UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

FRANCIELLEM MAYARA PONTES BEZERRA

RELATO DE EXPERIÊNCIA NA IMPLANTAÇÃO DE MEDIÇÃO NO DESENVOLVIMENTO DE SOFTWARE EM UMA INSTITUIÇÃO PÚBLICA

BELÉM
2017

FRANCIELLEM MAYARA PONTES BEZERRA

RELATO DE EXPERIÊNCIA NA IMPLANTAÇÃO DE MEDIÇÃO NO DESENVOLVIMENTO DE SOFTWARE EM UMA INSTITUIÇÃO PÚBLICA

Trabalho de Conclusão de Curso apresentado à UFPA, como requisito para obtenção do título de Bacharel em Sistemas de Informação da Faculdade de Computação, Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará.

Orientador: Prof. Dr. Rodrigo Quites Reis

BELÉM
2017

FRANCIELLEM MAYARA PONTES BEZERRA

RELATO DE EXPERIÊNCIA NA IMPLANTAÇÃO DE MEDIÇÃO NO DESENVOLVIMENTO DE SOFTWARE EM UMA INSTITUIÇÃO PÚBLICA

Trabalho de Conclusão de Curso apresentado à UFPA, como requisito para obtenção do título de Bacharel em Sistemas de Informação da Faculdade de Computação, Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará.

Orientador: Prof. Dr. Rodrigo Quites Reis

Data da aprovação: Belém-PA. _____/_____/_____

Banca Examinadora

_____
Presidente: Prof. Dr. Rodrigo Quites Reis – Orientador

_____
Prof.ª. Dra. Carla Alessandra Lima Reis

_____
Prof. Dr. Gustavo Henrique Lima Pinto

BELÉM
2017

AGRADECIMENTOS

Agradeço a minha família por todo o apoio oferecido, por aturarem meus momentos de estresse e por me motivar a sempre lutar pelas minhas metas de vida.

Agradeço imensamente a minha equipe de trabalho por todo o conhecimento que eu adquiri junto a vocês ao longo da minha permanência no estágio. Obrigado pelo apoio, pela paciência nos momentos de pessimismo e pela compreensão em minhas ausências nos períodos finais antecedentes a entrega deste trabalho.

A todos os amigos que me encontravam na rua e perguntavam quando era a minha defesa de TCC ou minha formatura: vocês me pressionavam e motivavam a nunca esquecer meu grande objetivo.

Ao prof. Rodrigo Quites Reis por toda a paciência dedicada, pelas criticas construtivas e por me orientar na execução deste trabalho.

RESUMO

Uma organização com a pretensão de alcançar maior produtividade e uma melhor qualidade no desenvolvimento de software precisa ter em mãos dados quantitativos, estes dados agrupados auxiliam na tomada de decisão da organização. Os dados quantitativos podem ser obtidos através da execução da Medição de Software. A Medição é uma fonte importante de informações para apoiar a tomada de decisão de gestores na organização. Assim, este trabalho relata a experiência em utilizar a medição de software em uma organização de desenvolvimento de software, com a construção do seu Plano de Medição, o acompanhamento das coletas de dados quantitativos e por fim a análise dos dados quantitativos.

PALAVRAS-CHAVE: dados quantitativos, medição de software, análise, tomada de decisão.

ABSTRACT

An organization that aims to achieve higher productivity and improvement on software development process demands to have quantitative data in hand. This grouped data consists of an information set to assist the analysis of possible strategies to base the required changes in a software organization. Quantitative data can be gathered through the application of the software measurement process. Software measurement is the source of important data to support the decision process for software project management. Thus, this work presents an experience report in the use of software measurement in a public organization, with respect to the planning, gathering and analysis of quantitative data.

Keywords: quantitative data, software measurement, analysis, decision making.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 CONSIDERAÇÕES INICIAIS

Em instituições públicas - onde frequentemente não há um cenário de competitividade e nem a busca por lucros financeiros - como encontrado em empresas privadas, o aspecto motivador de desenvolvimento de software está em conseguir melhorar o seu processo e fornecer um melhor serviço para o cidadão, mesmo com a ausência de pressão do mercado capitalista.

Para se obter qualidade no processo um dos primeiros passos a se destacar é mensurar o quanto a organização se conhece e percebe a existência do processo para alcançar os seus objetivos. Uma instituição que não possui esse entendimento, dificilmente consegue implantar melhorias como inovações tecnológicas eficazes e de processo. “Para produzir um processo com qualidade, a princípio é necessário conhecer seu conceito, suas características e aplicar seus métodos e técnicas para obter os resultados desejados” (Guarizzo, 2008, p.11).

Com a finalidade de se obter dados/indicadores quantitativos sobre a performance da organização e sobre as características de seus produtos é feito o uso de medição de software. Medição segundo a Softex tem como propósito “coletar, armazenar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais” (SOFTEX, 2016 p.32).

É interessante para instituição esses dados pois "se você não medir, não há nenhuma maneira real de determinar se você está melhorando. E se você não esta melhorando, você está perdido." (PRESSMAN, 2016, p 720).

A organização pública que servirá como ambiente de estudo para este trabalho é a COTIC, que é uma coordenadoria subordinada a Pró-Reitoria de Ensino de Graduação (PROEG) da Universidade Federal do Pará (UFPA). Foi criada há 8 anos e tem como objetivo atender as necessidades tecnológicas da PROEG. A COTIC atua no desenvolvimento e manutenção de sistemas que atendam na execução de atividades dos setores administrativos ligados a PROEG e também é responsável por gerenciar o *site* da Pró-Reitoria.

No momento da redação deste texto (março de 2017), há nove sistemas concluídos e dois projetos em andamento. Dentre os dois projetos em andamento

um deles é utilizado para realizar o acompanhamento da medição de software neste trabalho. Possui uma equipe formada de nove pessoas, com gestão distribuída entre dois servidores com vínculo efetivo à instituição, e sete bolsistas divididos em períodos da manhã e de tarde.

1.1 Objetivos

1.1.1 Objetivo Geral

Constitui objetivo geral deste texto discutir, planejar, implantar e avaliar o processo de medição de software no ambiente de desenvolvimento da COTIC da PROEG-UFPA.

1.1.2 Objetivos Específicos

Constituem objetivos específicos deste trabalho:

- Estudar técnicas de medição de software;
- Propor um modelo de medição que melhor de adeque ao processo;
- Propor e implantar um plano de medição para a COTIC;
- Realizar o acompanhamento das métricas;
- Registrar as lições aprendidas.

1.2 Metodologia

O procedimento adotado neste trabalho será o de pesquisa-ação. Primeiro foi realizado um levantamento do que a equipe da COTIC deseja medir. Este levantamento foi obtido por meio de reuniões iniciais com a equipe as informações foram escritas e armazenadas. Conhecendo as necessidades da organização serão determinados os objetivos para a construção de um plano de medição.

Com o conhecimento adquirido e se baseando no plano de medição foram iniciadas as ações de coleta de dados durante um período de quatro meses, com acompanhamentos diários, esse tempo determinado junto a equipe. Posteriormente com os resultados obtidos é feito as análises pertinentes no processo da COTIC.

1.3 Organização do Trabalho

Este trabalho está da seguinte forma:

Capítulo 2 – Apresenta o referencial teórico do trabalho, onde estão descritos conceitos de metodologias ágeis como o SCRUM, KANBAN e *eXtreme Programming*.

O capítulo também aborda a conceituação de medição de software e são apresentados os modelos de medição.

Capítulo 3 – Apresenta o estudo de caso na instituição pública, com a caracterização detalhada da COTIC junto com a apresentação do processo que é seguido por ela. É o capitulo onde são apresentados o plano de medição - com o acompanhamento da coleta de métricas e, por fim, os resultados da medição do projeto e os resultados obtidos com a implantação de medição

Capítulo 4 – Apresenta as considerações finais do trabalho dando ênfase aos resultados obtidos.

2. REFERENCIAL TEÓRICO

Neste capitulo são abordados alguns dos processos ágeis existentes usados por equipes de desenvolvimento. Foram escolhidos o Scrum, *eXtreme Programming* e Kanban para serem detalhados aqui por serem os modelos de processo de software adotados na COTIC.

Ainda neste capitulo é apresentado o conceito de medição de software e dois modelos de medição: o GQM e GQIM. Os dois modelos são apresentados por que o GQIM é usado como modelo do plano de medição da COTIC.

2.1 Processos Ágeis de Desenvolvimento de Software

Essa seção apresenta duas metodologias de processos ágeis de desenvolvimento de software e um método ágil de gerenciamento do processo.

- A subseção 2.1.1 aborda a metodologia do Scrum contendo uma contextualização de sua criação, seus papéis e suas praticas de cerimônias.
- A subseção 2.1.2 descreve a *Extreme Programming* e abarca uma breve contextualização de sua criação, posteriormente é explanado seus valores e práticas de utilização.
- A subseção 2.1.3 apresenta o método Kanban inserindo um contexto sobre a sua criação e também descreve seus cinco princípios.

2.1.1 Scrum

A metodologia Scrum foi criada em 1993 por Sutherland e Schwaber, com o intuito de apoiar o desenvolvimento de software. Seu objetivo era melhorar o processo e trazer uma maior eficiência e rapidez, que enfim fosse diferente do modelo Cascata usado na época (Sutherland, 2014).

A metodologia Scrum define alguns papéis para os profissionais envolvidos:

- O *Product Owner* é um papel que pode ser definido como a pessoa capaz de ter as decisões estratégicas do produto, e principalmente cabe a ele representar o cliente para a equipe;

- O *Scrum Master* o integrante da equipe precisa conhecer muito bem o método Scrum, para aplicar da melhor forma as técnicas no processo e ter condição de disseminar o conhecimento também do Scrum para a equipe, cabe a ele a ajudar a tirar os impedimentos e ser facilitador das cerimônias;
- A Equipe Scrum é composta por todos os membros da equipe que participam do desenvolvimento do projeto.

Tendo esses papéis divididos entre a equipe, o Scrum também possui em sua estrutura as chamadas cerimônias. É definido que a equipe deve estimar um tempo para durar cada uma dessas cerimônias. Os parágrafos a seguir descrevem as cerimônias no Scrum.

Sprint é o ponto central da metodologia, fica desprendido um tempo dedicado a execução da iteração da equipe. Esse tempo serve como um ciclo do processo e durante esse ciclo a equipe trabalha em tarefas do projeto buscando no final do ciclo entregar algo de valor para o cliente.

Planning é a reunião de planejamento da Sprint, é facilitada pelo *Scrum Master* e envolve todos os membros da equipe. No *Planning* é decidida qual é a meta da *Sprint* e o que deve ser estimado pela equipe a ser desenvolvido no período de iteração da mesma. São definidas as histórias de usuário e essas histórias compõem o chamado *Product Backlog* do produto. Após as histórias serem escolhidas para compor a Sprint, elas saem do *Product Backlog* e passam para o *Backlog da Sprint.*

Daily consiste em uma reunião de caráter diário. É definido um tempo pequeno de duração para realização. A Daily é o momento de analisar o dia de trabalho da iteração, visando a meta que foi estabelecida. A equipe verifica o que já foi feito para atingi-la, o que será desenvolvido no próximo dia de trabalho e os possíveis obstáculos que podem vir a surgir.

Retrospective é a reunião que encerra o ciclo de desenvolvimento da Sprint. Nela são colhidas as informações para a equipe fazer uma análise para poder fazer um balanço dos resultados obtidos, levantamento das atividades bem-sucedidas e também das que não deram certo. Por fim, essas informações servem para arquitetar as melhorias futuras no próximo ciclo.

2.1.2 *eXtreme Programming* (XP)

As práticas da metodologia *eXtreme Programming* (XP) descritas por Beck e Andres (2004) levaram mais de uma década para se compor e finalmente ganhar a sua estrutura atual. Kent Beck e Ward Cunningham foram os responsáveis por congregar algumas ideias de práticas quando ambos trabalhavam com Smalltalk na Tektronixs. No ano de 1996, Kent utilizou um conjunto dessas práticas em um sistema em que ele trabalhava. Logo, esse conjunto de práticas foi nomeado XP.

Na metodologia XP são determinados quatro valores que são desejáveis para uma equipe estimular em seu ambiente de trabalho. São eles:

- **Comunicação** - a metodologia não se refere apenas a comunicação entre os integrantes da equipe, mas também abrange a comunicação entre equipe que trabalha no software e o cliente a quem o software é destinado. A XP preza essa via de diálogo entre as duas partes interessadas no software, pois com uma constante comunicação entre elas é possível alinhar os conceitos das funcionalidades, tirando dúvidas do caminho do projeto, e também acolher possíveis ideias novas para o software.
- ***Feedback*** - o *feedback* é bom para ordenar as ideias, as necessidades, os acertos e os desvios. É por meio da coleta de *feedback* do cliente que os desenvolvedores são capazes de perceber se estão trabalhando no que realmente o cliente solicitou e necessita. É por meio do *feedback* que o cliente consegue verificar se o software está se tornando o que idealizou.
- **Simplicidade** - a simplicidade no produto consiste em garantir que a funcionalidade requisitada pelo cliente seja entregue como ele solicitou sem muitos acréscimos por meio dos desenvolvedores. Esse valor para a equipe representa o compromisso em se trabalhar apenas no que é necessário para entregar a funcionalidade e não perder tempo em ficar incrementando uma funcionalidade que pode vir a ser solicitada no futuro. Com o uso da simplicidade é possível ter funcionalidades sendo disponibilizadas mais rapidamente para o cliente.
- **Coragem** - consiste em manter um cliente presente capaz de acompanhar o andamento dos trabalhos realizados pela equipe e manter com ele uma comunicação e captura de *feedbacks*. Além disso, a coragem é traduzida

também pela persistência em se utilizar e manter as práticas recomendadas pela metodologia.

Nem só de valores se baseia a metodologia. A XP em sua estrutura recomenda um conjunto de práticas visando melhorar o desempenho da equipe no decorrer do desenvolvimento do software. Estas práticas são apresentadas nos parágrafos a seguir.

Ter um cliente presente: os próprios valores do XP indicam essa prática, pois para ser ter uma comunicação efetiva com recebimentos e envios de *feedback* - é desejável sempre que possível ter um cliente presente no andamento do projeto, para garantir aos programadores confiança na essência das tarefas.

"A **programação em par** é um estilo de programação onde dois programadores trabalham lado a lado em um computador" (WILLIAMS, 2002, pg 16). A prática consiste em dividir entre cada par de programadores os papéis que eles irão desempenhar durante a programação. Mesmo que sejam papeis diferentes existe fortemente a colaboração entre o par, pois eles têm o mesmo objetivo em comum. Os papéis que devem ser assumidos são:

- **Condutor** é a figura com a tarefa de conduzir o trabalho quando o par está definido. É ele a assumir a direção do desenvolvimento, que escreve as linhas de código.
- **Navegador** assume a figura de observador, ele vai verificar o trabalho do condutor e não deixar erros passarem sem ser percebidos. Ocorre com isso uma própria revisão de código no momento em que ele está sendo escrito.

Os programadores podem revezar quem desempenha esses papeis durante o trabalho. Uma pessoa pode começar o desenvolvimento como condutor e depois de algum tempo passar a desempenhar a função de navegador. Esse revezamento pode acontecer durante todo o trabalho dos dois.

Ter um Código coletivo: O código é desenvolvido por várias pessoas da equipe, e a ideia é não permitir que apenas uma pessoa trabalhe em uma funcionalidade, mas estimular a equipe a conhecer e ter a oportunidade de fazer modificações caso seja necessário. Com o uso do código coletivo, todas as pessoas têm autonomia para trabalhar em partes do código e não ficar esperando resultados de apenas uma pessoa.

Padrão de código: Essa prática torna-se interessante para a equipe por causa das propriedades das práticas de programação em par e do código coletivo, como o código pode passa por muitas pessoas, e nada impede que cada um codifique de uma maneira, o ambiente de desenvolvimento poderia se tornar caótico, com um gasto de tempo maior para se entender o código escrito pelo outro. Ter um padrão de código ajuda a homogeneizar a programação de todos e uma maior facilidade em que cada um seja capaz de intender e continuar o código construído.

O Desenvolvimento é conduzido por testes e é estimulada a realização de testes automatizados. A adoção dessa prática no desenvolvimento de software implica no aumento da probabilidade em se ter um sistema implantado com mais qualidade.

> A ocorrência de defeitos é uma realidade presente em qualquer projeto de desenvolvimento de software. É difícil, ou impossível, evitar que aconteçam defeitos no sistema ao longo do projeto, entretanto é possível fazer alguma coisa em relação à quando esses defeitos são detectados e qual o impacto que causarão no cronograma do projeto. E isso faz uma grande diferença na velocidade do desenvolvimento e na qualidade do software (TELES 2014, pg.113).

Na metodologia XP são recomendados dois tipos de testes: o teste de unidade e o teste de aceitação. A realização destes testes deve acompanhar todo o caminho das funcionalidades e devem ser executados várias vezes.

Na Orientação a Objetos, o teste de unidade se preocupa em testar os métodos das classes. É recomendado que antes dos programadores começarem a trabalhar na funcionalidade eles planejem e montem os testes de unidade. Nesse teste as classes são verificadas para atestar o seu bom funcionamento e não deixar passar eventuais falhas. Isso significa inclusive refazer todos os testes de unidade cada vez que uma nova classe é construída para analisar se algo foi prejudicado por essa nova adição.

Os testes de aceitação são um complemento dos testes de unidade, e neles são testadas as respostas do sistema. Primeiramente é feita uma lista de rotas que o usuário final pode tomar ao usar o sistema, e o teste de aceitação verifica se o sistema responde da maneira correta ao usuário. O objetivo de se realizar esse tipo de teste é mapear todo e qualquer cenário no sistema e assim entregar ao cliente as funcionalidades que o atendam corretamente.

***Design* simples.** A questão é atender da forma mais simples possível o que o cliente deseja, sem acréscimos dos desenvolvedores que podem ou não a vir ser usados. O *design* simples é uma prática que reflete diretamente o valor simplicidade da XP.

Refactoring. É verificado se o código escrito é legível, se há a capacidade de qualquer pessoa que futuramente tenha que dar manutenção no código consiga entende-lo de forma rápida e fácil. A *Refactoring* não busca mudar o objetivo de uma funcionalidade, mas sim a forma como a funcionalidade está escrita. É desejável o emprego de um código mais limpo sem a existência de duplicações de linhas de código.

Planning Game. A prática é destinada a todos os planejamentos do projeto. Como na XP é desejável ter um cliente presente, ele também deve estar junto com a equipe nesse momento de planejamento. O cliente descreve o que ele almeja em um cartão, e essa descrição é usada para compor as historias de usuário. De posse desses cartões a equipe estima o custo de implementar as historias de usuário, mantendo uma transparência no trabalho a ser executado para o cliente. A equipe pode também empregar o esquema de usar pontos para ajudar nas estimativas de cada historia. O ponto representa para o desenvolvimento o dia ideal de trabalho. Com isso a equipe contabiliza a quantidade de pontos que a história necessita para ser concluída.

Com os cartões prontos e estimados a equipe pode planejar um período de entrega de um conjunto de funcionalidades, as chamadas *releases* do projeto. Nas *releases* a equipe se compromete com o cliente a entregar algo funcional do software para ele. Para isso ser realizado é feito o planejamento também das iterações de trabalho. As iterações são realizadas em períodos pequenos (semanas) e servem para trabalhar nas historias prioritárias para estarem prontas quando o prazo da *release* se encerrar. O somatório de pontos das historias prioritárias determina a velocidade da iteração da equipe para o cliente.

A prática de ***Stand up Meeting*** na metodologia XP está presente a execução de uma reunião diária onde os integrantes ficam em pé para discutir quais historias da iteração que serão trabalhadas ao longo do dia. A *Stand up Meeting* serve para alinhar a equipe do dia de trabalho, com o intuito de alcançar maior produtividade.

A próxima a ser descrita é a Integração continua, como no desenvolvimento de software pode ter mais de um programador trabalhando em uma mesma tarefa em momentos diferentes, surge a necessidade de haver uma integração no código para evitar ao máximo conflitos nas modificações realizadas, e quanto mais cedo for percebido os conflitos, mas fácil saber onde ocorreu o problema.

***Releases* curtos**: a prática consiste em trabalhar com *releases* curtos e permite para o cliente o retorno mais rápido do investimento dispendido para o projeto. Com esta prática, a equipe se compromete a entregar as funcionalidades prioritárias do cliente em períodos curtos.

Metáfora é a prática que objetiva manter um alinhamento de ideias em todo o decorrer do processo, para que não haja uma convergência de objetivos, onde cada um almeja algo diferente.

Ter um **Ritmo sustentável**: essa prática está preocupada com as pessoas que atuam no projeto, com a qualidade de vida delas. Para isso se busca trabalhar no projeto sempre o tempo estipulado para jornada de trabalho diária e sem obrigar ao programador a sobrecarga de horas extras.

2.1.3 KANBAN

Kanban é um método criado por Taiichi Ohno, que presidia o sistema Toyota de produção. A ideia derivou-se da organização utilizada nos supermercados dos Estados Unidos, onde a tarefa de reabastecer a prateleira apenas quando os clientes já tivessem feito a retirada dos produtos da prateleira. Ou seja, os produtos só voltavam a prateleira quando restavam poucos ou nenhum disponível ao cliente.

Tendo visualizado essa forma de organização a Toyota decidiu utilizar essa ideia em sua produção com o uso de cartões. Nestes cartões vinham descritas informações relevantes para o trabalho. A meta era impedir o desperdício com superprodução de produtos e, para isso, os cartões eram usados para acompanhar a trajetória da produção.

O uso do método Kanban no desenvolvimento de software passou a ser difundido em 2007, por David Anderson e Rick Garber na conferência "*Lean New Product Development*" com relatos de suas experiências da adoção do método na empresa Corbis e, posteriormente, também apresentado no *Agile 2007*. Com esses

debates e trocas de experiências foi possível circular entre outros desenvolvedores o interesse de se usar o método em seus locais de trabalho (Anderson 2011, p. 8).

Segundo Anderson (2011), há cinco princípios que precisam ser levados em conta:

1- Visualização do fluxo de trabalho;
2- Ter um limite de trabalho em progresso (*work in progress* - WIP);
3- Medir e gerenciar o fluxo;
4- Ter políticas de processo explícitas;
5- Utilizar modelos para reconhecer possíveis melhorias.

O objetivo dessas propriedades é haver para organização um retorno satisfatório nos seus ganhos financeiros tal como uma equipe com um novo modo de interagir no processo utilizado no ambiente de trabalho.

A Figura 1 exemplifica um quadro Kanban, com cartões de *user story* (história de usuário) distribuídos em raias de desenvolvimento. Consta também na parte superior do quadro o limite de trabalho nas raias (Wip) considerado pela equipe. Assim não só o trabalho fica visível, mas também quem está trabalhando na *user story*.

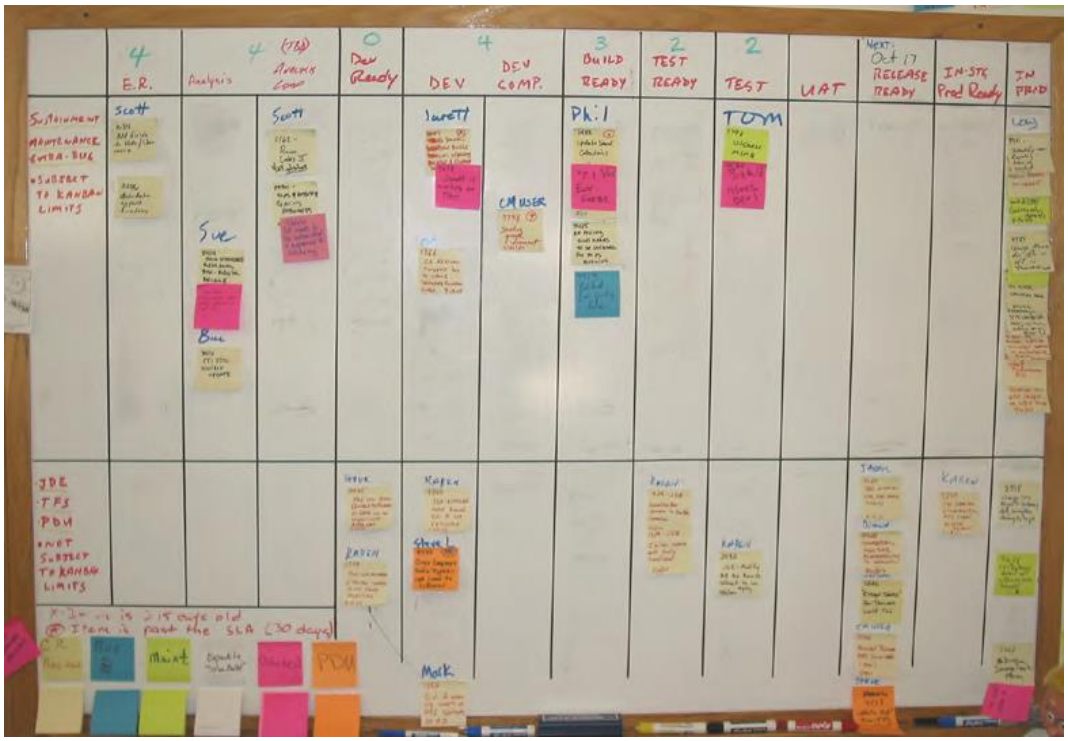

Figura 1 Quadro Kanban [Anderson 2011]

O método Kanban é adaptativo desde sua essência com sua concepção na área automobilística. Sendo assim é um método que pode ser usado por diferentes equipes e também de diferentes formas. Portanto, não há uma “receita” de como utilizá-lo, cada equipe pode adaptá-lo como melhor atender seu ambiente.

2.2 MEDIÇÃO DE SOFTWARE

Esta seção apresenta o conceito geral de medição no âmbito de desenvolvimento de software.

- A subseção 2.2.2 descreve sobre métricas que possuem o foco no processo, seus objetivos de uso.
- A subseção 2.2.3 trata do tipo de métrica de tamanho, é detalhado o tipo linhas de código (LOC).
- A subseção 2.2.4 aborda métricas do tipo funcional e para isso foi escolhido a análise de pontos por função (APF).

2.2.1 Conceito Geral

Conceitualmente para Pressman (2016, p. 654), "medição é o ato de determinar uma medida". Por sua vez, as medidas podem ser definidas como um meio quantitativo que servem para indicar algo sobre o produto ou processo.

Usa-se medição de software quando se quer ter em mãos os dados que indiquem o caminho mais adequado que a organização deve seguir. Sem o uso da medição a tomada de decisão fica embasada apenas na experiência que o gestor possui, decisão essa que pode mostrar-se correta ou errada.

A medição tem como principal foco apoiar a tomada de decisão em relação aos trabalhos, processos e atendimento aos objetivos organizacionais.

As métricas funcionam como um conjunto de medidas que servem para gerar dados quantitativos para os gestores; e, com a ajuda desses dados coletados, a empresa é capaz de gerar uma auto-avaliação do seu processo, desde que as métricas sejam estruturadas com o uso de objetivos claros para a organização, pois sem objetivos as métricas são apenas números soltos sem propósito e sem destino.

2.2.2 Métricas com enfoque no processo

Desenvolver software não é algo simples e por mais que seja trabalhoso as organizações precisam ter outra preocupação, que é a qualidade dos produtos desenvolvidos por ela. As organizações que têm a pretensão de serem competitivas no mercado precisam buscar entregar software com qualidade para o cliente, pois assim a organização pode almejar conseguir reconhecimento na área. Para isso os processos da organização devem ser pensados cuidadosamente pelos gestores. O processo é, de fato, importante para uma organização de desenvolvimento de software.

> Para que se tenha um setor de software competitivo, nacional e internacionalmente, é essencial que os empreendedores do setor coloquem a eficiência e a eficácia dos seus processos em foco nas empresas, visando à oferta de produtos de software e serviços correlatos conforme padrões internacionais de qualidade e desta forma, aumentando também a produtividade nas empresas" (SOFTEX, 2016 p.6).

Gestores que buscam melhorias de seus processos podem fazer uso das medições de processo, para indicar o caminho de possíveis melhorias.

Segundo Sommerville (2011, p.497)

> As medições de processo são dados quantitativos sobre o processo de software, tal como o tempo necessário para realizar alguma atividade do processo [...] As medições do processo podem ser usadas para avaliar a melhoria da eficiência de um processo.

Há inúmeras possibilidades para usar métricas para melhoria do processo. Pode-se verificar, por exemplo, se ocorrem muitos "gargalos" no processo. Para isso as métricas com esse foco são coletadas durante algum período de tempo com o objetivo de aprimorar o processo adotado. Enfim, com os dados coletados pode-se usar as métricas como base para buscar ideias para solucionar as não conformidades encontradas e evitar que os problemas continuem ocorrendo.

2.2.3 Métricas de Tamanho: Linhas de Código (*Lines of Code* - LOC)

Uma métrica de tamanho tem o objetivo de medir o tamanho do software desenvolvido. Uma técnica utilizada para medir o tamanho do software é fazer a contabilização da quantidade de linhas de código fonte que o mesmo foi escrito.

Projetos mesmo desenvolvidos pela mesma equipe são diferentes um do outro. Produtos de software invariavelmente possuem diferentes números de linhas de código, uns com uma quantidade maior e outros com quantidades menores. Consequentemente os que têm uma quantidade maior de linhas de código podem ter empregado maior esforço da equipe e podem resultar em maiores quantidades de erros e defeitos em comparação com o software que foram escritos com menos linhas de código.

Como os projetos têm tamanhos diferentes à equipe precisa encontrar uma forma de padronização dos resultados da medição e assim ter parâmetros para buscar melhorias na qualidade dos softwares produzidos “métricas de software orientadas a tamanho são criadas normalizando-se as medidas de qualidade e/ou produtividade considerando o tamanho do software que foi produzido” (PRESSMAN, 2015).

Ainda segundo Pressman (2015) há os seguintes possíveis cenários para uso da quantidade de linhas de código para ajudar a equipe a padronizar os projetos:

1- É verificado quantos erros ocorrem a cada determinada quantidades de linhas de código, por exemplo, contabilizar os erros a cada KLOC (mil linhas de código).
2- É verificado quantos defeitos ocorrem a cada determinada quantidades de linhas de código.
3- O custo financeiro de determinada quantidade de linhas de código.
4- O número de páginas esperadas por determinado número de código-fonte.

2.2.4 Métricas funcionais: Análise de ponto de função (APF)

Em seu artigo Albrecht (1979) relata o trabalho dele e de sua equipe em experiências de medir a produtividade dos projetos da IBM.

Após o trabalho pioneiro de Albrecht a análise de pontos de função foi ganhando espaço, e em 1986 foi criado um grupo internacional de análise de ponto de função. O *International Function Point Users Group* (IFPUG) é um grupo internacional responsável por fomentar a análise de ponto de função, ele age

promovendo encontros anuais e certificação para a comunidade, entre outras ações[1].

O objetivo em utilizar a análise de pontos de função, é conseguir fazer a medição do software baseado na visão do usuário. Como é buscado medir suas funcionalidades baseadas na visão do usuário o foco então dessa métrica de funcionalidade é medir seu aspecto funcional e não a tecnologia ou linhas de código.

Como pontos por função busca medir as funcionalidades, precisou-se criar uma forma de contagem dessas funcionalidades, segundo a publicação de Albrecht e Gaffney (1983) é explicado o tipo de contagem presentes na análise da seguinte forma:

- Arquivos Lógicos Internos: é um agrupamento lógico de dados do ponto de vista do usuário que é gerado, usado e mantido nas fronteiras da aplicação.
- Arquivos de Interface Externas: são também um grupo lógico de dados, porém diferente do tipo anterior, estão dentro da fronteira de outra aplicação externa.
- Entrada externa: contam cada processo de dados de usuário ou controle de informações que vêm de fora da fronteira da aplicação, ou seja, esse tipo inclui a entrada de dados do usuário ou entradas feitas de outra aplicação.
- Saída Externa: é o envio de dados ou de controle de informações fora da fronteira da aplicação, ou seja, as saídas externas são os tipos que fornecem informações para o usuário, tais como: relatórios ou mensagens para o usuário.
- Consultas Externas: são consultas de dados feitas pelo usuário que geram uma saída imediata fora das fronteiras da aplicação.

No Manual de Práticas de Contagem de Pontos de Função mantido pelo IFPUG é exposto o método de medição FSM para medir tamanho funcional, o mesmo possui seis passos que são descritos nos parágrafos seguintes:

Reunir a documentação disponível – é preciso ter reunido à documentação do software. O intuito é ter em mãos a listagem e o controle de quais funcionalidades o software está abarcando ou até mesmo das funcionalidades que podem vir a sofrer impactos pelo software desenvolvido.

[1]Mais informações disponível em: <http://www.ifpug.org>

Determinar o escopo e a fronteira da contagem, identificando os requisitos funcionais do usuário – Depois de ser reunida a documentação o próximo passo abrange três necessidades, a primeira é determinar o escopo da contagem, é estabelecido o seu objetivo e com o objetivo estabelecido é escolhido o tipo de contagem a ser feita, se é de projeto desenvolvido, aplicação ou de projeto de melhoria. A segunda necessidade é estabelecer qual o limite da aplicação que está sendo medida, ou seja, estabelecer até onde é um limite de uma aplicação, por fim levantar os requisitos funcionais.

Medir funções de dados – As funções de dados envolvem os Arquivos Lógicos Internos e os Arquivos de Interface Externa. Deve ser feito um agrupamento dos dados lógicos do software e para cada dado lógico do agrupamento identificar quais são os Arquivos Lógicos Internos e os de Interface Externa. Para cada arquivo deve ser contabilizado a quantidade de arquivos referenciados e os itens de dados referenciados. Após a contabilização é então verificada a complexidade mostrada na Tabela 1 e o tamanho na Tabela 2 é determinado depois de se ter o resultado da complexidade.

Tabela 1 – Complexidade das Funções de Dados

	1-19 itens de dados referenciados	20-50 itens de dados referenciados	>50 itens de dados referenciados
1 dado referenciado	Baixa	Baixa	Média
2-5 dados referenciados	Baixa	Média	Alta
>5 dados referenciados	Média	Alta	Alta

Tabela 2 – Tamanho das Funções

	Arquivo Lógico Interno	Arquivo de Interface Externa
Baixa	7	5
Média	10	7
Alta	15	10

Medir Funções de Transação – As funções de transação englobam os tipos de Entrada, Saída e Consulta Externa. Cada ação do usuário deve ser identificada e classificada entre uma das três opções de funções transacionais. Como é feito nas funções de dados deve ser contabilizado arquivos referenciados e os itens de dados referenciados. A complexidade funcional é verificada com os dados contabilizados, porém diferente das funções de dados que a tabela é homogênea para os dois tipos, a complexidade das funções transacionais é diferente. Para Entradas Externas é distribuído um intervalo de valores e para Saídas e consultas outro intervalo. Mostrado nas Tabelas 3 e 4.

Tabela 3 – Complexidade das funções de Entrada Externa

	1-4 itens de dados referenciados	5-15 itens de dados referenciados	>15 itens de dados referenciados
0-1dado referenciado	Baixa	Baixa	Média
2 dados referenciados	Baixa	Média	Alta
>2 dado referenciado	Média	Alta	Alta

Tabela 4 – Complexidade das funções de Saída e Consulta Externas

	1-5 itens de dados referenciados	6-19 itens de dados referenciados	>19 itens de dados referenciados
0-1dado referenciado	Baixa	Baixa	Média
2-3 dados referenciados	Baixa	Média	Alta
>3 dados referenciados	Média	Alta	Alta

O tamanho é determinado depois de ser aferida a complexidade das funções transacionais, como mostrado na Tabela 5.

Tabela 5 – Tamanho das Funções Transacionais

	Entradas	Saídas	Consultas
Baixa	3	4	3
Média	4	5	4
Alta	6	7	6

Calcular tamanho funcional – o calculo para determinar o tamanho funcional depende do tipo de contagem escolhido:

1. Tipo Projeto de Desenvolvimento: DFP = ADD + CFP, onde:
   DFP significa a contagem de pontos de função,
   ADD o tamanho das funções que serão entregues,
   CFP é o tamanho da funcionalidade de conversão.
2. Tipo de Aplicação: AFP = ADD, onde o AFP representa a contagem de pontos de função da aplicação.
3. Tipo Projeto de Melhoria: EFP= ADD + CHGA + CFP + DEL, onde:
   EFP é a contagem de pontos de função para o Projeto de Melhoria,
   ADD representa nos Projetos de Melhoria o tamanho das funções incluídas com o projeto,
   CHGA indica o tamanho das funções alteradas pelo projeto e
   DEL é o tamanho das funções excluídas.

Documentar e reportar – Documentar ao final relatando o acompanhamento da contagem de pontos de função e por fim reportar o padrão utilizado no trabalho.

2.3 Modelos de Medição

Nessa seção do capitulo são descritos os modelos de medição GQM e GQIM. Em cada subseção é relatado a estrutura cada um dos dois modelos.

2.3.1 GQM (*Goal Question Metric*)

O modelo *Goal Question Metric* (GQM) é definido por (Basiliet al., 1994) como uma abordagem que se baseia na ideia de antes de se começar a medição deva-se ter bem especificado para a organização quais são os objetivos que ela possui e quais são os objetivos dos projetos pelos quais ela é responsável.

O *framework* que o GQM fornece é dividido em três camadas (figura 2), cuja a definição deve ser feita do topo para baixo. A primeira camada é referente aos objetivos da organização para o processo de medição. A segunda camada é a responsável por mostrar as questões colhidas que busca responder a equipe sobre o objetivo. A terceira camada é onde fica de fato os dados quantificáveis que são as medidas.

Como já informado anteriormente a definição no GQM é feita de cima para baixo, onde primeiramente são definidos os objetivos da medição, posteriormente as questões e, por fim, as medidas. Diferentemente da definição, a interpretação segue o sentido de baixo para o topo. Assim, ao se obter os dados das medidas se responde as questões, e se verifica se os objetivos foram alcançados.

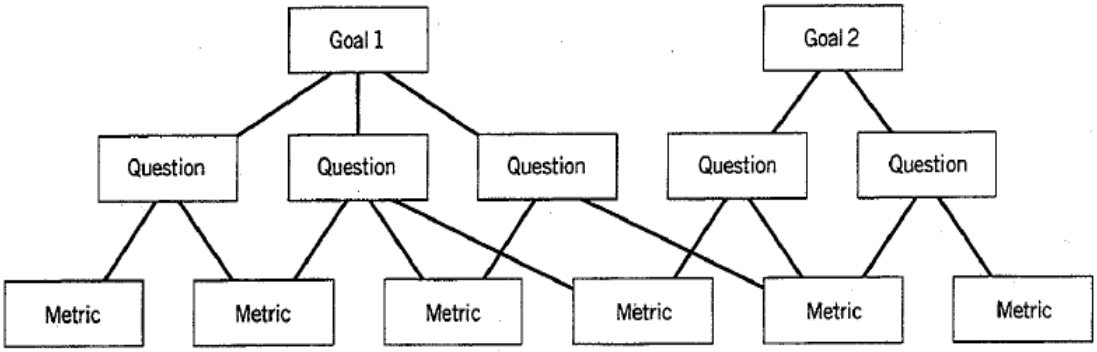

Figura 2 Modelo GQM [Basiliet al 1994].

2.3.2 GQIM (*Goal Question Indicator Measure*)

Em 1996, o *Software Engineering Institute* (SEI) da Universidade de Carnegie Mellon nos EUA (Park et al., 1996) propôs uma nova abordagem derivada do GQM, sendo denominada como GQIM. A abordagem do GQIM é uma evolução do GQM, trazendo assim uma ideia mais expansível da abordagem.

Uma das mudanças introduzidas no GQIM é o uso de Indicadores que servem para dar melhor estruturação no modelo de medição e, com isso, obter análises mais eficientes dos dados coletados.

O GQIM faz uso do que já era utilizado pelo GQM, porém acrescentando uma sequência de dez passos capazes de dar para organização uma maior segurança

em definir um modelo de medição que se mais esteja adequado para ser usado por ela. Os dez passos são (Park et al., 1996):

1. Identificar seus objetivos de negócio;
2. Identificar o que você quer saber ou aprender;
3. Identificar suas submetas;
4. Identificar as entidades e atributos relacionados à suas submetas;
5. Formalizar suas metas de medição;
6. Identificar questões quantificáveis e respectivos indicadores que você irá usar para ajudá-lo a atingir seus objetivos de medição;
7. Identificar os elementos de dados que você irá recolher para construir os indicadores que ajudam a responder as suas perguntas;
8. Definir e padronizar as medidas a serem utilizadas;
9. Identificar as ações que você vai tomar para implementar as medidas;
10. Preparar um plano de execução das ações.

A ideia do GQIM é seguir passo-a-passo respondendo cada item para, enfim se fazer um plano de medição tendo mais claro cada objetivo, questão, indicador e as medidas utilizadas.

A figura 3 ilustra o encadeamento dos passos do modelo GQIM, descritos anteriormente, indicados com a numeração correspondente a cada passo. O passo 1 e 2 representam a linha de partida do plano de medição GQIM, com a definição do que quer ser alcançado, o que é preciso para ser alcançado esse objetivo, por fim é definido o que a organização quer saber. Com essas respostas é possível formar um modelo mental do que consiste o processo e o que o detém.

O passo 3 e 4 fica o levantamento das submetas, entidades e atributos, ao fim desse dessa parte inicial.

Tendo sido colhido essas informações dos passos anteriores, os passos 5, 6, 7 e 8 são formadas a árvore binária do modelo GQIM.

Na ultima etapa fica a área de definição com a obtenção da lista de definições e formulário de regras. Nessa etapa ficam o passo 9 que consiste em ser feito uma analise e diagnostico das duas listas obtidas, por fim no passo 10 é a istagem do plano de implementação.

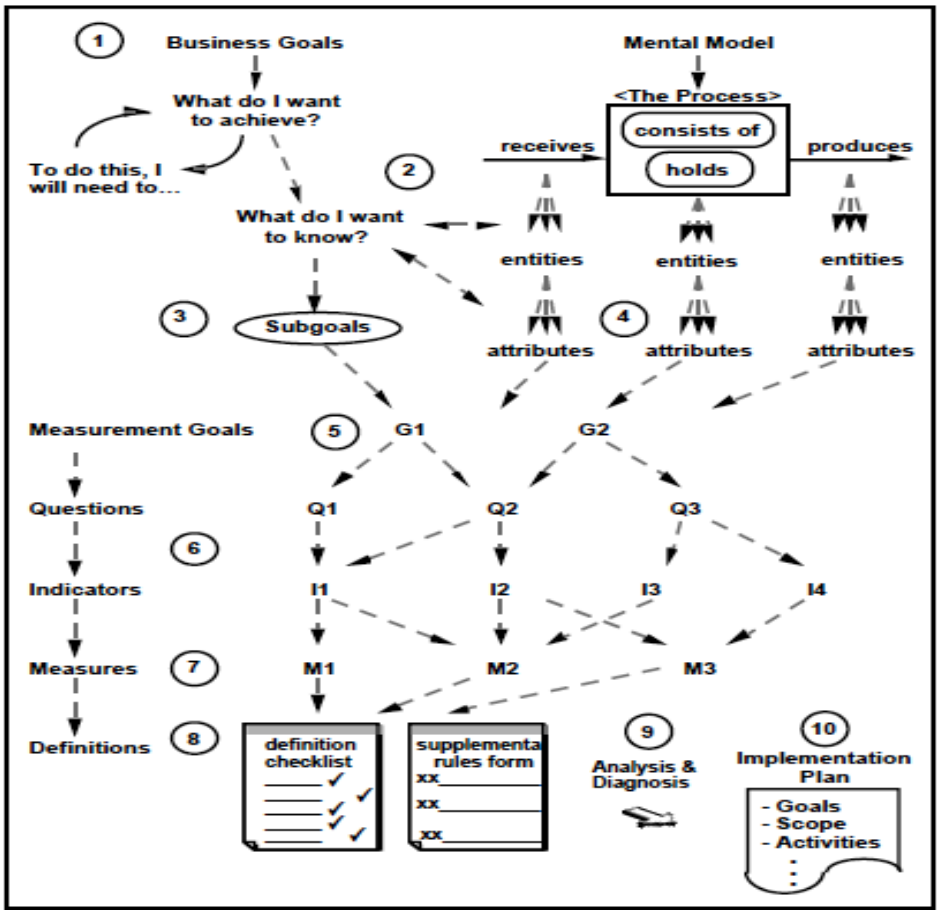

Figura 3 – Abordagem do GQIM [Park et al., 1996].

3 ESTUDO DE CASO EM MEDIÇÃO DE SOFTWARE

Esse capítulo descreve o ambiente da coordenadoria usado como estudo de caso deste trabalho, introduz o plano de medição construído para atender a COTIC, por fim expõe os resultados obtidos.

- A seção 3.1 caracteriza o processo que a equipe segue na COTIC.
- A seção 3.2 apresenta as necessidades e resultados esperados pela a equipe com a implantação do uso de medição de software.
- A seção 3.3 descreve o projeto que é acompanhado com o plano de medição.
- A seção 3.4 decompõe o plano de medição construído para o ambiente.
- A seção 3.5 insere a coleta de dados.
- A seção 3.6 ilustra os resultados e faz a análise da coleta de dados

3.1 Processo Seguido Pela COTIC

A COTIC emprega em seu processo o uso de três metodologias ágeis, que são: Scrum, Kanban e XP. Nessa seção é descrita a forma como estas metodologias são adotadas pela equipe.

A ***Sprint*** tem a duração de duas semanas. Porém, esse período pode ser estendido devido a ocorrência de desfalques de dias, ocasionado em decorrência de feriados, por exemplo. O objetivo é manter com frequência a duração de dez dias por iteração.

O ***Planning*** geralmente ocorre na primeira segunda feira do período de iteração. Como a maioria da equipe é composta por bolsistas que estagiam em apenas um turno é reservado um tempo onde há a possibilidade de estar presente a maioria das pessoas. E, nesse tempo, são escolhidas as histórias prioritárias no *Backlog* para compor a próxima Sprint, podendo ser inclusive de projetos diferentes.

Depois de escolher as histórias prioritárias é jogado o *Planning Poker* para se estimar o valor de esforço de cada história para a equipe. A prática de *Planning*

Poker é descrita segundo Cohn como uma prática onde os participantes da equipe utilizam cartas para estimar o esforço em pontos das histórias, a numeração dessas cartas está distribuída em valores de 0, 1, 2, 3, 5, 8, 13, 20, 40 e 100 (COHN, 2006 pg. 56). Na COTIC são usadas cartas com numeração 1, 2, 3 e 5 para estimar. Quando todos os integrantes viram as cartas é alta a possibilidade de haver divergências entre os valores. Então, os integrantes que estimaram o maior e menor valores defendem os seus pontos de vista buscando convencer os demais integrantes a se juntar a eles na próxima jogada. Após essas discussões, as cartas são jogadas novamente até o consenso ser alcançado pela equipe.

Com as histórias estimadas é feito um somatório dos valores e esse resultado torna-se uma das metas da *Sprint*. A equipe tem o objetivo de alcançar o resultado dos somatórios terminando a programação da história.

Durante o *Planning* é decidido quem será o *Scrum Master* da *Sprint*. A prática de se escolher em cada *Sprint* um *Scrum Master* novo também é justificada devido a COTIC ser um ambiente de trabalho onde há muitos bolsistas. Verificou-se ao longo das iterações a necessidade de ter essa rotação no papel de *Scrum Master* possibilitando a cada membro praticar essa função. Na COTIC se faz presente apenas dois papéis no Scrum: o *Scrum Master* e a equipe, o papel de *product owner* não está presente no processo adotado.

As ***Daily Meetings*** na COTIC ocorrem duas vezes ao dia, ocorrendo uma no turno da manhã e outra no turno da tarde. Nelas, cada membro da equipe decide a história de usuário que ele vai trabalhar no turno. Também são verificadas as anotações deixadas no turno anterior à reunião atual. Essas anotações ficam armazenadas na ferramenta *online* Trello. As anotações são necessárias para a equipe verificar as pendências deixadas pelo turno anterior e os pontos positivos e negativos enfrentados no turno.

As ***Retrospective Meetings*** são realizadas no fechamento da iteração e geralmente são alocadas a cada sexta feira. A duração é a mesma do *Planning*. É desejável encontrar um horário viável para a maioria dos membros da equipe. No fechamento do ciclo é verificado se a meta estipulada foi alcançada, se as histórias de usuário que foram escolhidas estão prontas, e também é discutido entre a equipe os acontecimentos positivos de tudo que a equipe acertou na iteração e de tudo que

a equipe errou também. O resultado disso tem o potencial de gerar possíveis planos de ações para melhorar os acertos e corrigir os erros.

No que diz respeito às praticas da metodologia XP são inseridos no processo apenas algumas delas, as quais são discutidas a seguir.

A prática do **Código Coletivo** no processo consiste em todos os membros da equipe participarem em algum momento da história de usuário. Durante as fases do desenvolvimento o código é coletivo, ele não fica apenas na responsabilidade de uma pessoa. Ocorre na COTIC um revezamento até entre turnos. Por exemplo, se uma historia é alocada para ser codificada no turno da manhã por dois membros, no turno da tarde ela será continuada por outras duas pessoas. Nessa fase praticamente todos contribuem com o código de alguma forma.

Para ajudar a codificação ser coletiva foi definido um **Padrão**, aonde todo o código desenvolvido na coordenadoria tem que obrigatoriamente segui-lo. Pode ser citado como exemplo: os atributos obrigatoriamente são do tipo privado ou protegido e possuem métodos *get* e *set* associados.

Cada funcionalidade é uma história de usuário e cada história é testada para que o sistema seja entregue sem erros e defeitos. No processo da COTIC existe uma raia no Kanban chamada *System Test*, destinada a realizar testes automatizados. No momento apenas os testes de aceitação são realizados, e o objetivo da equipe é passar a realizar também testes de unidade. Para processar o teste de aceitação é usada a ferramenta Selenium para testar todo o sistema e verificar se há alguma parte das funcionalidades que não está funcionando da maneira esperada. É testado todo o sistema novamente toda vez que fica pronta uma nova funcionalidade.

Como há uma alta rotatividade no desenvolvimento do código a integração contínua busca minimizar problemas dá equipe com conflitos. No processo é usado o serviço *online* Gitlab para manter versões do código e ajudar a manter constante a integração do desenvolvimento. Atualmente todos os projetos desenvolvidos pela coordenadoria se encontram *online*. Usando o versionamento é possível manter as informações do andamento das histórias, verificar até onde o desenvolvedor conseguiu alcançar, e também é possível verificar onde ocorreram conflitos entre códigos e corrigir usando comparação. Para garantir que o código seja o mais atual,

todos os desenvolvedores são orientados que antes de começarem qualquer atividade, é preciso obter o código mais atual armazenado no Gitlab.

A prática ***Refactoring*** da XP é relacionada com uma área no Kanban dedicada denominada *Code Review*. É determinado internamente que quem está apto para realizar o *Code Review* é quem não participou do desenvolvimento ou participou pouco dessa fase da historia. A ideia é que a pessoa responsável pela revisão não deve ser influenciada e assim poder verificar: se o código é legível, se está duplicado, e se segue o padrão de código. Caso haja necessidade, o responsável pelo *Code Review* precisa apontar um possível caminho de ***Refactoring*** para os desenvolvedores melhorarem o código.

A programação em par ao longo do tempo foi sendo inserida na cultura da COTIC, e hoje no processo a maior parte de desenvolvimento é feito em pares. Os dois papeis da programação em par são assumidos por um período de tempo pelo par de programadores. Fica-se entre 10 a no máximo 20 minutos exercendo um papel e, ao fim desse tempo, o par troca entre si, e essa troca vai acontecendo em todo turno de trabalho que o par esta trabalhando na historia.

O Kanban da coordenadoria é dividido em três grandes áreas que são: o *To do* (faça), *Doing* (fazendo) e *Done* (feito). Dentro dessas áreas são subdivididas várias raias.

A área *To do* possui três raias ordenadas em *Requested, Analysis e Backlog*. Na primeira raia ficam todas as solicitações de funcionalidades que os *Stakeholders* fazem para o projeto. Cada nova funcionalidade é escrita em um *post-it* é colada no quadro. Após isso ela fica aguardando a equipe discuti-la na próxima raia. Na segunda raia é onde a equipe insere as histórias para ser analisadas e descreve as funcionalidades que serão programadas para ela. Quando a história sai dessa raia é inserida no *Backlog*. Nesse momento a história está pronta para começar, então cabe à equipe no *Planning* escolher quais serão implementadas na Sprint.

No *Doing* está presente a raia de *Dev* que corresponde o desenvolvimento das histórias de usuário. Ficam nessa raia apenas as histórias que estão no processo de programação. Na raia de *Text UX* ficam as histórias terminadas no DEV. Nela é feito o teste de usabilidade das funcionalidades. A pessoa que testa a história verifica se as respostas do sistema estão corretas de acordo com as especificações e também verifica se a interface está agradável e de fácil

entendimento para o usuário final. A próxima raia é a *Code Review*, já explicada anteriormente. Se a história passa na revisão de código ela fica em *Waiting* esperando o teste de aceitação ser preparado para só então passar para o *System Test*.

Na área de *Done* ficam as raias de *Artifact, Deploy e Delivered*. No *Artifact* é onde a história fica para ser feito o manual do usuário que será entregue para os *Stakeholders* quando o projeto for entregue. *Deploy* é onde o código desenvolvido e testado localmente é enfim enviado para produção. Após, a historia é passada para o *Delivered* que é a entrega para o usuário final e, por fim, o cartão é armazenado.

A prática na coordenação estabelece que o processo lide apenas com três histórias por vez estejam nas raias de *Dev, Text Ux e Code Review.* Uma quarta história só é aceita quando uma dessas três saem dessa área. A próxima wip é limitada a uma história e fica no *System Test.*

As prioridades são divididas em três níveis: alta, média e baixa, e são diferenciadas no Kanban por meio de cores colocadas no *post-it*: vermelho significa alta prioridade, o amarelo indica média e, o verde é associado com baixa prioridade.

3.2 Necessidades e Resultados Esperados da Medição na COTIC

Foram identificados no primeiro momento de composição deste trabalho as necessidades no ambiente de estudo e também quais expectativas de resultados o plano traria para a equipe, essas informações foram obtidas junto à equipe da COTIC por meio de reuniões.

A necessidade principal percebida é ter dados quantitativos na COTIC vinculados a um propósito. Foi relatado por parte da equipe que já havia uma coleta mínima de dados gerados durante a *Sprint*, porém estes dados apenas serviam para auxiliar o andamento da *Retrospective* da referida iteração sendo ao final da mesma descartados.

Não ter uma meta para os dados coletados, tinha como consequência fazer a equipe tomar decisões baseadas apenas em opiniões individuais sem embasamento no histórico do processo.

A coordenadoria espera como resultado da implementação das métricas ter seu próprio plano de medição focado em atender as necessidades do ambiente e

utilizá-lo para saber de que maneira otimizar o seu processo, sair do campo de incertezas e basear-se com dados reais para tomar decisões.

3.3 Projeto Acompanhado

O projeto escolhido para acompanhamento neste trabalho é o Sistema Integrado da Secretaria Executiva (SISE). O SISE é um sistema interno que atende a Secretaria Executiva da PROEG auxiliando-a a realizar os trabalhos do dia-a-dia do setor.

O sistema tem dois perfis de acesso: 1) o da secretaria, cujas funcionalidades principais são gerenciar funcionários e documentos internos e externos da Pró-Reitoria, e 2) o segundo perfil é o administrador do sistema. Esse último perfil é um usuário com acesso a todas as funcionalidades do SISE, podendo gerenciar todos os usuários do sistema.

A arquitetura utilizada é a *Model – view –controller* (MVC). A arquitetura MVC segundo Keenskang (1979) é dividida em camadas distintas cada uma com especificidades próprias em algum momento se interligam e convergem entre si.

De forma muito simplificada:

- *Model* é onde se encontra informações do usuário,
- *View* é a parte responsável por mostrar informações para o usuário e o
- *Controller* é o que controla as ações fazendo a interligação da *view* e *model*.

Para compor a parte técnica do SISE a linguagem de programação escolhida é o PHP. Para inserir o layout no sistema foi escolhido o *Framework* Materialize para inserir um *design* CSS padrão para interfaces do sistema, o SISE é o primeiro sistema desenvolvido pela equipe a fazer uso do Materialize.

O SISE para os padrões da COTIC não é um sistema complexo. É um projeto que contém dezesseis histórias de usuário e tem estimado duas datas para serem entregues as duas *Releases* definidas junto aos *Stakeholders* do SISE.

3.4 Plano de Medição

Com os objetivos da equipe definidos, o próximo passo é construir um plano de medição. O modelo de medição escolhido para a estruturação do plano foi o

modelo GQIM, por considerar a sua estrutura simples de ser entendida e viável para ser usada no ambiente.

O plano de medição deste trabalho segue o exemplo de outro plano de medição implantado em outra instituição usando o *template* adotado pelo Laboratório de Engenharia de Software da UFPA.

Baseando-se nessas informações foi elaborada a primeira versão de um plano de medição para ser experimentado no ambiente de trabalho e depois ao longo do planejamento foram sendo realizadas apresentações para a equipe do plano de medição e colhidos *feedbacks*. O plano de medição foi sendo construído até se encontrar hoje estruturado da seguinte forma:

1) Meta do negócio e objetivos da medição

O plano de medição desenvolvido para ser aplicado na COTIC evoluiu em três versões. Na primeira versão ele possuía para implementação da coleta de métricas duas metas:

1. Melhorar produtividade do desenvolvimento do produto, o objetivo da meta é analisar o esforço empregado nas fases do desenvolvimento do projeto.
2. Melhorar a qualidade dos produtos entregues ao cliente, o objetivo da meta é analisar a quantidade de erros do sistema desenvolvido.

Porém, na segunda versão do documento, verificou-se que não teria tempo hábil durante a realização deste trabalho para se desenvolver as duas metas. A decisão tomada no momento foi utilizar apenas a primeira meta de negócio, e reservar a outra para uso futuro.

2) Questões da medição

Tendo sido definido a meta e o objetivo da medição são formulados os questionamentos que respondem a meta, no plano de medição é escolhido três questões.

A primeira questão está focada em obter o quanto de esforço é empregado em uma história de usuário para o processo da COTIC, é escolhido essa questão por causa do interesse em saber em quanto tempo a coordenadoria pode se comprometer a entregar futuramente historias de usuário.

A segunda questão busca como resultado descobrir quanto tempo de desenvolvimento é gasto sendo refeito a funcionalidade já implantada do sistema,

quanto tempo se gasta fazendo modificações em historias já entregues as partes interessadas.

Por fim, a terceira questão se preocupa em saber em que parte do processo a equipe possui o menor ritmo de trabalho, ou seja, onde está se concentrando o maior despendimento de tempo no processo do setor. A Tabela 6 ilustra as questões do plano de medição

Tabela 6 – Plano de medição: questões

ID Questão	Questionamento
Q.1	Qual a distribuição de esforço pelas raias de desenvolvimento?
Q.2	Qual esforço é dispendido em retrabalho?
Q.3	Onde ocorre maior entrave no processo?

3) Indicadores

Há três indicadores e para cada questionamento há um indicador ligado a ele diretamente.

O primeiro indicador busca esclarecer o tempo que é gasto no desenvolvimento e agregar com o tempo que é gasto com realização de testes, para responder o questionamento do tempo de trabalho no processo.

O segundo indicador descreve o esforço que é dispendido em retrabalho. Esse retrabalho é o tempo que a equipe leva para refazer a história que apresentou não conformidades após ser entregue ao cliente final.

O terceiro indicador descreve o percentual de distribuição do esforço pelas raias da área *Doing* do processo. Ele é inserido para indicar onde o processo se encontra maior tempo de trabalho.

Os indicadores tem um formato de gráfico de barra para ilustrar as respostas as perguntas. A Tabela 7 ilustra os indicadores ligados a sua respectiva questão

Tabela 7 – Plano de medição: questões e indicadores

ID Questão	ID indicador	Indicador
Q.1	I.1	Esforço que leva para desenvolver e testar uma US
Q.2	I.2	Índice de tempo gasto em retrabalho
Q.3	I.3	Distribuição de esforço nas raias do processo

4) Métricas

A Tabela 8 descreve as métricas que compõem o plano de medição com seus respectivos indicadores e suas descrições.

Tabela 8 – Plano de medição: Métricas

ID indicador	ID métrica	Descrição
I.1	M.1	Tamanho funcional da US
I.1	M.2	Quantidade de Homens Hora
I.1	M.3	Total de esforço no processo
I.2	M.4	Esforço em manutenção
I.2	M.5	Esforço total do projeto (até a primeira *release*)
I.2	M.6	Total de tempo empregado em retrabalho
I.3	M.7	Quantidade de tempo da US na raia DEV
I.3	M.8	Quantidade de tempo da US na raia UX
I.3	M.9	Quantidade de tempo da US na raia CODE REVIEW
I.3	M.10	Quantidade de tempo da US na raia SYSTEM TEST
I.3	M.11	Taxa de distribuição de esforço na raia Dev
I.3	M.12	Taxa de distribuição de esforço na raia UX
I.3	M.13	Taxa de distribuição de esforço na raia CODE REVIEW
I.3	M.14	Taxa de distribuição de esforço na raia SYSTEM TEST

Kitchenham et al (1995) caracterizam os elementos que compõem a estrutura de medição. São eles:

A **Entidade** é descrita como sendo tudo aquilo que se pode considerar um objeto no mundo real. No âmbito de medição de software as entidades podem ser de processos, produtos e recursos.

Os **Atributos** estão relacionados as entidades, pois eles representam as propriedades da entidade. A **Unidade** é o que indica como medir o atributo que ela está relacionada.

O **Tipo de Escala** é determinado para corresponder ao tipo de Unidade escolhido. Podem ser: Nominal, Ordinal, Intervalar e Racional.

O **Valor** é aplicado quando se mede um atributo, com uma unidade específica para obtenção de um Valor.

Os **Instrumentos de Medição** são utilizados para obter a medição do valor de um determinado atributo. A medição do atributo é dividida em direta, chamadas também de Básicas, e indiretas que também podem ser chamadas de Derivadas. A medição Básica obtém apenas os valores de um determinado atributo, a medição Derivada depende dos valores obtidos na medição das Básicas, para só então enfim ser calculado o valor da Derivada.

Na Tabela 9 é descrito para cada métrica o seu tipo, a fórmula de cálculo a ser aplicada e a sua unidade de medida. Como o tipo de escala é o Racional para todas as métricas a informação foi omitida da Tabela 9.

Tabela 9 – Elementos de Medição

ID de métrica	Tipo de métrica	Fórmula de cálculo	Unidade
M.1	Básica	-	PF (Pontos de função)
M.2	Básica	-	horas(h)
M.3	Derivada	M.3 = M.2/M.1	Horas/PF
M.4	Básica	-	horas(h)
M.5	Básica	-	horas(h)
M.6	Derivada	M.6 = M.4/M.5	horas(h)
M.7	Básica	-	horas(h)
M.8	Básica	-	horas(h)
M.9	Básica	-	horas(h)
M.10	Básica	-	horas(h)
M.11	Derivada	M.11 = M.7/M.5	%
M.12	Derivada	M.12 = M.8/M.5	%
M.13	Derivada	M13 = M9/M5	%
M.14	Derivada	M14 = M10/M5	%

3.5 Coleta de Dados

O período de realização da coleta de métricas na COTIC teve início no dia 30 de janeiro de 2017, com a primeira iteração envolvendo histórias de usuário do projeto SISE. O término do período de coleta é no dia 13 de abril de 2017, antes do término da primeira *release* do SISE agendada para o dia 17 do mesmo mês.

A coleta de dados abrange cinco *Sprints* compostas por dez dias cada. Ao todo são quarenta e nove dias úteis de trabalho. Durante esses quarenta e nove dias são acompanhadas sete histórias de usuário do projeto SISE com o plano de medição.

Os dados coletados estão armazenados em planilha eletrônica, onde todos os membros da equipe têm acesso e podem acompanhar os trabalhos. Entretanto a inserção de novos dados é sempre feita pelo *Scrum Master*.

Nessa planilha eletrônica o *Scrum Master* ao final do turno de trabalho insere a quantidade de horas trabalhadas da história informadas pelo membro que estava trabalhando nela. É separado os valores por dia para cada raia e ao final da Sprint é feito o somatório para totalizar as horas trabalhadas em cada raia.

A periodicidade da coleta de dados para cada uma das métrica está descrita na Tabela 10.

Tabela 10 – periodicidade da coleta de dados

ID métrica	Período
M.1	Na fase de planejamento do projeto
M.2, M.3	Fase final do desenvolvimento do projeto
M.4, M.6	Após da entrega do sistema, nas próximas iterações.
M.5	No final da fase de planejamento da *release*
M.7, M.8, M.9, M.10	No final de cada atividade
M.11, M.12, M.13, M.14	No final de cada *Sprint*.

3.6 Resultados e Análise da Coleta de Dados

1) Indicador Um

Durante o período de concepção do plano de medição foi verificado que as histórias de usuário dos projetos desenvolvidos pela COTIC não seguiam um padrão de tamanho, não se tinha nenhuma contagem de linhas de código. Em outras palavras, não se sabia se o tamanho de cada história era homogêneo da COTIC. Essa característica gerou um problema no indicador I.1, pois tornou-se interessante para a medição saber o tamanho de uma história, para assim a equipe ter a consciência do esforço empregado em seu processo para cada tamanho de história.

Para obter dados para o indicador I.1 no primeiro pensou-se em utilizar o LOC, porém a COTIC não tinha uma normalização em seus projetos. Então tomou-se a decisão de se medir o tamanho funcional das histórias, usando o cálculo de Ponto de Função. O resultado da métrica M.1 está no Gráfico 1, o mesmo mostra a discrepância no tamanho das histórias do SISE.

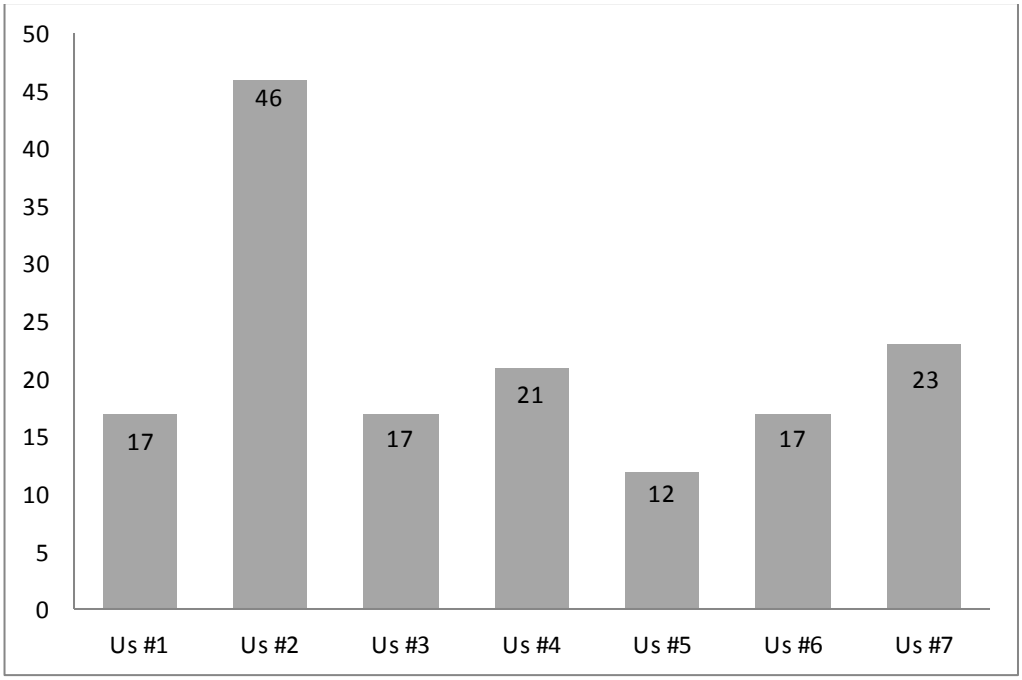

Gráfico 1 – Tamanho funcional das histórias de usuário

O resultado da métrica M.2 está ilustrado no Gráfico 2. Para obter esses resultados de Homem Hora da M.2 foram coletados o tempo de trabalho empregado nas raias do processo.

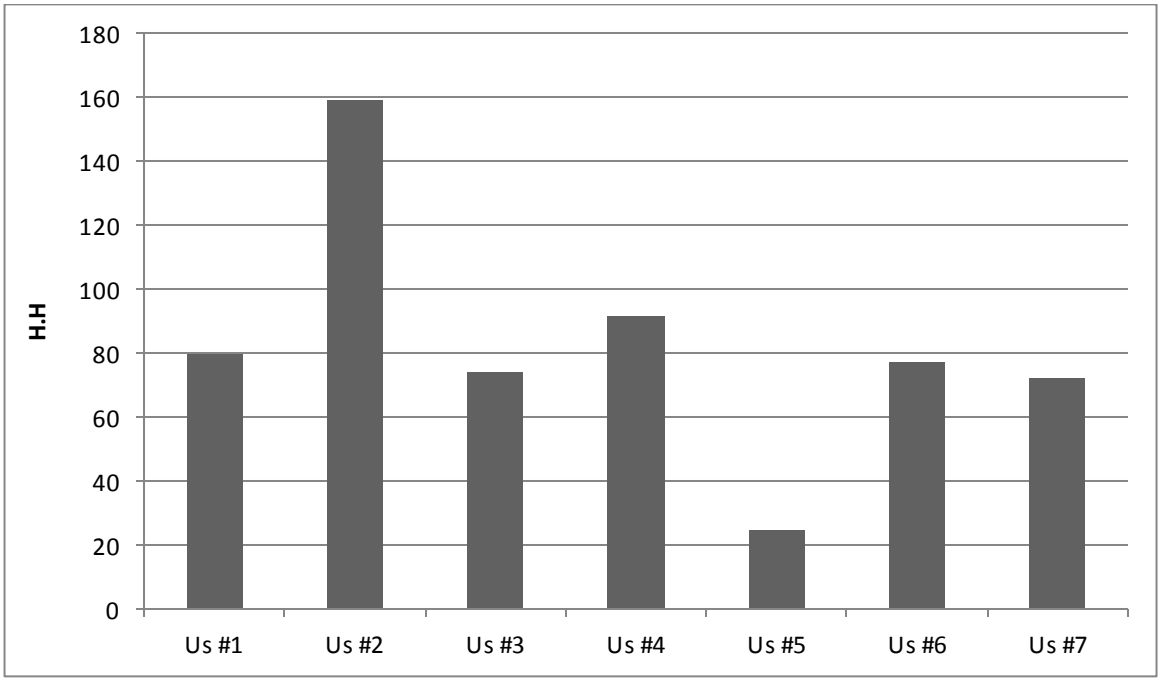

Gráfico 2 – Quantidade de homem hora nas raias do processo

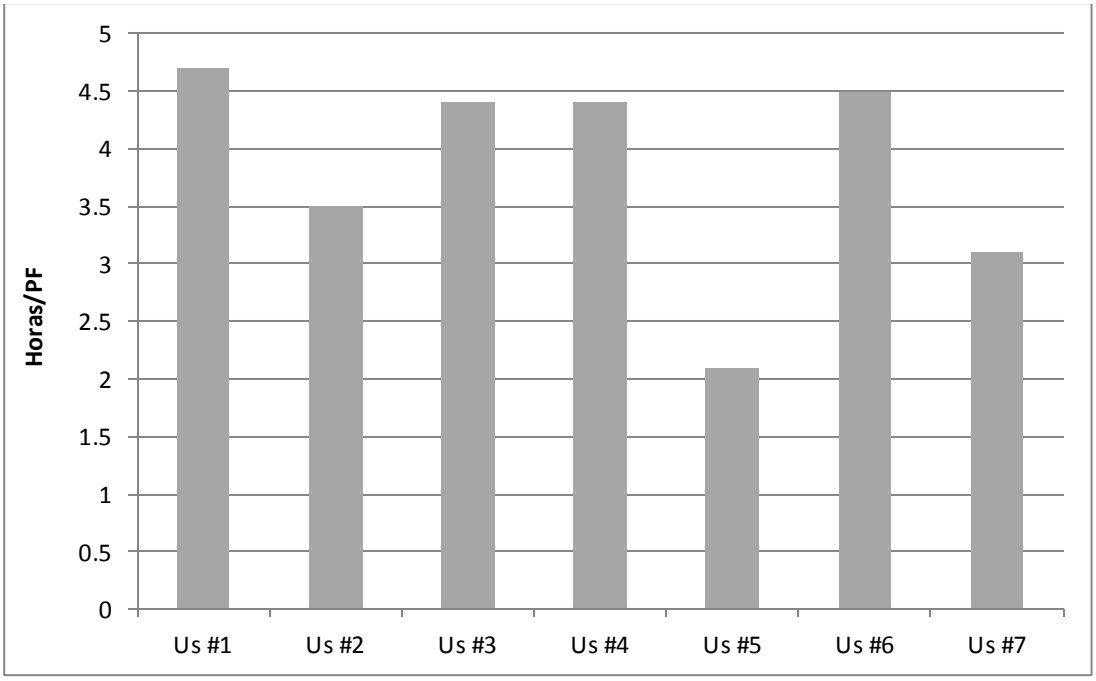

Gráfico 3 – Resultado de Produtividade no Processo

É possível visualizar no Gráfico 3 que as histórias de usuário 1, 3, 4 e 6 tem praticamente a mesma produtividade, pode ser observado que essas histórias tem em comum seu valor de tamanho funcional, sendo a história 1, 3 e 6 de tamanho funcional 17 e a história 4 de tamanho 21 só um pouco maior do que as outras.

2) Indicador Dois

Os resultados das métricas ligadas ao indicador dois até o final do acompanhamento deste trabalho não puderam ser contabilizados, pelo fato da equipe não ter conseguido entregar no tempo previsto as historias para a primeira *release.*

3) Indicador Três

A **primeira iteração** é composta por duas histórias de usuário do SISE. O Gráfico 4 ilustra os resultados das métricas M.7, M.8, M.9 e M.10 obtidos na execução da primeira iteração. O M.5 corresponde ao valor de 392 horas até a primeira *Release.*

A primeira história de usuário utilizou todos os dez dias da iteração. A segunda história, porém não conseguiu ser finalizada na *Sprint* e nem ao menos sair da área de desenvolvimento ao final da mesma iteração, os resultados podem ser observados no Gráfico 4. Como a história 2 não se concluiu nesta iteração, ela continua a ser desenvolvida e medida na próxima *Sprint.*

É observado nessa Sprint pela equipe uma maior dificuldade em codificar essas primeiras histórias pelo fato de se usar o *framework* Materialize, no qual a equipe não possuía grande conhecimento do mesmo.

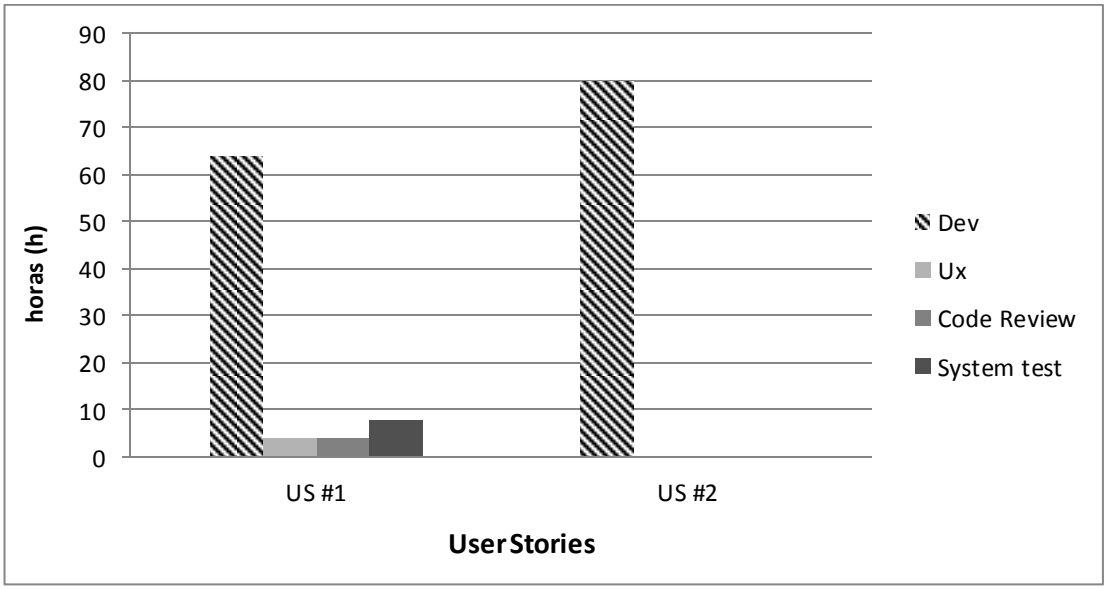

Gráfico 4 – Resultado da medição na 1º Iteração

Após o fim da iteração são computados, enfim, as métricas M.11, M.12, M.13 e M.14 para a primeira iteração.

Os resultados obtidos são mostrados no Gráfico 5. Entretanto, foi subtraído o resultado da história de usuário dois, pois a mesma não se encerra nessa *Sprint*.

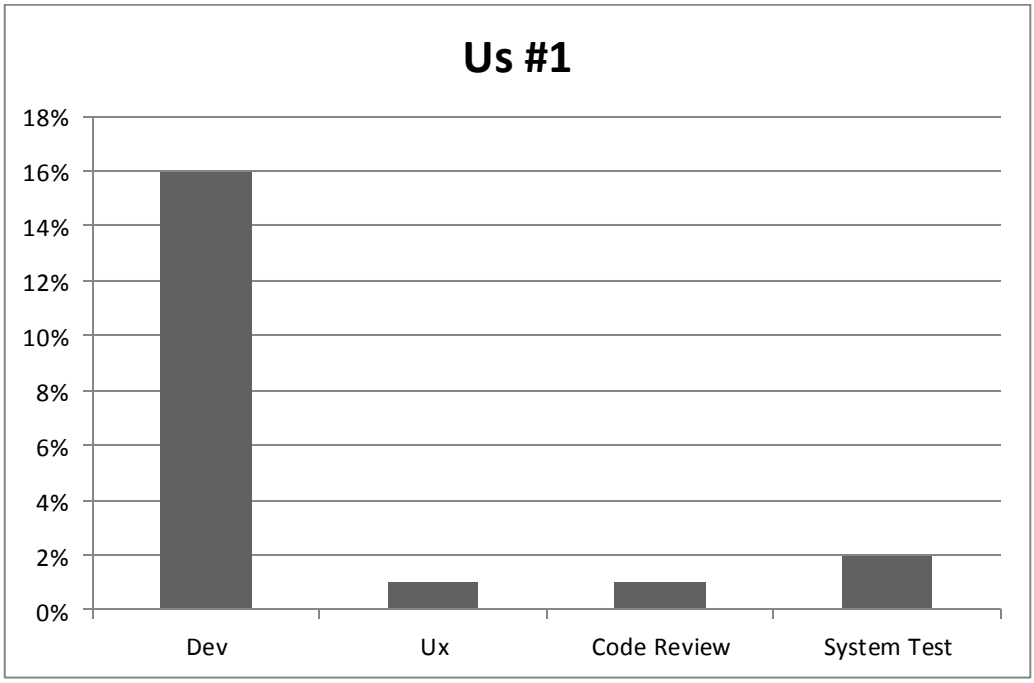

Gráfico 5 – Resultado das taxas de distribuição nas raias da 1º Iteração

Na **segunda iteração** é acompanhado novamente duas historias de usuário. Entretanto, como a história de usuário dois não foi finalizada na iteração anterior, a mesma foi inserida nessa segunda iteração. Como consequência apenas uma historia nova foi inserida ao processo. Os resultados da coleta são apresentados no Gráfico 6.

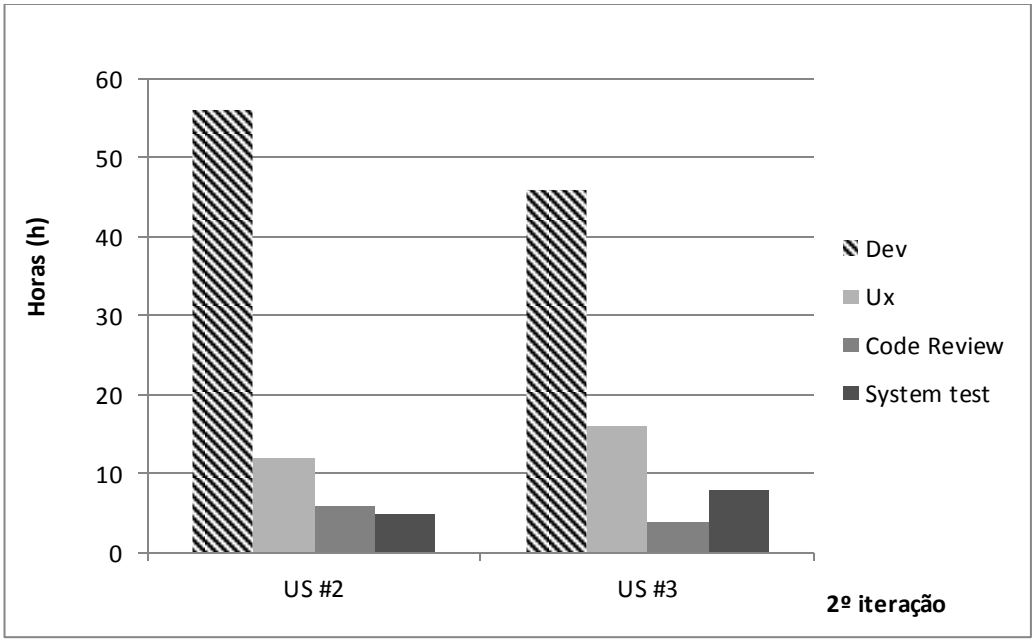

Gráfico 6 – Resultado da medição 2º Iteração

O resultado da métrica M.11 da historia dois foi gerado, sendo somado o resultado da métrica M.7 das duas iterações.

Em comparação com a iteração anterior houve um crescimento do tempo da história na raia Ux.

Em reunião com a equipe para apresentar os resultados da iteração, houve a detecção do possível motivo do crescimento de esforço na raia Ux. Havia uma falha de continuidade em realizar os testes de aceitação em turnos diferentes, pois caso o teste de aceitação realizado por um membro do time não terminasse em seu turno de trabalho, o mesmo no próximo turno, não era continuado porque as informações do que foi testado não estava sendo repassadas.

A decisão tomada pela equipe foi ter o teste de aceitação descrito com as funcionalidades a serem testadas na ferramenta *Trello*. É mantido pelo membro da equipe o teste de aceitação, contendo uma lista do que foi testado e o que falta a ser

testado, para assim ser possível a continuidade do trabalho. O Gráfico 7 ilustra os resultados percentuais em cada raia

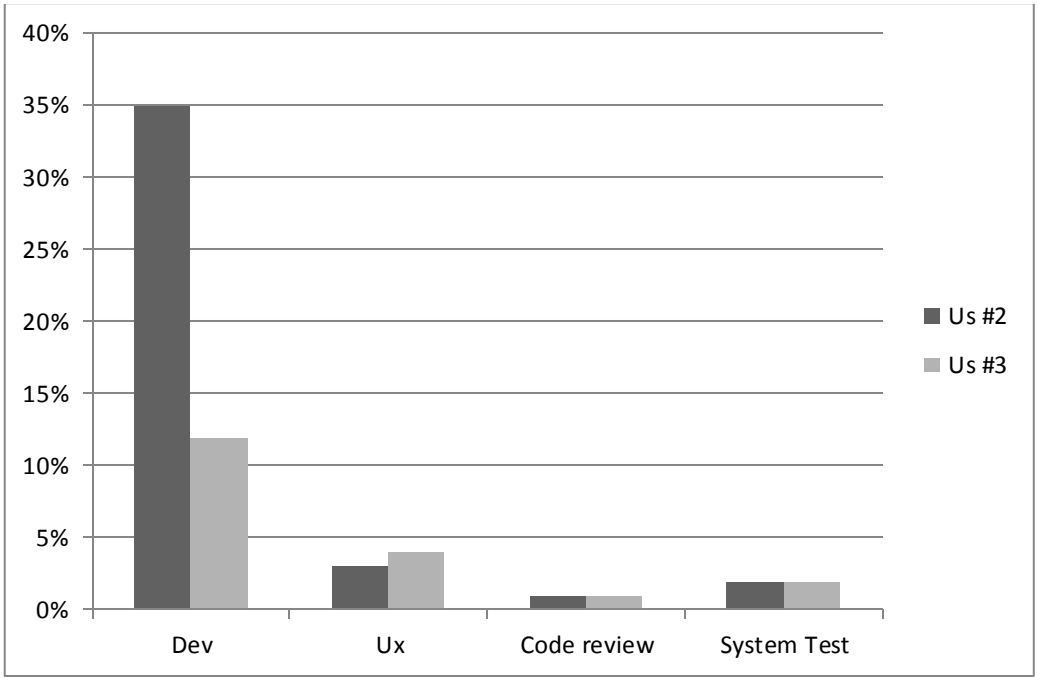

Gráfico 7 – Resultado das taxas de distribuição nas raias da 2º Iteração

Os resultados da **Terceira Iteração** estão ilustrados no Gráfico 8. Novamente houve a ocorrência de uma história não ser entregue ao final da iteração. No caso em questão foi a história quatro que terminou a *Sprint* na raia de *Ux*.

A história cinco por ser a menor em tamanho funcional e mais simples não apresentou problemas para ser entregue.

Nessa Sprint a equipe esteve por ordens superiores da Pró-Reitoria focada em outras demandas do setor. Isso refletiu no desenvolvimento das historias da iteração.

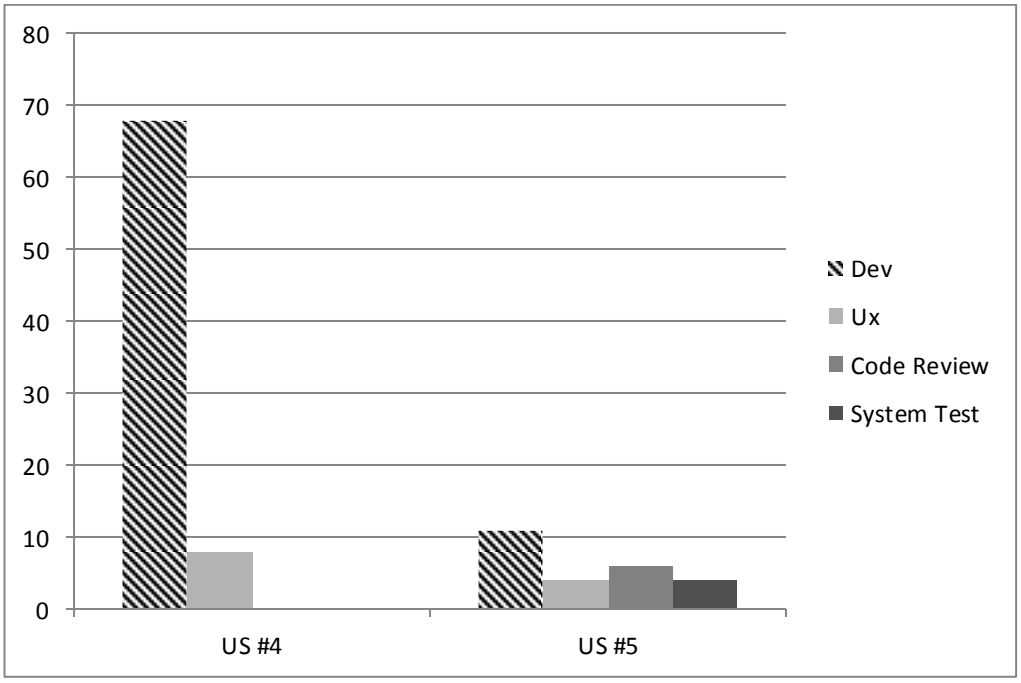

Gráfico 8 – Resultado da medição 3º Iteração

O resultado apresentado no Gráfico 9 é apenas para a história cinco, o resultado da história quatro é computado na próxima iteração.

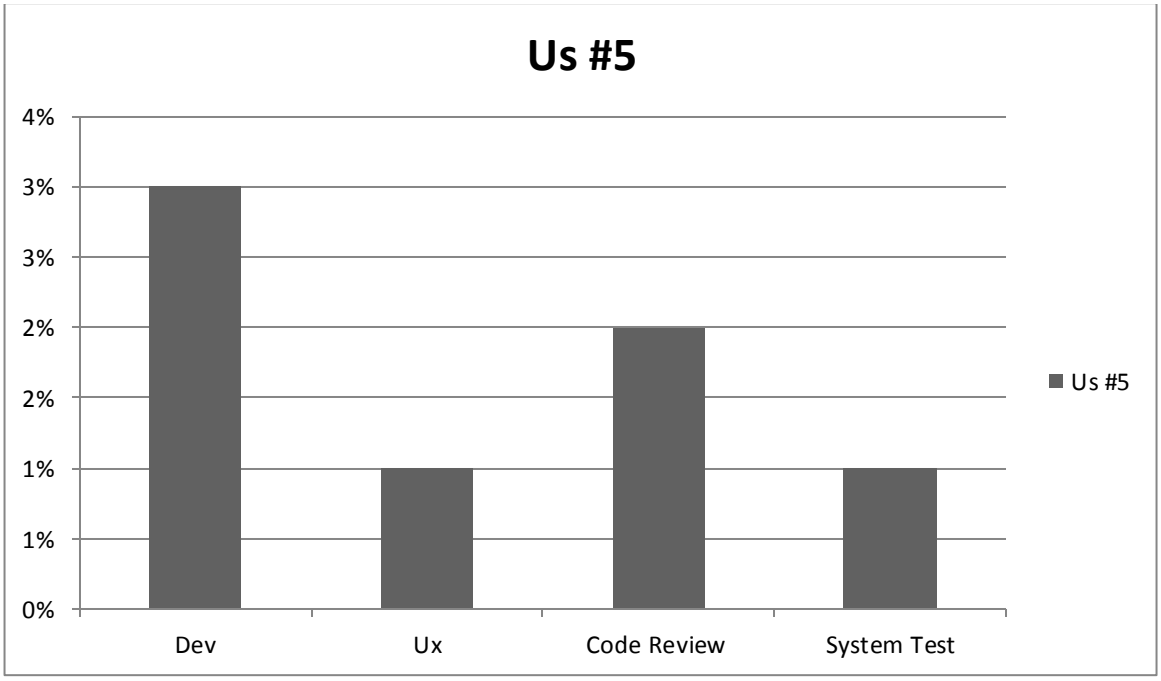

Gráfico 9 – Resultado das taxas de distribuição nas raias da 3º Iteração

Na **iteração quatro** a história quatro é finalizada sendo realizado seu *Code Review* e *System Test*. A historia seis foi finalizada na iteração e é possível perceber um aumento novamente do tempo dedicado a raia *Ux*, nessa historia ocorreu o que a mesma retornou duas vezes para a raia Dev porque as funcionalidades apresentavam não conformidades no teste de aceitação, ou seja houveram erros lógicos de programação. Os resultados do emprego de horas nas raias são ilustrados no Gráfico 10 e no Gráfico 11 os percentuais.

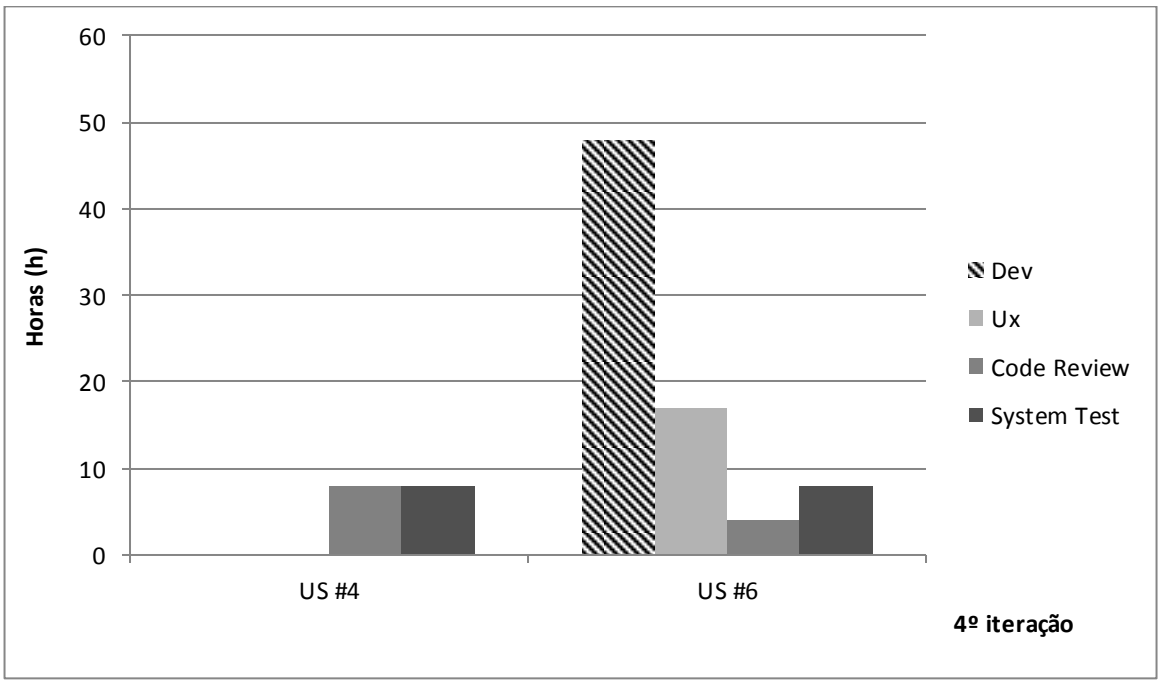

Gráfico 10 – Resultado da medição da 4º Iteração

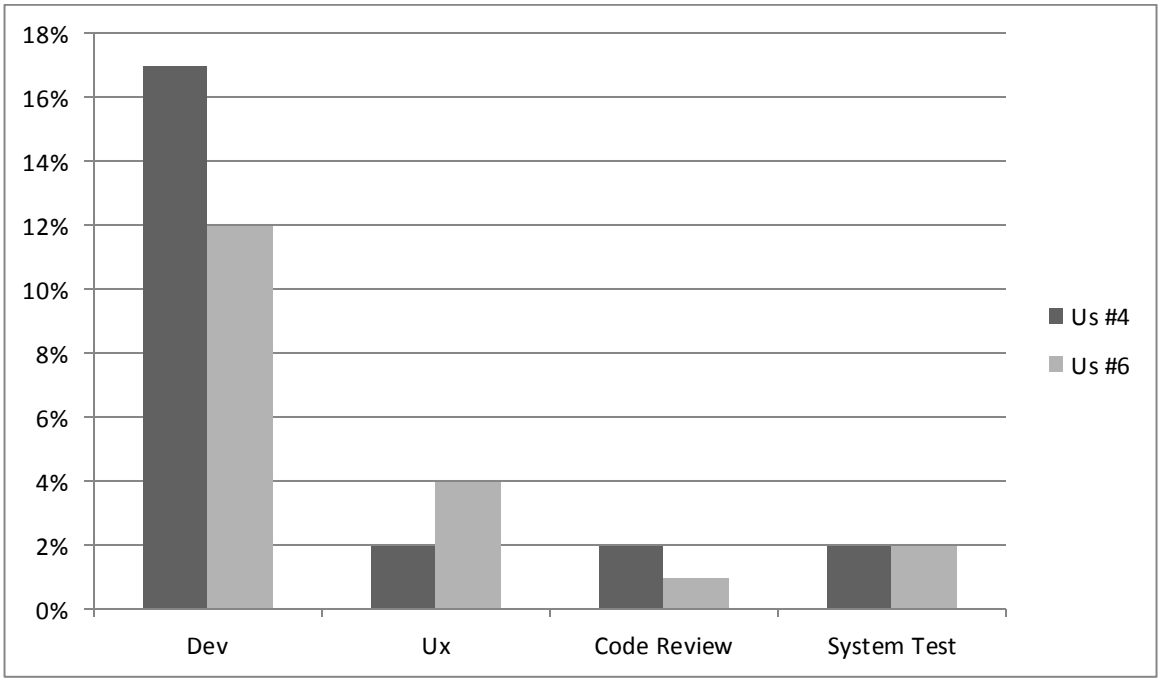

Gráfico 11 – Resultado das taxas de distribuição nas raias da 4º Iteração

A quinta e última iteração teve, diferentemente das anteriores, apenas nove dias úteis de trabalho. A equipe não conseguiu terminar a tempo a história 7, a mesma ainda permanece na raia *Code Review* na próxima iteração, como ilustrado no Gráfico 12 e no Gráfico 13 é mostrado os percentuais.

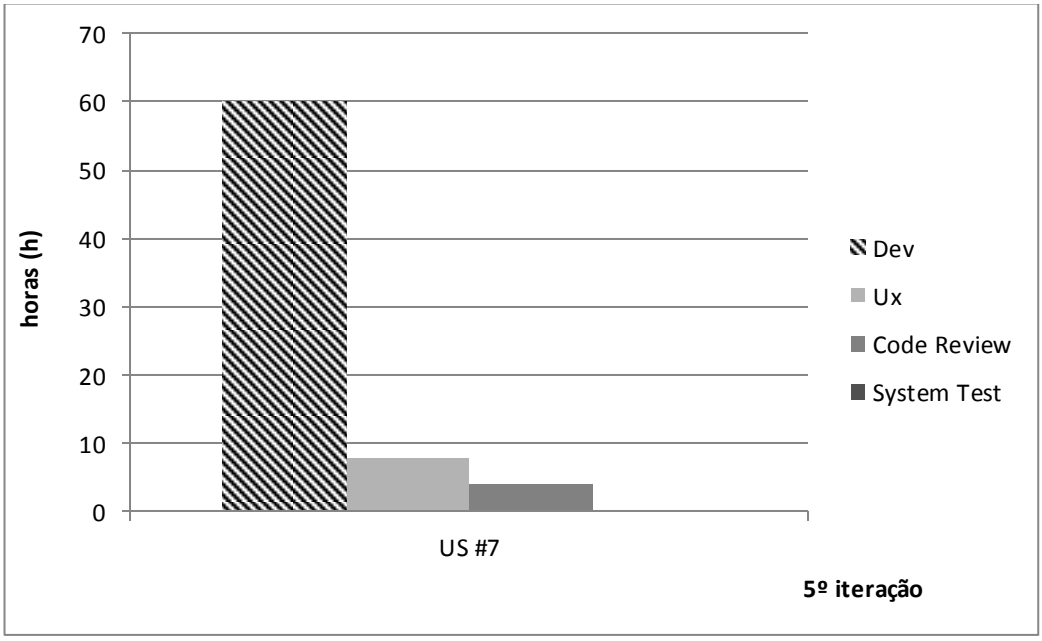

Gráfico 12 – Resultado da medição da 5º Iteração

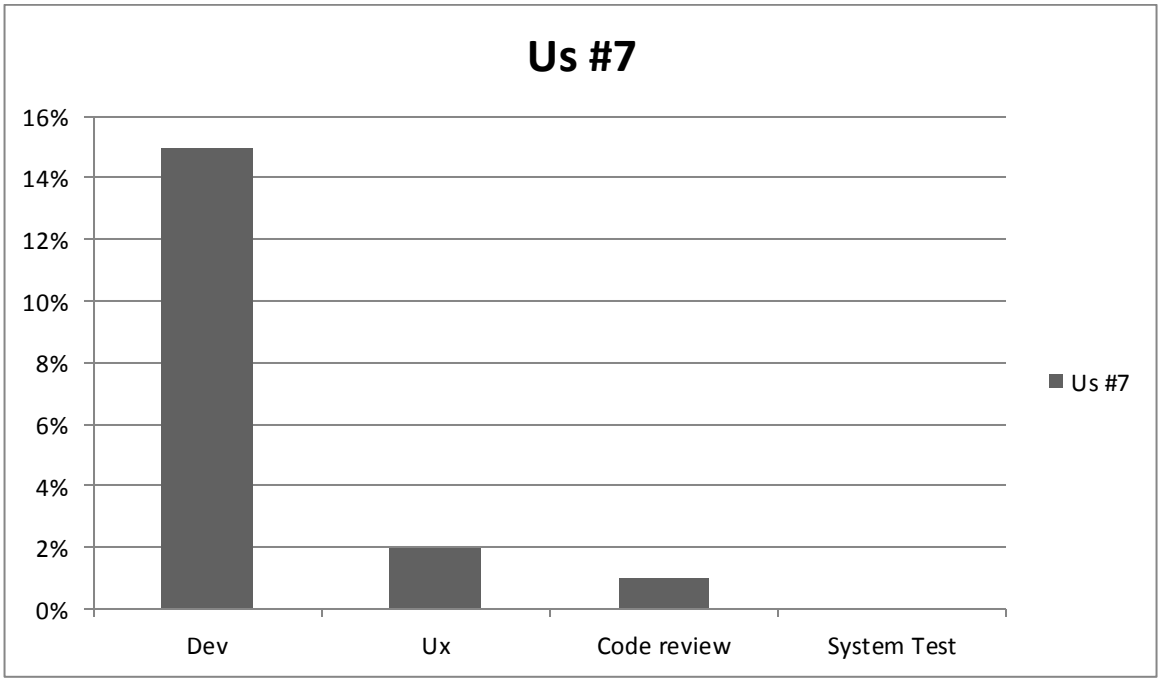

Gráfico 13 – Resultado das taxas de distribuição nas raias da 5º Iteração

Com o fim das iterações foi possui tirar algumas informações sobre o I.3. A raia Dev é a que possui a maior taxa de esforço entre as raias, a raia Ux apresenta maiores oscilações entre as iterações. As raias de *Code Review* e *System Test* se mantêm estáveis em praticamente todas as iterações, algo que não era esperado pela equipe.

Há a ocorrência no setor de falhas nas entregas das histórias. Houve a ocorrência de três casos onde elas não conseguiram ser entregues no prazo determinado, causando atrasos no cronograma, prejudicando a entrega do sistema.

4 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as considerações finais sobre as lições aprendidas e resultados obtidos com o uso de medição de software na COTIC.

No decorrer da execução deste trabalho é constatado o quanto compor e inserir o processo de medição em uma organização não é uma tarefa trivial. Principalmente em uma organização que ainda não possui o costume de usar a medição.

É um processo longo decidir as metas de negócio, as questões, os indicadores e por fim as métricas. E para ser capaz de determinar todos esses elementos é preciso parar e refletir sobre todo o processo, isso gera como resultado um conhecimento maior do trabalho realizado pela equipe.

Este trabalho alcançou os objetivos iniciais buscados, construiu e implantou no setor da COTIC um plano de medição para realizar o acompanhamento das métricas. Por fim a equipe começa a manter um histórico de dados capaz de auxiliar a tomada de decisões futuras.

5 TRABALHOS FUTUROS

Uma das atividades futuras é fazer uso da meta de negócio subtraída neste trabalho, com o objetivo de poder acompanhar a produtividade e a qualidade do software entregue.

É esperado também com uma das atividades futuras evoluir o Plano de medição, através do acrescimento de um indicador focado em medir o esforço em atividades de atendimento ao usuário. Isto é motivado pelo fato que a COTIC presta suporte técnico em todos os seus sistemas e isto certamente possui impacto em seu trabalho de desenvolvimento de histórias.

É desejada a realização do acompanhamento dos resultados do Plano de Medição, quando houver mudanças bruscas de membros da equipe, com a finalidade de se observar quais os impactos da rotatividade dos membros bolsistas no ambiente de trabalho.

Por fim, vislumbra-se a possibilidade de se implantar uma ferramenta propícia a armazenar os dados coletados no processo da COTIC, substituindo o uso de planilhas para a armazenagem.

REFERÊNCIAS

ALBRECHT, Alan. **Measuring Application Development productivity**. Share/Guide IBM ApplicationDevelopmentSymposium, 1979.

ALBRECHT, Alan; GAFFNEY, John. “Software Function, Lines of Code and Development Effort Prediction: a Software Science Validation”. IEEE Transactions Of Software Engineering, 1983.

ANDERSON, David. **Kanban:** mudança evolucionária de sucesso para seu negócio de tecnologia. Sequim: Blue Hole Press, 2011.

ANÁLISE DE RISCOS PELO USO DE MÉTODOS ÁGEIS NA GESTÃO DE PROJETOS DE DESENVOLVIMENTO DE SOFTWARE. Revista de Gestão e projetos: GEP, 2014.

BARVINSKI, Carla Adriana. CAGNIN, Maria Istela. LIMA FILHO, Nilson E. S. VERONEZI, Leandro. ***XP Tracking Tool****:* Uma Ferramenta de Acompanhamento de Projetos Ágeis. **In:** Anais do AgileBrazil 2010- Conferência Brasileira sobre Métodos Ágeis de Desenvolvimento de Software.

BASILI, Victor; CALDIERA, Gianluigi; ROMBACK, H. Dieter. **Goal Question Metric Paradigm**. In: Encyclopediaof Software Engineering. [s.l.]: John Wiley& Sons, Inc., 1994.

BECK, Kent; Andres, Cynthia. **Extreme programming Explained**: Embrace chance, 2 edn. Addison-Wesley, 2004.

BECK, Kent. **Smalltalk Best Practice Patterns**. Upper Saddle River: Prentice Hall, 1996.

BORGE, Pereira Eduardo. Um Modelo de Medição para Processos de Desenvolvimentos de Software 2003. Tese em Mestrado – Universidade Federal de Minas Gerais.

COHN, Mike. **Agile Estimating and Planning**. Upper Saddle River: Prentice Hall, 2005.

FOWLER, Martin. **Refactoring Improvingthe design of existing code**. Addison-Wesley. 1999.

GitLab inc. GitLab. Disponível na internet em: http://www.gitlab.com. Acesso em: 10 de Abril de 2017.

Kitchenham., B.; Pfleeger, S. L.; Fenton, N. Towards a Framework for Software

Measurement Validation. IEEE Transactions on Software Engineering, vol. 21, nº 12. December, 1995.
International Function Point Users Group. Manual de Práticas de Contagem de Pontos de Função, Versão 4.3.1. 2010.

GUARIZZO, Karina. Métricas de Software. 2008. Tese de Mestrado – Faculdade de Jaguariúna.

KEENSKAUG, Trygve. **Working With Objects the Optam Software Engineering Method**. Manning, 1995.

MEIRELLES, Paulo Roberto Miranda. Monitoramento de Métricas de Código-Fonte em Projetos de Software Livre 2013. Tese de Mestrado - Universidade de São Paulo.

MetricsViews. Vol5. Allan J. Albrecht Father of Function Points. Disponível em: <http://www.ifpug.org/metricviews/>. Acessado em: 16/03/2017.

MPS.BR - Melhoria de Processo do Software Brasileiro: Guia Geral MPS de Software. SOFTEX: Rio de Janeiro, 2016.

PARK, Robert; GOETHERT, Wolfhart; FLORAC, William**. Goal-Driven Software Measurement-AGuidebook.** Pittsburgh: Carnegie MellonUniversity, 1996.

PRESSMAN, Roger; MAXIM, Bruce. Engenharia de Software: Uma Abordagem Profissional. 8ª edição. Porto Alegre: MCGraw-Hill , 2016.

Selenium inc. Selenium. Disponível na internet em: http//www.seleniumhq.org. Acesso em: 10 de Abril de 2017.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª edição. São Paulo: Pearson, 2011.

SUTHERLAND, Jeff. **Scrum: a arte de fazer o dobro do trabalho na metade do tempo**. São Paulo: LEYA, 2014.

SUTHERLAND, Jeff; SCHWABER, Ken**. Guia do Scrum. Um guia definitivo para o Scrum: As regras do jogo**. Disponível em:<http://www.scrumguides.org/download.html>. Acessado em: 05/02/2017.

Trello inc. Trello. Disponível na internet em: http://www.trello.com. Acesso em: 10 de Abril de 2017.

APÊNDICE A – PLANO DE MEDIÇÃO

Versões e revisões deste documento				
Data	**Comentário**	**Versão**	**Autor**	**Revisor**
01/10/2016	criação do documento	1.0	Franciellem Bezerra	Rodrigo Quites
20/1/2017	é priorizado a meta de negócio MM2. reorganização dos indicadores e métricas.	2.0	Franciellem Bezerra	Rodrigo Quites
16/02/2017	modificação do indicador I3	3.0	Franciellem Bezerra	Rodrigo Quites
05/04/2017	reestruturação das métricas ligadas I3	4.0	Franciellem Bezerra B.	Rodrigo Quites

Necessidades de Informação e objetivos de Medição			
Prioridade da Necessidade de Informação	**Meta de Negócio/Necessidade de Informação**	**ID Meta de Medição**	**Objetivos de Medição**
1	Melhorar a qualidade dos produtos entregues ao cliente	MM1	analisar a quantidade de erros do sistema desenvolvido
1	Melhorar produtividade do desenvolvimento do produto	MM2	analisar o esforço empregado nas fases do desenvolvimento do projeto

Metas de medição e questões		
Id meta de medição	**Id questão**	**Questão**
MM1	Q1	Qual o grau de correção do software?

MM1	Q2	Qual o grau de erros e defeitos encontrados no produto?
MM2	Q3	Qual a distribuição de esforço pelas raias de desenvolvimento?
MM2	Q4	Qual esforço é dispendido em retrabalho?
MM2	Q5	Onde ocorre maior gargalo no processo?

Questões e indicadores				
Id questão	**Id indicador**	**Indicador**	**Formato**	**Entidade**
Q1	I1	taxa de acerto entre o que foi solicitado pelo cliente e o que a equipe entregou	BARRA VERTICAL	ATIVIDADE
Q2	I2	taxa de defeitos encontrado pelo usuário final	BARRA VERTICAL	PROJETO
Q3	I3	esforço que leva para desenvolver e testar uma US	BARRA VERTICAL	ATIVIDADE
Q4	I4	índice de tempo gasto em retrabalho	BARRA VERTICAL	ATIVIDADE
Q5	I5	distribuição de esforço nas raias do processo	PIZZA	ATIVIDADE

Indicadores e Métricas						
Id indicador	**Id métrica**	**Descrição da Métrica**	**Tipo de Métrica**	**Fórmula de Cálculo**	**Tipo de Escala**	**UNIDADE**
I1	M1	Quantidade de não conformidades encontradas	Básica	N/A	racional	Artefato
I1	M2	Quantidade total de Us's	Básica	N/A	racional	UNIDADE
I1	M3	Taxa de acertos da	Derivada	M3 = M1/M2	racional	UNIDADE

		equipe				
I2	M4	Quantidades de defeitos	Básica	N/A	racional	UNIDADE
I2	M5	Número de linhas de código	Básica	N/A	racional	TURNOS
I2	M6	Densidade de defeitos	Derivada	M6 = M4/M5	racional	TURNOS
I3	M7	Tamanho funcional da US	Básica	N/A	racional	PF
I3	M8	Quantidade de HH	Básica	N/A	racional	Horas(h)
I3	M9	Total de esforço no processo	Derivada	M9=M8/M7	racional	Horas/PF
I4	M10	Esforço em manutenção	Básica	N/A	racional	horas(h)
I4	M11	Esforço total do projeto	Básica	N/A	racional	horas(h)
I4	M12	Total de tempo empregado em retrabalho	Derivada	M12 = M10/M11	racional	horas(h)
I5	M13	Quantidade de tempo da us na raia DEV	Básica	N/A	racional	horas(h)
I5	M14	Quantidade de tempo da us na raia UX	Básica	N/A	racional	horas(h)
I5	M15	Quantidade de tempo da us na raia CODE REVIEW	Básica	N/A	racional	horas(h)

I5	M16	Quantidade de tempo da us na raia TEST SYSTEM	Básica	N/A	racional	horas(h)
I5	M17	Taxa de distribuição de esforço na raia Dev	Derivada	M17= M13/M11	racional	%
I5	M18	Taxa de distribuição de esforço na raia UX	Derivada	M18 = M14/M11	racional	%
I5	M19	Taxa de distribuição de esforço na raia CODE REVIEW	Derivada	M19 = M19/M11	racional	%
I5	M20	Taxa de distribuição de esforço na raia TEST SYSTEM	Derivada	M20 = M16/M11	racional	%

Id Métrica	**Periodicidade/Frequência**	**Responsável**	**Ferramentas**
M1		Scrum Master	Planilha Excel
M2		Scrum Master	Planilha Excel
M3		Scrum Master	Planilha Excel
M4		Scrum Master	Planilha Excel
M5		Scrum Master	Planilha Excel
M6		Scrum Master	Planilha Excel
M7		Scrum Master	Planilha Excel

M8		Scrum Master	Planilha Excel
M9		Scrum Master	Planilha Excel
M10	No final da fase da primeira release	Scrum Master	Planilha Excel
M11	No final da fase de planejamento da release	Scrum Master	Planilha Excel
M12	No final da fase da primeira release	Scrum Master	Planilha Excel
M13	No final de cada atividade	Scrum Master	Planilha Excel
M14	No final de cada atividade	Scrum Master	Planilha Excel
M15	No final de cada atividade	Scrum Master	Planilha Excel
M16	No final de cada atividade	Scrum Master	Planilha Excel
M17	No final de cada iteração	Scrum Master	Planilha Excel
M18	No final de cada iteração	Scrum Master	Planilha Excel
M19	No final de cada iteração	Scrum Master	Planilha Excel
M20	No final de cada iteração	Scrum Master	Planilha Excel