1) Existem processos de verificação e validação definidos para metodologias ágeis de desenvolvimento de software na literatura?

(Q2) Quais as técnicas mais utilizadas para a verificação, validação e teste de projeto de software ágil?

4. String de Busca

("Agile software development" **OR** "Agile methodologies") **AND** (((Verification **AND** Validation **AND** Tests) OR ("V&V")) **OR** (Quality Assurance) **OR** (Quality Control)) **AND** (Process Structure)

5. Fontes de Pesquisa

As buscas serão feitas em bases de dados digitais. Foi selecionado uma base que já faz a indexação de outras, sendo elas:

- **Scopus (**http://www.scopus.com/**)**

6. Processo de Seleção dos Estudos

A estratégia de pesquisa definida é:

1. Pesquisa de trabalhos nas fontes definidas utilizando as strings de busca.
2. Leitura do título, resumo e palavras chaves dos trabalhos aplicando os critérios de

inclusão e exclusão definidos neste protocolo. Com base nesses critérios os trabalhos serão pré-selecionados.

3. Os trabalhos que forem selecionados no passo anterior deverão ser lidos por completo, e por fim as informações presentes no formulário de coleta de dados que será desenvolvido deverão ser extraídas destes trabalhos.

6.1 Critérios de Inclusão

1. Os trabalhos devem estar disponíveis em bases de dados digitais previamente definidas, de preferência de forma gratuita.
2. Serão considerados artigos publicados a partir do ano de 2001. Entretanto pode-se encontrar fontes clássicas com definições (livros com conceitos clássicos ou artigos pioneiros) que também serão considerados.
3. O artigo deve possuir menção a técnicas/atividades de verificação, validação e testes de software em projetos de desenvolvimento ágil

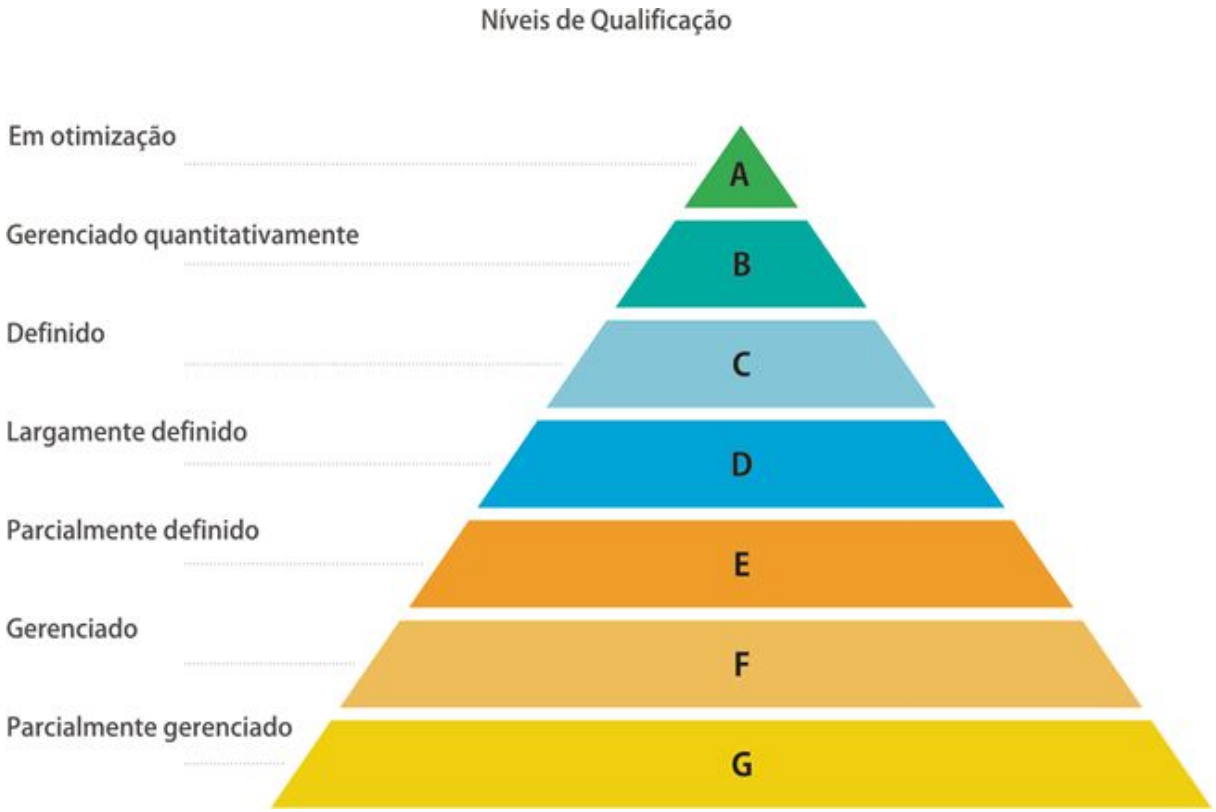
6.2 Critérios de Exclusão

1. Trabalhos que, forem identificados como fora do escopo e tema do trabalho, que não tratem de VeriVal no contexto ágil.
2. Trabalhos publicados antes do ano de 2001 que não sejam de fontes clássicas e renomadas em relação ao tema do trabalho.
3. Trabalhos não disponíveis em bases de dados digitais e/ou que não estejam disponibilizados de forma gratuita.
4. Trabalhos duplicados, que foram publicados em mais de uma base de dados .

7. Proposta de Solução

O Processo modelado e definido para contextos ágeis deverá conter as seguintes atvidades:

- Atividades propostas pelo MPS-Br nível D incluídas;



7.1 Atividades que devem ser atendidas para Validação (MPS-Br nível D**)**

VAL1 - Produtos de trabalho a serem validados são identificados;

VAL2 - Uma estratégia de validação é desenvolvida e implementada,estabelecendo cronograma, participantes envolvidos, métodos para validação e qualquer material a ser utilizado na validação;

VAL3 - Critérios e procedimentos para validação dos produtos de trabalho a serem validados são identificados e um ambiente para validação é estabelecido;

VAL4 - Atividades de validação são executadas para garantir que o produto esteja pronto para uso no ambiente operacional pretendido;

VAL5 - Problemas são identificados e registrados;

VAL6 - Resultados de atividades de validação são analisados e disponibilizados para as partes interessadas;

VAL7 - Evidências de que os produtos de software desenvolvidos estão prontos para o uso pretendido são fornecidas.

7.2 Atividades que devem ser atendidas Para Verificação (MPS-Br nível D**)**

VER1 - Produtos de trabalho a serem verificados são identificados;
VER2 - Uma estratégia de verificação é desenvolvida e implementada, estabelecendo cronograma, revisores envolvidos, métodos para verificação e qualquer material a ser utilizado na verificação;
VER3 - Critérios e procedimentos para verificação dos produtos de trabalho a serem verificados são identificados e um ambiente para verificação é estabelecido;
VER4 - Atividades de verificação, incluindo testes e revisões por pares, são executadas;
VER5 - Defeitos são identificados e registrados;
VER6 - Resultados de atividades de verificação são analisados e disponibilizados para as partes interessadas.

8. Aplicação: Estudo de Caso

1) 1 equipe de MDS e GPP;
2) 2 Sprints sem aplicação de Técnicas X 2 Sprints com aplicação de Técnicas de Verificação e Validação conforme processo proposto;
3) Aplicação no Código Fonte e nos Testes;
4) Colher Métricas ao fim de cada sprint;
5) Análise dos resultados juntamente com os artefatos produzidos;

9 . Execução da Pesquisa e Resposta das Questões

A pesquisa na base de dados definida foi realizada utilizando a String de busca inicial, que resultou em 10 trabalhos. Como o número de trabalhos retornados inicial foi pouco optou-se por testar outras strings, e estas estão descritas abaixo:

String 1: (“Agile software development” **OR** “Agile methodologies”) **AND** (((Verification **AND** Validation **AND** Tests) OR (“V&V”)) **OR** (Quality Assurance) **OR** (Quality Control)) **AND** (Process Structure)

Resultados da String 1: 10

String 2: (("Agile software development" **OR** "Agile methodologies") **AND** (((verification **AND** validation) **OR** ("V&V")) **AND** ("Quality Assurance") **OR** ("Quality Control")))
Resultados da String 2: 2

String 3: (("Agile software development" **OR** "Agile methodologies") **AND** ((verification AND validation) **OR** ("V&V")))
Resultados da String 3: 14

String 4: (("Agile software development") **AND** ((verification **OR** validation **OR** ("V&V")) **AND** ("Software Process Improvement") **OR** ("Process Model Reference")))

Resultados da String 3: 6

O título, resumo e palavras chaves dos 14 trabalhos foram lidos e os critérios de aceitação e exclusão analisados. Após leitura, 4 artigos foram classificados e estão descritos na tabela 1:

ID Trabalho	Título	Autor	Ano
1	Using a Validation Model to Measure the Agility of Software Development in a Large Software Development Organization	Mikio, I., Masayuki O., Takahiro T., Michiko O., Sanshiro S.	***2009***
2	A Framework For Software Quality Assurance Using Agile Methodology	Maria Sagheer, Tehreem Zafar, Mehreen Sirshar	***2015***

3	Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process	Tarhan, A., Yilmaz, S.G.	***2014***
4	Verification, Validation and Agile Software Testing	Gilberto Gampert	***2011***

(Q1) Existem processos de verificação e validação definidos para metodologias ágeis de desenvolvimento de software na literatura?

Não foi encontrado na pesquisa realizada neste trabalho nenhum artigo que propusesse um processo de Verificação e Validação de Software para metodologias ágeis. Entretanto, alguns autores descrevem as atividades de verificação e validação realizadas em seus trabalhos e sua importância na qualidade final do Produto. Alguns trabalhos também realizavam o levantamento do processo de desenvolvimento atual apontando em quais etapas a instituição realizava alguma prática dos processos de verificação e validação, mas não propõem nunhum modelo de processo a ser seguido.

(Q2) Quais as técnicas mais utilizadas para a verificação, validação e teste de projeto de software ágil?

A maior parte parte dos trabalhos descrevem apenas técnicas de testes de software no contexto ágil.

Segundo o autores do trabalho 1, em ágil o software é desenvolvido em incrementos que são fáceis de testar. Para testar as técnicas como teste de unidade, testes de aceitação, refatoração e programação em pares são usados. As ferramentas são usadas para análise do sistema, análise do código, teste do software e execução. Em ágil após cada iteração, o software é testado usando algumas ferramentas específicas para alcançar alta qualidade.

O trabalho 4 apresenta apresenta uma breve introdução ao tema Verificação, Validação e Teste de Software Ágil e traz um estudo de caso onde pode-se verificar que a utilização de técnicas de teste de software, tais como o TDD, podem diminuir drasticamente a ocorrência de erros no software. E segundo o autor os testes iniciam por um pequeno componente ou grupo de componentes, que são testados para verificar erros nos dados e na lógica. Após esta etapa, os componentes são integrados e neste ponto testes são executados para descobrir erros ao atender os requisitos do cliente. Conforme os erros são descobertos, devem ser diagnosticados e corrigidos (depuração). Um modelo de teste de software deve ter as seguintes características básicas:

- Proceder revisões técnicas eficazes;
- Iniciar no nível de componente e evoluir para a integração do sistema;
- Ter uma técnica de teste definida. Diferentes técnicas de teste são apropriadas a diferentes abordagens da engenharia de software;
- O teste é executado pelo desenvolvedor e (em grandes projetos) por uma equipe de testes;
- A depuração é uma atividade distinta do teste, porém deve estar associada com a estratégia de testes.

10. Processo Atual da Equipe de MDS/GPP

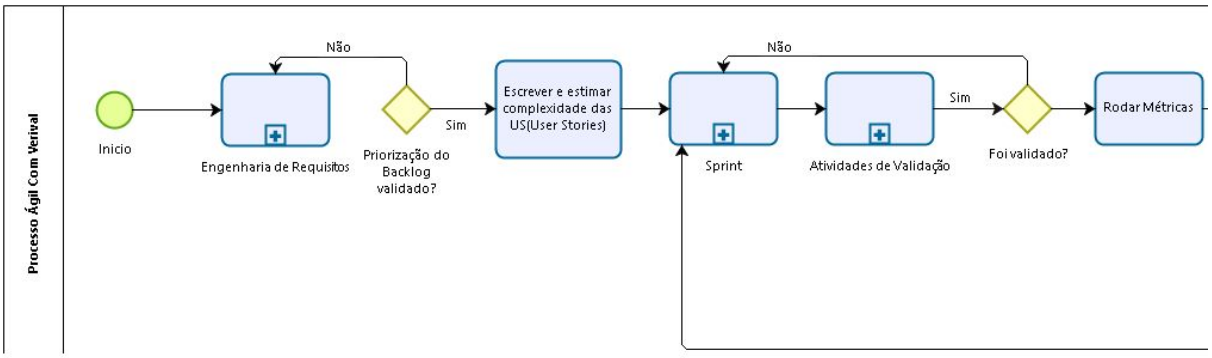
Durantes as sprints 5 e 6 em que a equipe de GPP/MDS não aplicou o processo novo o processo era o seguinte:

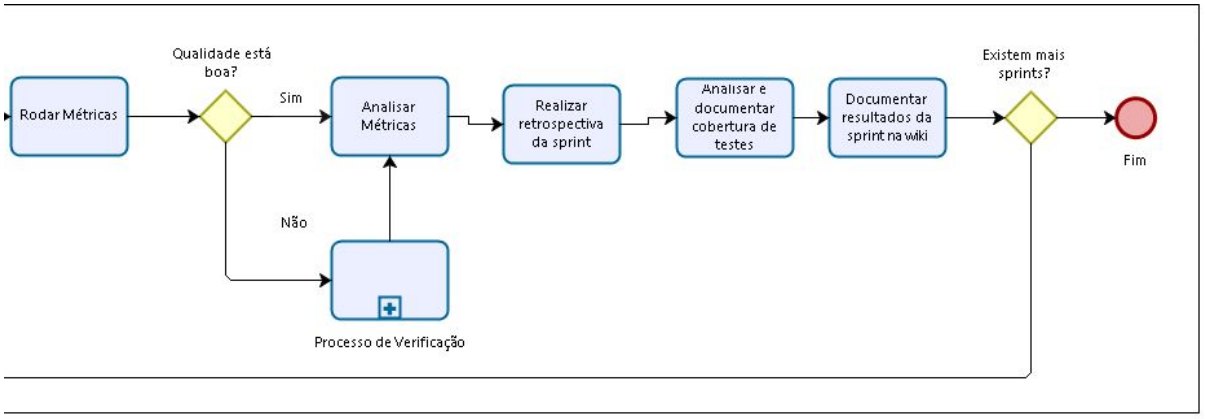
- Ocorreu uma reunião com o cliente para entender seu ambiente real, contexto que está inserido e as suas necessidades.
- Foi criado o backlog do produto para poder ter visão do que será necessário desenvolver para satisfazer o cliente.
- O backlog foi priorizado em ordem do que agrega mais valor para o cliente junto com o PO, porém sem a validação com o cliente.
- As histórias de usuários foram escritas, detalhadas e a sua complexidade foi estimada através do planning poker com a equipe inteira de gerência e desenvolvimento.
- Execução da sprint
    - Planejamento da sprint: As duplas eram definidas e as histórias de usuário eram alocadas para as equipes.
    - Escrever critérios de aceitação: O PO escreve os critérios de aceitação para cada história sem a validação com o cliente.
    - Codificar USs: As equipes desenvolvem em pares as histórias da semana.
    - Realizar testes unitários: Testes unitários para as histórias eram feitos com a ferramenta escolhida e era verificado se todos os testes estavam passando para não quebrar a build no travis.
    - Reunião diária: Durante as sprint todos os dias occoriam reuniões de 15

minutos para ter um status do que o grupo estava fazendo e quais problemas eles estavam passando.
    - Teste de integração: Eram criados cenários e passos simulando o acesso à ferramenta e passando por todos os caminhos imagináveis para ver se a solução parava de funcionar em algum ponto.
    - Realizar reunião de revisão de sprint: A equipe se reunia aos sábados para ver como o software estava funcionando com todas as funcionalidades integradas.
- Rodar métricas: O tracker rodava as métricas das ferramentas automatizadas para depois poder analisar os resultados armazenados.
- Analisar métricas: Com os arquivos das ferramentas armazenados, acontecia uma análise dos número obtidos para saber quais decisões deveriam ser tomadas para deixar o software sempre dentro da qualidade máxima esperada e que fora definida nas baselines do plano de qualidade.
- Realizar retrospectiva da sprint: O grupo se reunia para conversar a respeito do que foi bom, ruim e o que precisa melhorar na sprint seguinte, as opiniões eram anotadas para que pudessem ser resolvidas e realmente melhoradas na semana seguinte.
- Analisar e documentar cobertura de testes: A respeito da cobertura de testes coletada, o tracker realizava uma análise identificando qual o nível de satisfação e em caso de estar não satisfatório o que seria feito e como seria feito para melhorar.
- Documentar resultados da sprint na Wiki: Todos os resultados analisados e coletados eram documentados na wiki para dar visibilidade a todos da equipe e stakeholders.

11.Proposta de Processo

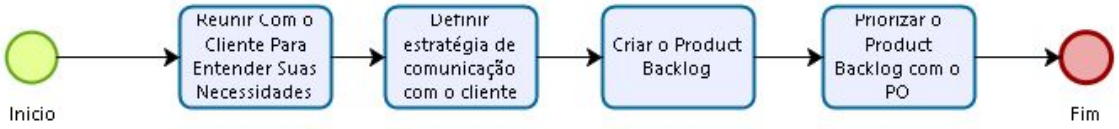
a. Processo Geral





b. Descrição das Atividades :

Subprocesso de Engenharia de Requisitos



Powered by bizagi Modeler

1. **Engenharia de Requisitos**

   1.1. **Reunir Com o Cliente Para Entender Suas Necessidades**

Sendo a primeira atividade do processo de engenharia de requisitos, tem como objetivo realizar um encontro entre a equipe de gerência e o cliente afim de serem entendidos as necessidades do cliente. Com esta atividade é levantado os requisitos iniciais para o software e a equipe de gerência possui o dever de avaliar se o projeto é viável ou não.

1.2. Definir Estratégia de Comunicação Com o Cliente

A equipe de gerência para atingir este objetivo precisa entender a rotina do cliente, os horários livres do mesmo e com isso será possível atingir da melhor forma possível as reuniões presenciais. Caso não seja possível realizar reuniões presenciais por motivo impossibilidade por parte do cliente ou da equipe de gerência, terão que ser realizadas reuniões-não presenciais, utilizando ferramentas que possibilitem uma videoconferência, como skype e hangout.

1.3. Criar o *Product Backlog*

Os gerentes juntamente com a equipe de desenvolvimento dividem as macros atividades que terão de realizar em atividades atômicas para que sejam independentes e tenham capacidade de ser desenvolvida.

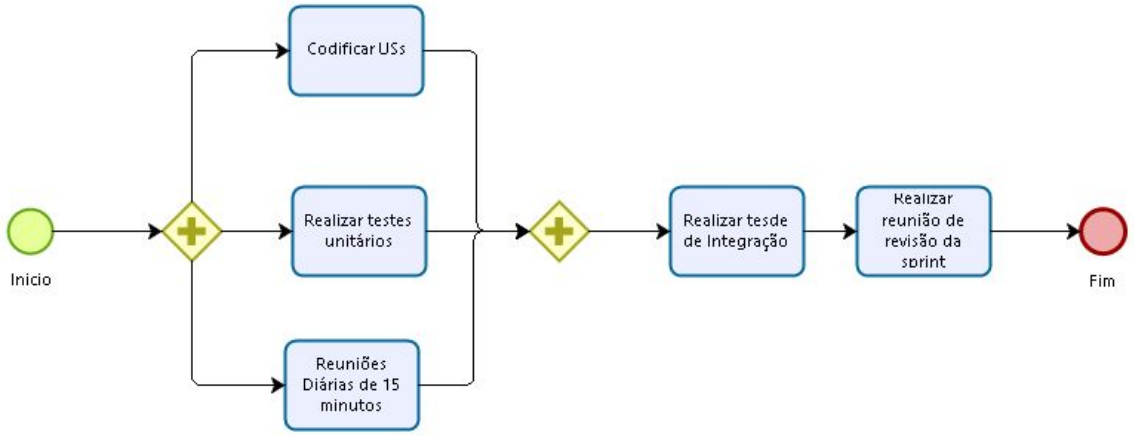
1.4. Priorizar o *Product Backlog* Com o *PO*

Após ser criado o artefato de ***Product Backlog***, a equipe de gerência possui o dever de se reunir com o ***PO*** para priorizar as histórias que mais agregam valores para o cliente naquela ***Sprint***. Caso as atividades estejam corretas de acordo como gosto do cliente e sejam validadas o próximo passo é a atividade 2, caso contrário ocorrerá uma reunião com o cliente para entender o contexto do negócio e as atividades se repetirão até que haja a validação do cliente.

2. Escrever e Estimar Complexidade Das *US(User Stories)*

Com o ***product backlog*** detalhado, as histórias de usuário são detalhadas para que os desenvolvedores compreendam o que está sendo pedido para ser desenvolvido. Nesta atividade também são estimadas a complexidade e o valor de cada história através do ***poker planning***.

Subprocesso da Sprint



3. Sprint

3.1. Planejamento da *Sprint*

No planejamento da ***Sprint*** são escolhidas as histórias de usuário mais importantes para o cliente sem esquecer do ***velocity*** para não ultrapassar a quantidade de pontos que um time pode entregar. As duplas são definidas de acordo com o nível de conhecimento visando sempre equilibrar o conhecimento dos membros e as histórias são designadas.

3.2. Planejar Atividades de Verificação

As atividades de verificação serão planejadas pela equipe. Uma das atividades que ocorrerá é a revisão em pares, onde as duplas estarão pareando e uma de cada vez estará escrevendo código e a outra estará avisando de possíveis no código. A equipe sempre estará procurando por inconsistências de requisitos nas histórias de usuário, para garantir que eles sejam corrigidos assim que encontrados e evitar aumentar o custo da correção. Após ter as abordagens estáticas definidas, as abordagens dinâmicas a serem usadas serão planejadas, como por exemplo definir quais ferramentas seriam utilizadas para rodar testes unitários e os testes de aceitação automatizados.

3.3. Planejar Atividades de Validação

A equipe definirá como e quando serão realizados as atividades para validar com o cliente. Será definido o ambiente para que ocorra a preparação da validação e suas atividades seguintes.

3.4. Validar *USs* da Sprint Junto ao *PO*

O PO poderá aprovar ou reprovar as US, em caso de aprovação ele escreverá os critérios de aceitação, em caso de reprovação, novas histórias deverão ser escritas para que estejam condizentes com o atual projeto a ser desenvolvido.

3.5. Escrever Critérios de Aceitação

Com as histórias já priorizadas para a ***Sprint***, o ***PO*** possui o dever de escrever os critérios de aceitação das histórias da ***Sprint***. Os critérios de aceitação possui o dever de auxiliar os desenvolvedores com a parte da implementação das histórias, pois os critérios de aceitação tem o dever de mostrar exatamente o que o cliente quer e espera ao final da ***Sprint***.

3.6. Codificar *USs*

Esta atividade se resume apenas em realizar a implementação das histórias planejadas na ***Sprint.***

3.7. Realizar Testes Unitários

Uma história entregue no processo ágil é uma história testada, então os desenvolvedores devem realizar testes unitários sobre a funcionalidade que foram

entregue a eles na ***Sprint***. Os testes unitários servem também para agregar qualidade ao código e a história/funcionalidade.

3.8. Reuniões Diárias de 15 Minutos

A equipe de gerência juntamente com a equipe de desenvolvimento realizam reuniões diárias com o objetivo de todos estarem por dentro do que está acontecendo com os pares de desenvolvimento. Estas reuniões possuem o timebox de 15 minutos, e todos os membros devem falar o que fez ontem, hoje, o que vai fazer amanhã e quais são seus impedimentos na sua história.

3.9. Realizar Teste de Integração

Ao final de cada ***Sprint***, os desenvolvedores devem realizar os testes de integração para verificar se todos os módulos do sistema estão funcionando corretamente. Com a realização desta atividade, será validado que os módulos estão funcionando perfeitamente ou não, e que o sistema está fazendo o que realmente deve fazer.

3.10. Realizar Reunião de Revisão de *Sprint*

Esta atividade se resume em realizar uma reunião com a equipe de gerência e desenvolvimento para que veja como está o funcionamento do software e verifique se os incrementos de software entregues naquela ***Sprint*** estão condizentes com os critérios de aceitação e os artefatos anteriores**.**

Subprocesso de Atividades de Validação

4. Atividades de Validação

4.1. Estabelecer Ambiente de Validação

Após o término da ***Sprint*** o incremento de software será apresentado ao cliente para que o mesmo possa avaliar e validar, ou não. Para isso é necessário combinar um local de apresentação para que todos os componentes necessários estejam prontos para funcionar, simulando um ambiente real de uso. Então é necessário revisar o escopo do projeto e os equipamentos necessários para fazer a preparação do ambiente.

4.2. Validar Incremento de *Software* da *Sprint* Junto ao Cliente

Esta atividade tem o objetivo de validar os incrementos de ***softwares*** propostos na ***Sprint*** pelo PO representando o cliente. O cliente irá verificar se o que foi escrito pelo PO está condizente com o que ele espera e se está de acordo com o que ele espera, se o software está funcionando até aquela entrega da maneira que foi proposta.

4.3. Registrar Problemas de Validação

Caso o cliente não esteja de acordo com o que foi escrito pelo ***PO***, as diferenças de opiniões devem ser registradas e revisadas pela equipe de gerência no próximo planejamento de ***Sprint.*** Os problemas registrados pode ser desacordos de opiniões entre cliente e o ***PO***, e até o que ficou de acordo com os dois, contudo no produto de ***software*** está errado.

5. Rodar Métricas

As ferramentas automatizadas deverão estar instaladas para que possam rodar as métricas. Os resultados deverão ser armazenados para que possam ser avaliados e analisados.

6. Analisar Métricas

Após serem realizadas anteriores (rodar métricas e colher métricas), está atividade tem como por objetivo verificar todas as métricas retiradas das ferramentas e de análise fora das ferramentas, como amostras. O objetivo de analisar estas métricas é o observar os indicadores daquela ***Sprint*** e a partir dessa análise se tomar uma ação. Ex: Caso a complexidade ciclomática estiver muito alta, a equipe de gerência deve mostrar o porque isso não é bom e a equipe de desenvolvimento deve tentar diminuir na próxima ***Sprint.***

7. Realizar Retrospectiva da Sprint

Esta atividade é a de verificar tudo o que foi bom e ruim na realização da ***Sprint***. Então é visto o que deu certo, o que foi ruim e o que deve ser melhorado na próxima ***Sprint.*** Os três itens anteriormente citados, devem ser documentados para que o próximo ***Scrum Master*** possa analisar e tentar melhorar na ***Sprint*** dele***.***

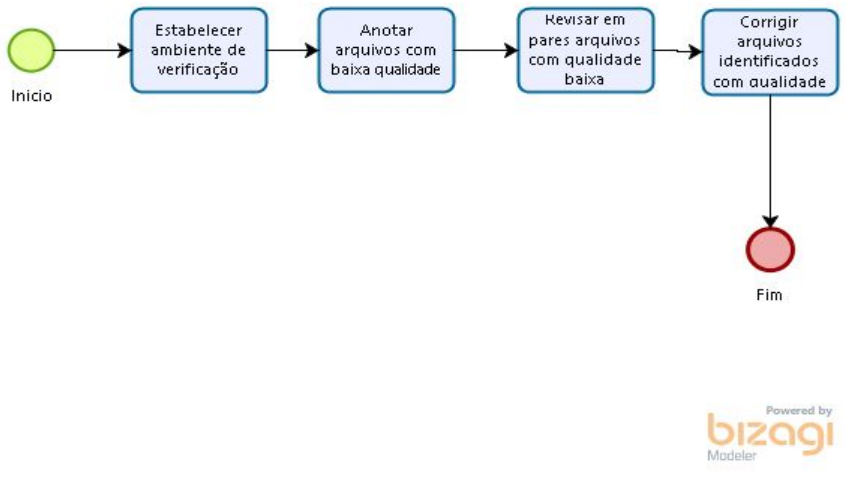
8. Analisar e Documentar Cobertura de Testes

A cobertura dos testes é uma métrica que pode ser usada para mensurar a qualidade do código da aplicação. Então esta atividade tem como objetivo analisar esta cobertura de testes. O mínimo exigido na disciplina de GPP/MDS é de 90%, então esse é o parâmetro exigido nas entregas de cada funcionalidade. Esta cobertura deve ser documentada para que possa analisada ao final de cada ***Sprint*** para verificar se o código continua com qualidade.

9. Documentar Resultados da *Sprint* na Wiki

Esta atividade possui o objetivo de documentar tudo o que foi feito na ***Sprint,*** o que foi bom, ruim, as histórias concluídas e as que ficaram como débito para a próxima ***Sprint.*** Como resultado dessa atividade, será obtido uma documentação que poderá ser utilizada em outras ***Sprints*** ou em outros projetos.

Subprocesso de Verificação



10. Processo de Verificação

10.1. Estabelecer Ambiente de Verificação

Um ambiente com a ferramentas necessárias deve ser estabelecido, no caso em estudo apenas é necessário um notebook com internet, a ferramenta rails versão 5.0 anotada, ruby na versão 5.0.1, rubocop, ferramenta para rodar os testes automatizados.

10.2. Anotar Arquivos de Baixa Qualidade

Ao analisar os arquivos de saída da ferramenta, caso algum apresente resultado menor que a baseline definida eles devem ser monitorados até que sejam resolvidos e a qualidade volte a estar satisfatória, então os arquivos devem ser anotados para que possam ser corrigidos.

10.3. Revisar em Pares Arquivos Com Qualidade Baixa

Os arquivos com baixa qualidade devem ser revisados para que seja encontrado o motivo deles estarem abaixo da qualidade esperada.

10.4. Corrigir Arquivos Identificados Com Qualidade

Após esses arquivos serem examinados, agora é corrigir estes arquivos e fazer com que eles atingiram a qualidade esperada. Com estes arquivos alcançando a qualidade esperada, este processo se encerrará.

c. Métricas Para controle e Estudo

i. Métricas de Qualidade de Código

- **Tamanho das Classes**

O tamanho das classes será medido para que possa ser analisado a coesão do código.

- **ABC metric**

ABC Metric é uma métrica de software que define um score ABC. que representa o tamanho de um conjunto de declarações. O cálculo é feito utilizando número de atribuições (A), número de branchs (B) e número de condicionais (C). ABC Metric também pode ser aplicada a métodos, funções, classes e módulos de um programa.

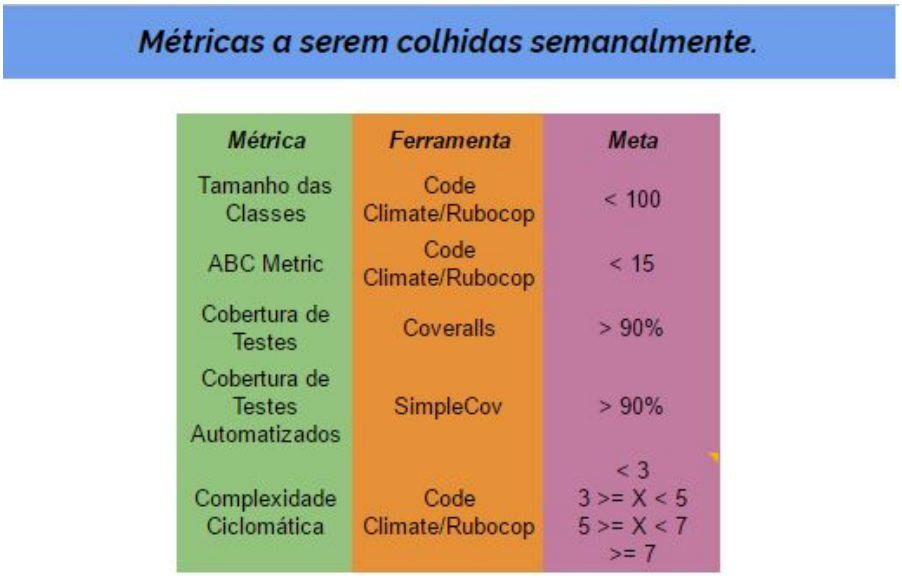
- **Cobertura de Teste**

Cobertura de testes (Code Coverage), é uma métrica utilizada para medir a quantidade de código que foi testada por um conjunto de casos de teste.

- **Complexidade ciclomática**

Complexidade ciclomática serve para mensurar a complexidade de um determinado módulo, a partir do número de caminhos independentes que podem ser executados até o seu fim.

- Os valores usados como referência para demonstrar se a métrica de código está adequada e as ferramentas utilizadas em cada métrica podem ser vistas na imagem abaixo:



ii. Métricas de Processo

- **Produtividade**

Tendo como referência o livro ***Scrum: a arte de fazer o dobro do trabalho na metade do tempo,*** de Jeff Sutherland, um dos principais fatores que influenciam o andamento do trabalho de uma equipe ágil é o humor da equipe durante a semana de trabalho. Para pode analisar o índice de produtividade de uma equipe ágil são analisadas outras duas métricas:

- **Quadro de Sentimentos**

Quadro físico, onde, diariamente todos os integrantes colocariam um ***post-it*** da cor relacionada ao seu humor no dia. Um exemplo do quadro pode ser visto abaixo:

- **Burndown**

Gráfico utilizado para realizar o acompanhamento da equipe, em relação à entrega de pontos e realização de estórias, por ***Sprint.***

12. Execução do Estudo de Caso

a. Descrição dos participantes do estudo

Para o estudo de caso proposto, foi selecionado uma equipe de time ágil da disciplina de GPP/MDS. A equipe escolhida foi a Escola-X.

As metodologias utilizadas na Escola-X abordam práticas de ***SCRUM*** e ***XP***:

- **Papéis**

Scrum Master: É responsável por ajudar a todos os envolvidos a entender e abraçar os valores, princípios e práticas do ***Scrum***.

Product Owner: É o ponto central com poderes de liderança sobre o produto.

Tracker: Acompanha o andamento do projeto nos âmbitos de prazo, custo e qualidade e sugere melhoria dos mesmos.

Desenvolvedor: É responsável pela concepção, construção e testes da aplicação proposta.

- ***Sprints***

  As ***Sprints*** tem duração de uma semana e abrangem alguns ritos específicos

- ***Sprint planning***

  O ***Sprint Planning*** é sempre realizado junto ao ***Sprint Review e tem duração de no máximo 2hr. Nele são definidos os papéis da Sprint, quais histórias são realizadas, o quadro de conhecimentos é atualizado e são geradas as duplas para o pareamento.***

- ***Daily Scrum***

  São feitas reuniões diárias a fim de manter o time alinhado diariamente.

- ***Sprint Review***

  Ao final de cada ***Sprint***, ocorre a ***Sprint Review***, onde são verificadas quais histórias foram concluídas e é feita a adaptação das novas funcionalidades com as antigas

- ***Sprint Retrospective***

  A retrospectiva acontece na forma de todos os integrantes falarem sobre os pontos positivos, pontos negativos e pontos a melhorar na ***Sprint*** seguida.

b. Descrição da execução do estudo de caso

Para a realização do estudo de caso e dentro do período de tempo hábil, foram escolhidas 4 ***Sprints*** para a realização do experimento

- ***Sprints*** **5 e 6**
  O processo foi seguido normalmente, sem a interferência do processo proposto por este trabalho.

- ***Sprints*** **7 e 8**
  O processo ágil de verificação e validação proposto foi aplicado dentro do processo que já estava sendo seguido pela equipe.

  Ao final de cada ***Sprint***, as métricas foram colhidas e documentadas.
  A análise e interpretação dos resultados pode ser vista com detalhes na

seção seguinte.

13.Resultados

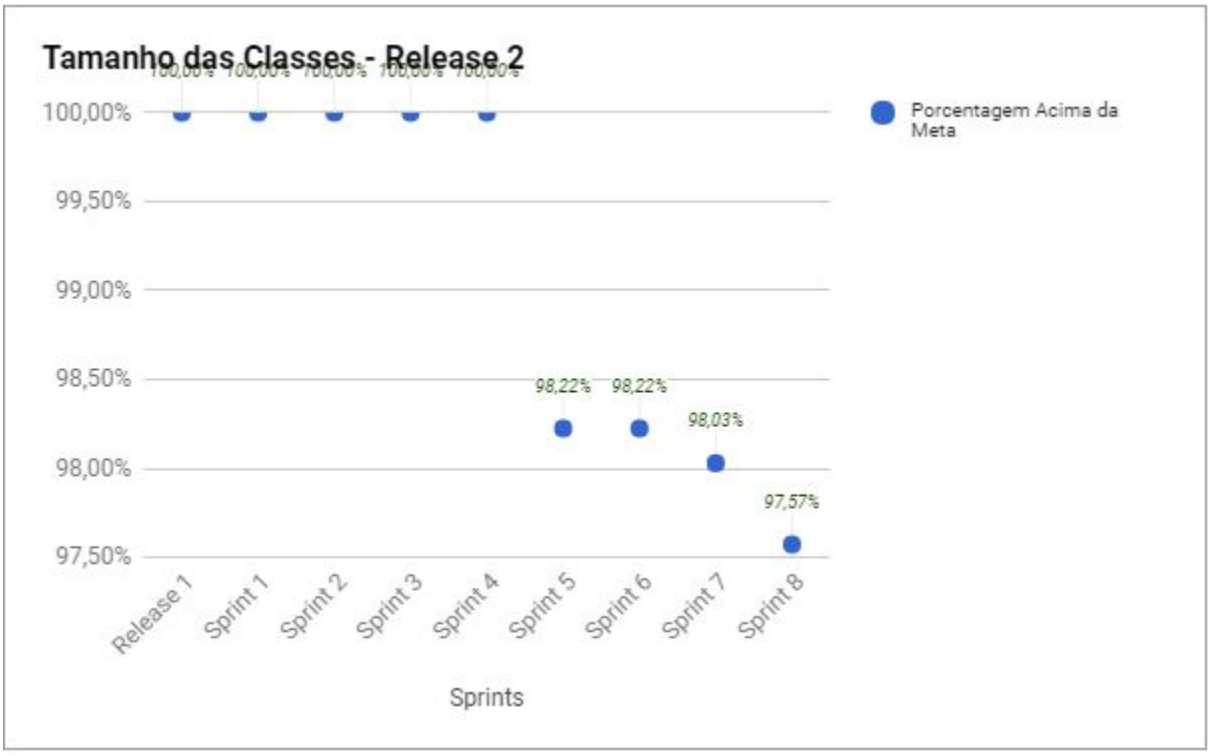
- ***Resultado das Métricas***
    - **De Qualidade do Código**

      Todas as métricas referente a estrutura e qualidade do código feito, foram colhidas e documentadas ao final de cada ***Sprint*** (semanalmente) e, não apenas isso, fora pego também as métricas da ***Release 1,*** para que assim a análise pudesse ser mais abrangente e tivesse mais dados. As baselines de cada métrica está definida no plano de métricas da equipe de GPP (Plano de métricas).

        - Tamanho de Classe

Tamanho de Classes

Sprint	*Qntd de Arquivos*	*Arquivos com Erros*	*Porcentagem Acima da Meta*
Release 1	79	0	100,00%
Sprint 1	103	0	100,00%
Sprint 2	157	0	100,00%
Sprint 3	157	0	100,00%
Sprint 4	157	0	100,00%
Sprint 5	169	3	98,22%
Sprint 6	169	3	98,22%
Sprint 7	203	4	98,03%
Sprint 8	206	5	97,57%



- Fórmula usada para gerar o indicador

$$\left(1 - \left(\frac{QE}{QT}\right)\right) * 100$$

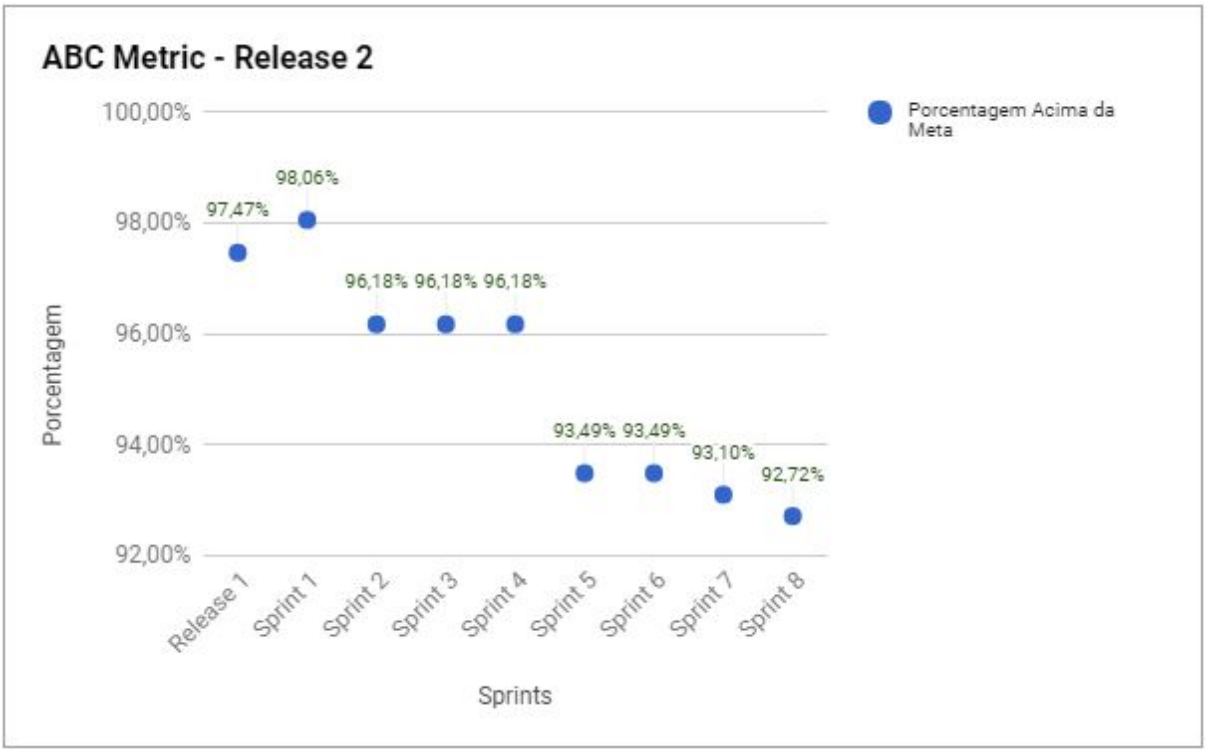
Onde:

QE = Quantidade de Arquivos com erros

QT = Quantidade de Arquivos totais no projeto

- ABC Metric

ABC Metric			
Sprint	***Qntd de Arquivos***	***Arquivos com Erros***	***Porcentagem Acima da Meta***
Release 1	79	2	97,47%
Sprint 1	103	2	98,06%
Sprint 2	157	6	96,18%
Sprint 3	157	6	96,18%
Sprint 4	157	6	96,18%
Sprint 5	169	11	93,49%
Sprint 6	169	11	93,49%
Sprint 7	203	14	93,10%
Sprint 8	206	15	92,72%



- Fórmula usada para gerar o indicador

$$\left(1 - \left(\frac{QE}{QT}\right)\right) * 100$$

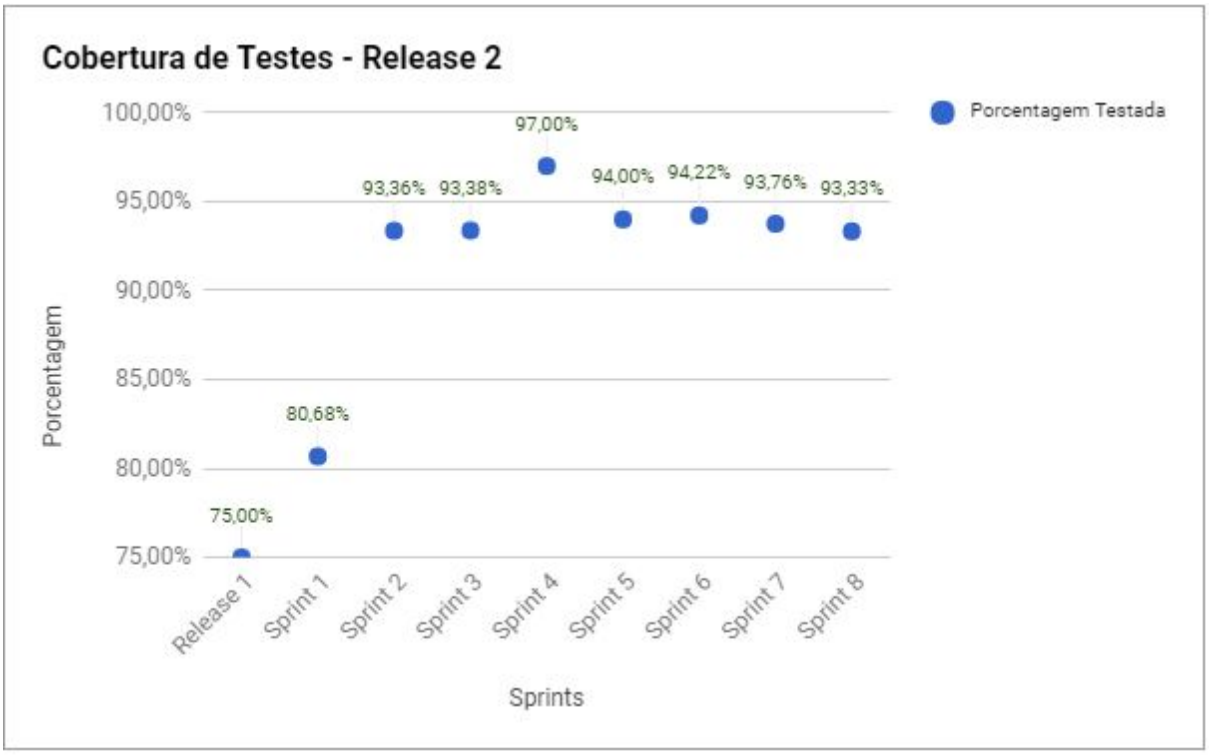
Onde:

QE = Quantidade de Arquivos com erros

QT = Quantidade de Arquivos totais no projeto

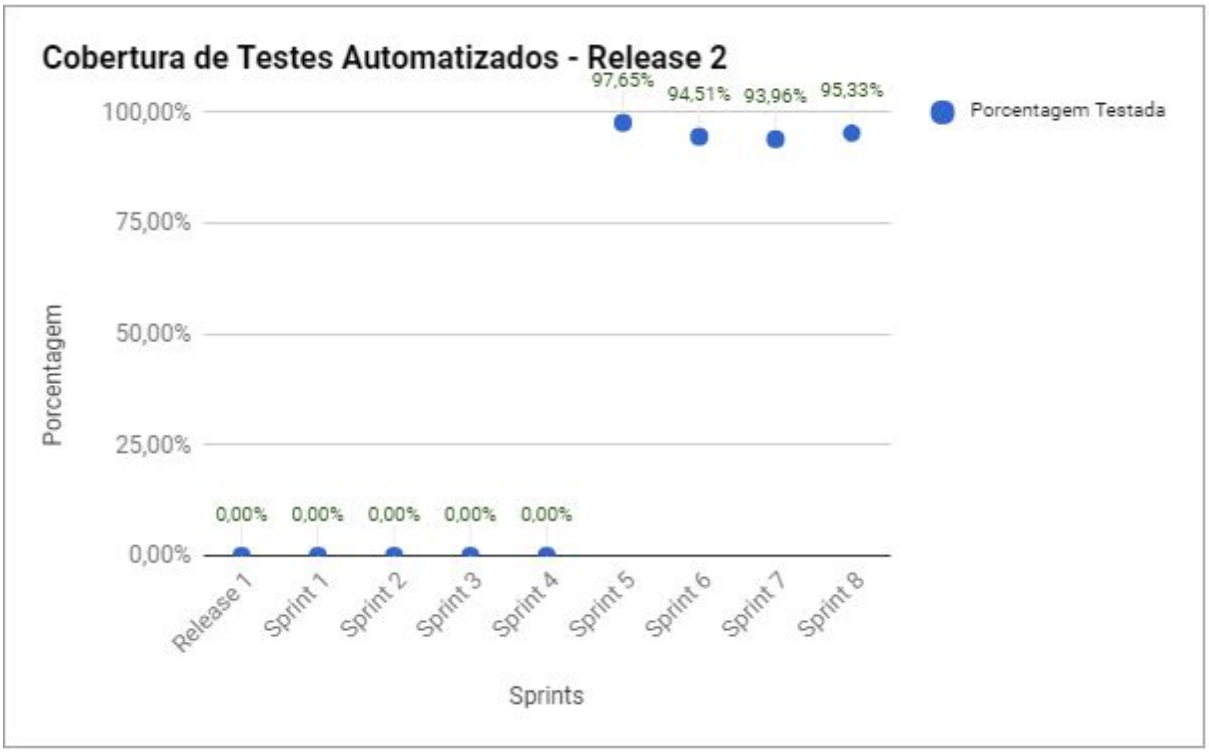
- Cobertura de Testes
  - Unitários

Cobertura de Testes	
Sprint	***Porcentagem Testada***
Release 1	75,00%
Sprint 1	80,68%
Sprint 2	93,36%
Sprint 3	93,38%
Sprint 4	97,00%
Sprint 5	94,00%
Sprint 6	94,22%
Sprint 7	93,76%
Sprint 8	93,33%



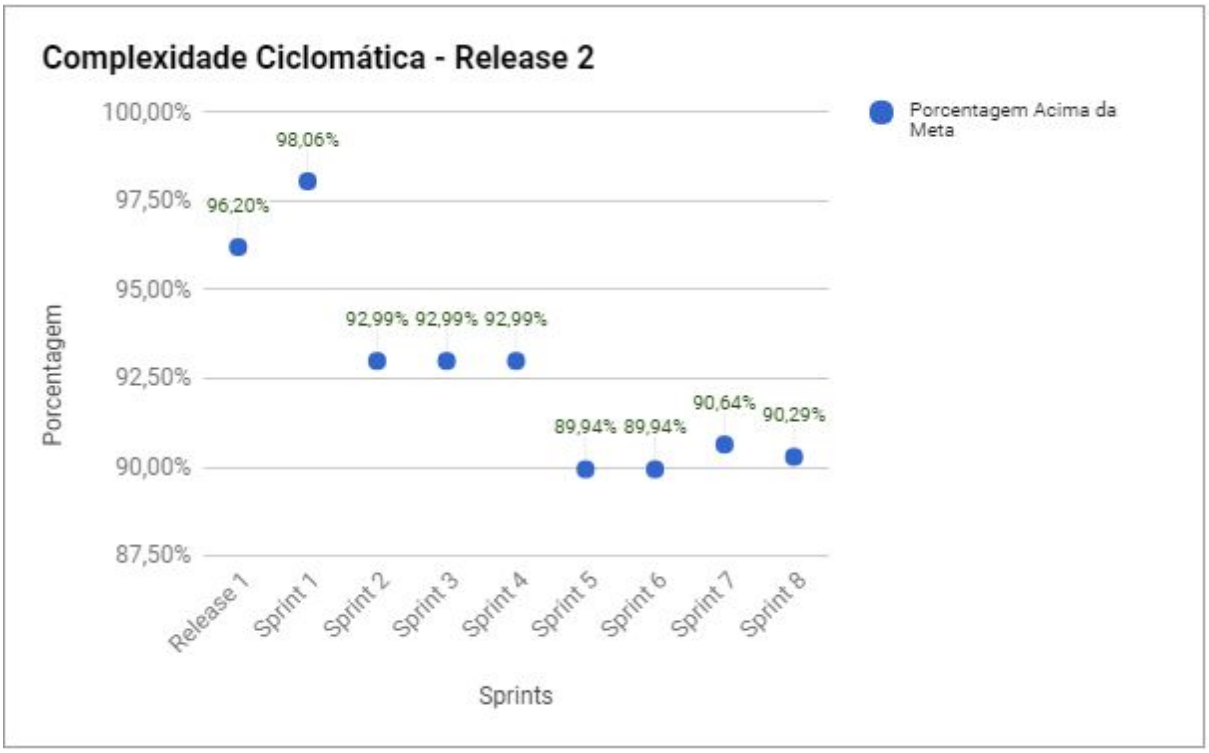
- Automatizados

Cobertura de Testes	
Sprint	***Porcentagem Testada***
Release 1	0,00%
Sprint 1	0,00%
Sprint 2	0,00%
Sprint 3	0,00%
Sprint 4	0,00%
Sprint 5	97,65%
Sprint 6	94,51%
Sprint 7	93,96%
Sprint 8	95,33%



- Complexidade Ciclomática

Complexidade Ciclomática			
Sprint	***Qntd de Arquivos***	***Arquivos com Erros***	***Porcentagem Acima da Meta***
Release 1	79	3	96,20%
Sprint 1	103	2	98,06%
Sprint 2	157	11	92,99%
Sprint 3	157	11	92,99%
Sprint 4	157	11	92,99%
Sprint 5	169	17	89,94%
Sprint 6	169	17	89,94%
Sprint 7	203	19	90,64%
Sprint 8	206	20	90,29%



- Fórmula usada para gerar o indicador

$$\left(1 - \left(\frac{QE}{QT}\right)\right) * 100$$

Onde:

QE = Quantidade de Arquivos com erros

QT = Quantidade de Arquivos totais no projeto

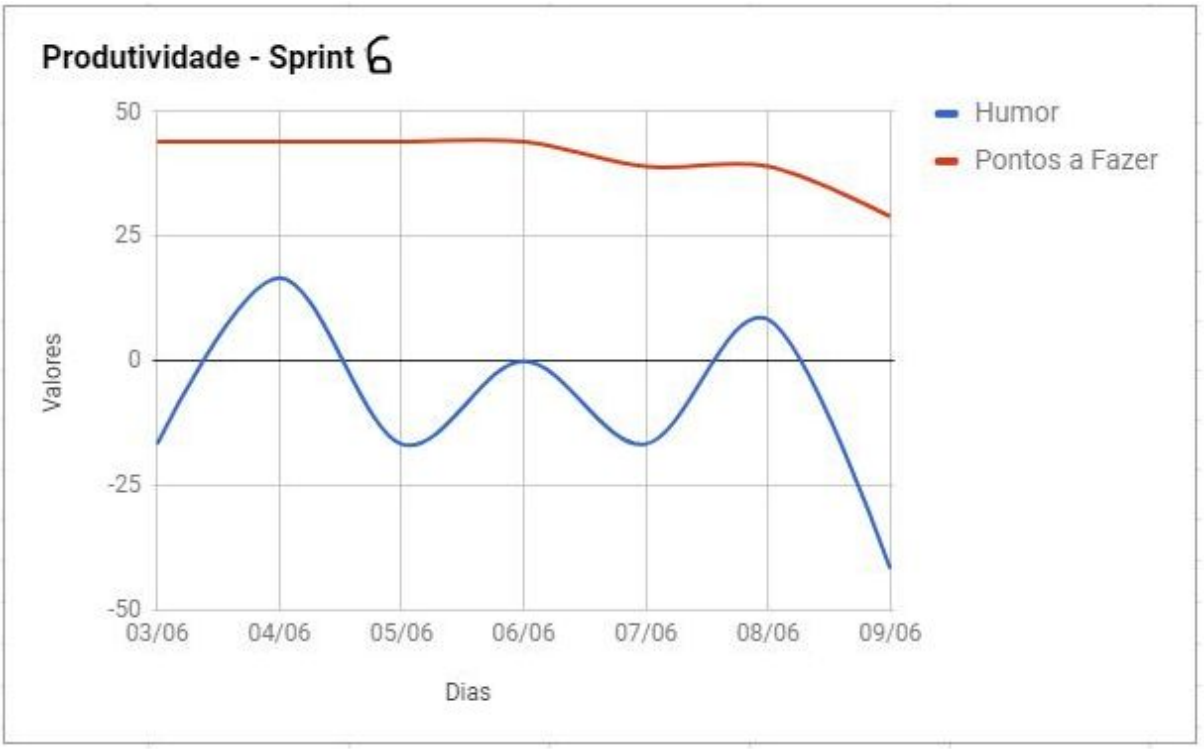
- **De Processo**
  - Mudança dos ***Pontos Negativos*** das retrospectivas
    - Sprint 6


Pontos Positivos
Pontos Negativos
Pontos a Melhorar
Dupla de pareamento
Daily meetings
Integração da equipe
Equipe
Descrição das estórias
Entrega das estórias
Comunicação
Entrega da histórias
Motivação
Comunicação

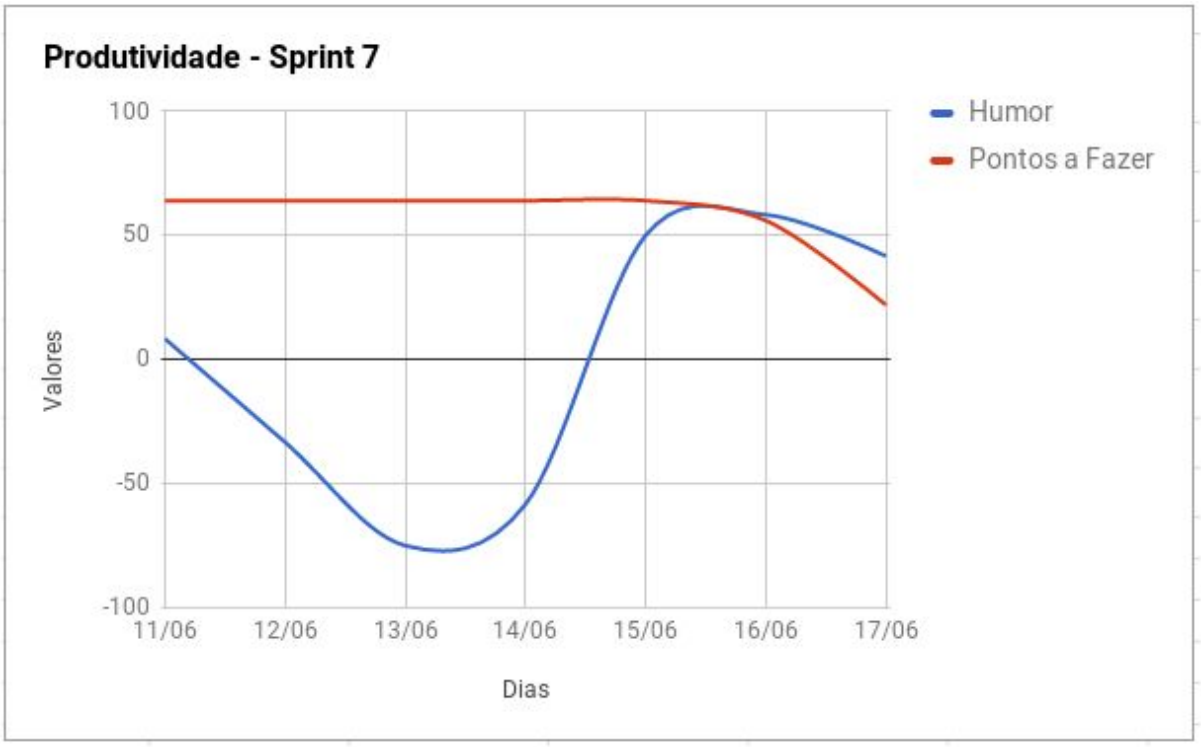
- Sprint 7


Pontos Positivos
Pontos Negativos
Pontos a Melhorar
Dupla de pareamento
Campus Party
Poucas pessoas na reunião
Falta de motivação
Ausência de membros
Motivação
Atualização da wiki
Presenças das dailys

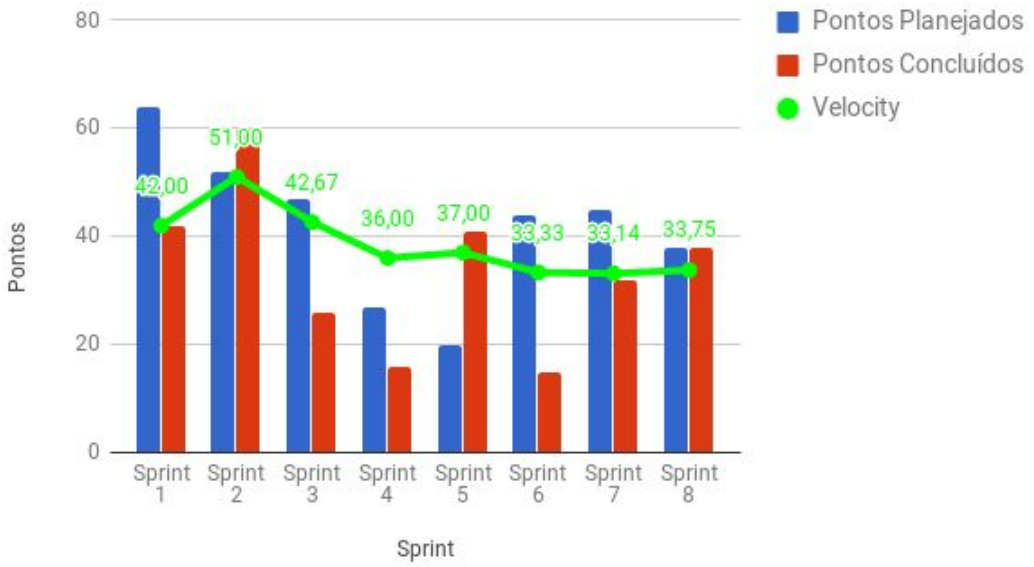
- Aumento de Produtividade
  - Sprint 6



  - Sprint 7



- Velocity com pouca variância



14.Conclusão

Analisando os resultados obtidos, quanto às métricas de qualidade de código, há indícios que a adoção do processo de verificação e validação proposto não influenciou na cobertura de testes, visto que essa medição até piorou. Contudo na complexidade ciclomática, observou-se uma leve regressão nos indicadores, constatada exatamente a partir da Sprint em que o processo foi adotado. Levando a crer que o processo adotado foi importante principalmente na complexidade do sistema, que foi onde foi notada a maior diferença entre as Sprints em que o processo não foi executado e as Sprints com a execução do processo.

Quanto a métricas de processo, observou-se que a quantidade de pontos entregues por sprint se manteve constante nas sprints experimentais e há indícios de que o processo agregado não impactou neste ponto, portanto serão necessários mais estudos de caso para poder tirar resultados mais conclusivos se o processo proposto impacta na quantidade de pontos a ser entregues.

Conclui-se que o processo sugerido neste trabalho é interessante e agrega valor a equipes ágeis, entretanto há indícios que seu impacto é menor em equipes que já estão preocupadas com o valor das métricas medidas e que já atingiram um certo nível de maturidade, que foi o caso da equipe estudada a partir da Release 2, mesmo assim foi possível notar uma discreta diferença no tamanho das classes e principalmente na complexidade do sistema desenvolvido, afetando características

importantes como a facilidade de manutenção do software. Para poder ter resultados mais estáveis, será necessária a aplicação do processo em mais equipes para comparar os resultados de cada uma.

15. Referências

Escola-X, disponível em: https://github.com/fga-gpp-mds/2017.1-Escola-X Acesso em Junho de 2017

Relatório de métricas. Disponível em : https://github.com/fga-gpp-mds/2017.1-Escola-X/wiki/Relat%C3%B3rio-de-Qualidade#43-m%C3%A9tricas. Acesso em Junho de 2017

FGA (GPP/MDS) disponível em: https://github.com/fga-gpp-mds Acesso em Junho de 2017

Boehm, B. W.: Verifying and Validating Software Requirements and Design Specifications, IEEE Software, Volume 1, Issue 1 (January 1984)