UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

MARIANA APARECIDA DE MATTOS

ESTUDO DE CASO COM USE CASE 2.0

Florianópolis,
2018/1

Mariana Aparecida de Mattos

ESTUDO DE CASO COM USE CASE 2.0

Trabalho de Conclusão de Curso apresentado ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientador: Prof. Dr. Jean Carlo Rossa Hauck

Florianópolis,

2018/1

Mariana Aparecida de Mattos

ESTUDO DE CASO COM USE CASE 2.0

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação

Orientador:

Prof. Dr. Jean Carlo Rossa Hauck
Universidade Federal de Santa Catarina

Banca Examinadora

Prof. Dra. Fabiane Barreto Vavassori Benitti
Universidade Federal de Santa Catarina

Paula Cristina Peters Mendes Bastos
Universidade Estácio de Sá

Prof. Dr. Raul Sidnei Wazlawick
Universidade Federal de Santa Catarina

RESUMO

A metodologia ágil está conquistando cada vez mais espaço dentro do desenvolvimento de software. Uma pesquisa realizada recentemente pela VersionOne, revelou um crescimento massivo na adesão de metodologias ágeis. Contudo, um relatório apresentado pela Standish Group, revelou que somente 29% dos projetos em 2015 foram entregues com sucesso. Dentre os fatores para o sucesso apresentados no relatório, os principais estão relacionados a falha no levantamento de requisitos e a falta de participação do cliente. Apesar da importância da elicitação de requisitos para o sucesso de desenvolvimento de software, as metodologias ágeis consideram essas atividades burocráticas, tornando o projeto menos ágil. Com o intuito de resolver esse problema, em 2011, surgiu a abordagem Use Case 2.0, uma evolução da técnica de casos de uso, que vem como uma prática ágil e escalável para capturar requisitos e auxiliar na gestão do desenvolvimento. Diante desse cenário, este trabalho tem o objetivo de implantar e avaliar a abordagem Use Case 2.0 em uma unidade organizacional. A pesquisa é feita através de um estudo de caso aplicado no laboratório Bridge da Universidade Federal de Santa Catarina. Para tanto, inicialmente são revisados os conceitos fundamentais relacionados ao tema proposto através uma revisão da literatura. A partir dos conhecimentos adquiridos, é realizada uma avaliação do método de definição de requisitos utilizado por uma unidade organizacional do projeto SISMOB do laboratório. Em seguida, são aplicadas as técnicas da abordagem Use Case 2.0 em uma equipe do projeto SISMOB e, são feitas avaliações quanto aos resultados obtidos.

Palavras- chave: Desenvolvimento ágil, Elicitação de requisitos, Elicitação ágil de requisitos, Casos de uso, Use Case, Use Case 2.0

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

UML - Unified Modeling Language

SRL - Systematic literature review

ER - Engenharia de requisitos

IEEE - Institute of Electrical and Electronics Engineers

OOSSE- Object-Oriented Software Engineering

XP - Extreme Programming

SISMOB - Sistema de Monitoramento de Obras

RNI - Registro Nacional de Implantes

FB - Fluxo Básico

FA - Fluxo Alternativo

PCU – Pontos por Caso de Uso

SUMÁRIO

1. INTRODUÇÃO

Os métodos ágeis vêm ganhando cada vez mais espaço no desenvolvimento de software, o que desperta interesse tanto de profissionais da área como de pesquisadores (JAQUEIRA et al., 2012). Uma pesquisa realizada entre julho e novembro de 2015 pela VersionOne envolvendo mais de 3.000 profissionais da área de TI, revelou um crescimento massivo na adoção de práticas ágeis nas organizações, onde 95% dos entrevistados revelaram utilizar abordagens ágeis na maioria dos seus projetos (VERSIONONE, 2015).

A metodologia ágil advém de uma insatisfação das abordagens pesadas das metodologias tradicionais e a busca por um método de desenvolvimento que focasse no software e não em sua concepção e documentação (SOMMERVILLE, 2011). A definição oficial de desenvolvimento ágil de software surgiu através do Manifesto Ágil, (BECK et al., 2001) onde 17 profissionais representantes de métodos de desenvolvimento de software se reuniram para produzir um manifesto com princípios e valores que dariam origem e serviriam de base para um gerenciamento de projetos diferenciados por sua eficácia e eficiência.

Apesar de mais de uma década ter passado desde o manifesto, estudos ainda apontam problemas em projetos que usam abordagens ágeis, como expectativa do cliente não atendidas e dificuldade em estimar prazos e orçamentos (KAMEI, 2012; READ; BRIGGS, 2012, apud MEDEIROS et al., 2015).

Um relatório apresentado pelo Standish Group, o Chaos Report 2015, mostra que somente 29% dos projetos de software foram entregues com sucesso em 2014. Sendo que 19% falharam (foram cancelados antes de seu término ou entregues e nunca utilizados) e 52% estavam em estado crítico (atrasados, acima do orçamento, e/ou com funcionalidades faltando) (HASTIE, 2015). O estudo ainda apresenta uma análise sobre os fatores de sucesso para projetos de software. Dentre os 10 fatores apresentados, os 3 principais estão relacionados ao levantamento de requisitos - 13,1% são por requisitos incompletos, 12,4 % por falta de envolvimento do usuário e 8,7% por mudança de requisitos e especificação (HASTIE, 2015).

Uma das abordagens para a engenharia de requisitos é a técnica de casos de uso ou *Use Case*, um método de elicitação de requisitos, proposto por Ivar Jacobson na década de 90 e, posteriormente, incorporado à linguagem UML (SOMMERVILLE, 2011).

Segundo Jacobson et al. (2011), caso de uso pode ser visto como uma sequência de ações realizadas pelo sistema para produzir um resultado observável de valor para um determinado usuário. O conjunto de todos os casos de uso definem todas as formas de uso do sistema e o seu valor (JACOBSON et al., 2011).

Nas metodologias ágeis, a definição de requisitos é feita através de um envolvimento massivo do cliente, sendo esse o responsável por fornecer e priorizar os novos requisitos, definindo quais destes devem ser incluídos em cada iteração. Nos métodos ágeis os requisitos estão em constante modificação e são criados a partir do *feedback* dos s*takeholders*, minimizando assim, a documentação gerada (SOMMERVILLE, 2011). Porém, como a definição dos requisitos é centrada no cliente, é necessário que estes estejam dispostos e disponíveis em tempo integral (SOMMERVILLE, 2011).

Apesar da importância da elicitação e documentação de requisitos no sucesso do desenvolvimento do software e na minimização dos riscos do projeto, estas atividades são vistas pela metodologia ágil como burocráticas, tornando o processo menos ágil. Sendo assim, a falta de documentação é um dos seus principais desafios (JAQUEIRA et al., 2012).

Com o intuito de solucionar o problema da engenharia de requisitos e os métodos ágeis, novas técnicas surgiram para auxiliar a análise de requisitos em um ambiente ágil. Neste cenário, a abordagem Use Case 2.0, vem como uma prática escalável e ágil que usa casos de uso para capturar um conjunto de requisitos e impulsionar o desenvolvimento incremental de um sistema. Criado por Ivar Jacobson, Ian Spence e Kurt Bittner, o *Use-Case 2.0 - The Guide to Succeeding with Use Cases*, nada mais é do que uma evolução da técnica de casos de uso. Entre as atualizações propostas, as principais são a utilização de história de usuário para a definição do escopo do caso de uso, e o *slice,* ou corte vertical dos casos de uso, que definem um conjunto de fatias do caso de uso (JACOBSON et al., 2011).

Dentro deste contexto, este trabalho apresenta um estudo de caso da abordagem Use Case 2.0 para especificação de requisitos no projeto SISMOB do laboratório Bridge da Universidade Federal de Santa Catarina. Será apresentada toda a experiência relacionada a aplicação da abordagem e seus resultados, visando verificar o impacto da abordagem na organização. Por fim, direções futuras para o trabalho serão sugeridas.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O presente trabalho tem como objetivo geral implantar e avaliar a abordagem Use Case 2.0 em uma unidade organizacional, por meio de um estudo de caso aplicado ao projeto SISMOB do laboratório Bridge da Universidade Federal de Santa Catarina.

1.1.2 Objetivos específicos

- Realizar uma análise da literatura quanto metodologias de definição de requisitos de software, com enfoque na abordagem Use Case 2.0.

- Realizar uma revisão sistemática da literatura sobre a utilização da abordagem Use Case 2.0.

- Realizar um diagnóstico inicial de alinhamento da unidade organizacional, buscando determinar quais metodologias de definição de requisitos são utilizadas pela organização.

- Desenvolver e aplicar um conjunto de Use Cases para definição de requisitos da unidade organizacional, utilizando a abordagem Use Case 2.0

- Avaliar a abordagem Use Case 2.0 desenvolvido através de sua aplicação em um estudo de casos na unidade organizacional.

1.2 MÉTODO DE PESQUISA

Este trabalho pretende aplicar os conhecimentos adquiridos a partir de um embasamento teórico sobre a abordagem Use Case 2.0 e aplicá-los em uma unidade

organizacional, observando e analisando os impactos dessa aplicação dentro de um ciclo de desenvolvimento de software e discutindo os resultados obtidos através dessa análise.

De acordo com Yin (2011, p. 32),

> "Um estudo de caso é uma investigação empírica que investiga um fenômeno contemporâneo dentro de seu contexto da vida real, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos. Em outras palavras, você poderia utilizar o método de estudo de caso quando deliberadamente quisesse lidar com condições contextuais - acreditando que elas poderiam ser altamente pertinentes ao seu fenômeno de estudo."

Nesse contexto, o presente trabalho será um estudo de caso, exploratório, desenvolvido dentro de um ambiente real de um laboratório de desenvolvimento de software de forma a demonstrar como a aplicação da abordagem Use Case 2.0 impacta em um time de desenvolvimento de software. Visando alcançar o objetivo deste trabalho, a metodologia adotada foi dividida em três partes:

Etapa 1. Síntese da fundamentação teórica - Na primeira etapa do projeto são revisados os conceitos fundamentais da literatura.

Atividade 1.1 - Análise da engenharia de requisitos.

Atividade 1.2 - Análise sobre métodos ágeis de desenvolvimento.

Atividade 1.3 - Análise da metodologia de casos de uso para definição de requisitos.

Atividade 1.4 - Análise da abordagem Use Case 2.0.

Etapa 2. Definição do estado da arte - Nesta etapa é realizada uma revisão sistemática da literatura (SRL) com o objetivo de (KITCHENHAM, 2004) fornecer meios para identificar, selecionar e analisar evidências relacionadas a experiências de utilização da abordagem Use Case 2.0 em ambientes reais.

Para aplicação do SLR, primeiramente é necessário a definição de um protocolo contendo a motivação e apresentação de um problema real, fundamentado na literatura. Após definido a problemática, é necessário o levantamento de uma ou mais perguntas de pesquisa que devem ser respondidas ao final da revisão. É necessário também definir em

quais bases serão realizadas as buscas e determinar os critérios de inclusão e exclusão dos estudos encontrados. Para realizar a busca deve-se definir inicialmente os termos de busca e a partir dele, criar uma *string* adaptada para cada base de dados pesquisada. Logo após definida a *string*, é preciso executar as buscas nas bases definidas e, somente os estudos mais relevantes devem ser considerados. Para isso, é necessário realizar uma análise superficial dos estudos encontrados, descartando os que não atende o critério de inclusão. Em seguida, deve-se realizar uma análise mais aprofundado dos estudos selecionados, realizado uma leitura completa dos mesmos e, definindo assim, os estudos relevantes a revisão. Por fim, é necessário realizar a extração dos dados que podem responder as perguntas da pesquisa.

Atividade 2.1 - Descrição do problema

Atividade 2.2 - Especificação das perguntas de pesquisa

Atividade 2.3 - Definição do protocolo de busca

Atividade 2.4 - Execução da busca

Atividade 2.5 - Extração e análise dos dados

Etapa 3. Definição, aplicação e análise do estudo de caso - Nesta terceira etapa é definida contextualização, objetivos, planejamento, aplicação e avaliação do estudo de caso.

Atividade 3.1 - Análise de contexto.

Atividade 3.2 - Definição do estudo.

Atividade 3.3 - Planejamento do estudo.

Atividade 3.4 - Execução do estudo.

Atividade 3.5 - Análise dos dados coletados.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os conceitos teóricos fundamentais utilizados para a compreensão deste trabalho. Primeiramente são apresentados os conceitos de engenharia de requisitos e metodologias de desenvolvimento ágil. Em um segundo momento são apresentadas as técnicas de casos de uso e Use Case 2.0.

2.1 ENGENHARIA DE REQUISITOS

Os requisitos são as descrições das funcionalidades de um sistema, seus serviços e restrições de funcionamento, refletindo a necessidade dos usuários para um sistema que serve a uma determinada finalidade (SOMMERVILLE, 2011). Segundo Pressman (2011) uma compreensão completa dos requisitos é fundamental para o sucesso de um produto. Os requisitos são o primeiro passo para a definição de um sistema e, por isso, um ponto decisivo no sucesso do desenvolvimento de um software. O processo de descobrir analisar, documentar e verificar os requisitos é chamado de engenharia de requisitos (SOMMERVILLE, 2011).

O *Institute of Electrical and Electronics Engineers* (ISO/IEC/IEEE, 2011) define Engenharia de Requisitos (ER) como o processo de aquisição, refinamento e verificação das necessidades dos clientes para um sistema de software, com o objetivo de ter uma especificação completa e correta dos requisitos de software. Segundo Sommerville (2011), a engenharia de requisitos tem como objetivo produzir um documento de requisitos acordados que especifica um sistema que satisfaz os requisitos do *stakeholders* (SOMMERVILLE, 2011).

Na literatura existem várias propostas de procedimentos de análise e gerenciamento da engenharia de requisitos (KOTONYA e SOMMERVILLE, 1998). Este trabalho baseia-se nos fundamentos do modelo proposto por Sommerville (2011) que descreve a ER como sendo composta por quatro atividades de alto nível, conforme Figura 1, para o levantamento de requisitos, sendo estas, **estudo de viabilidade** - busca avaliar se um sistema é útil para a empresa; **elicitação e análise de requisitos** - visa descobrir os requisitos; **especificação de requisitos** - converte os requisitos levantados em uma forma padrão; **validação de requisitos** - realiza a validação dos requisitos junto ao cliente.

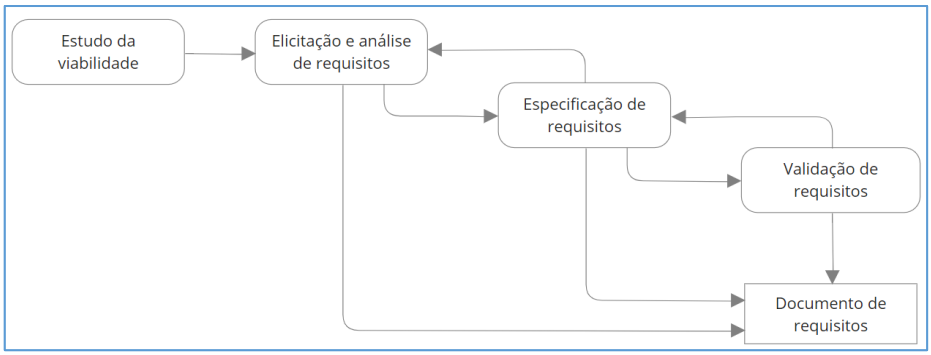


Figura 1 - Os requisitos da engenharia de processo
Fonte: SOMMERVILLE, 2011 - Adaptado

As atividades no processo de requisitos não são feitas em apenas uma sequência. Na prática, a engenharia de requisitos é um processo iterativo, inter-relacionados, com retroalimentação, em que as atividades são organizadas em forma de espiral, conforme Figura 2 e, são repetidas até a aceitação de um documento de requisitos. A quantidade de esforço dedicado a cada atividade em cada iteração depende do estágio do processo e de que tipo de sistema está sendo desenvolvido. Caso algum problema seja encontrado ao longo do desenvolvimento do sistema, as etapas de elicitação, especificação, validação e documentação entram novamente na espiral. Isso ocorre tantas vezes quantas forem necessárias, até que não haja mais inconsistências nos requisitos. Nesse momento, o documento final de requisitos é elaborado e inicia-se o processo de gerenciamento de requisitos que visa compreender e controlar as mudanças nos requisitos de um sistema (SOMMERVILLE, 2011).

2.1.1 Estudo de viabilidade

A etapa de estudo de viabilidade, também conhecida como etapa de concepção, serve como ponto de partida para as demais atividades da engenharia de requisitos. Seu objetivo é um entendimento do problema a ser tratado pelo software a ser desenvolvido, quais são os *stakeholders* e a natureza da solução desejada (PRESSMAN, 2011). Nesta etapa, é realizada uma estimativa acerca das possibilidades de se satisfazerem as necessidades do usuário identificado, usando as tecnologias atuais, a fim de definir se haverá um avanço do projeto. (SOMMERVILLE, 2011).

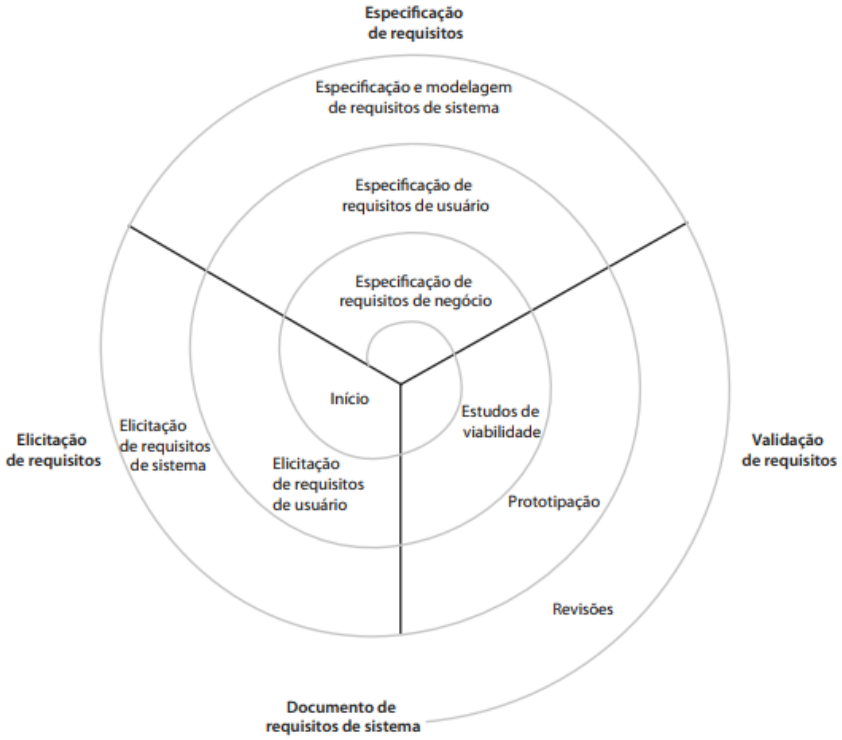


Figura 2 - Espiral do processo de Engenharia de Requisitos
Fonte: SOMMERVILLE, 2011

Segundo Pressman (2011), deve-se inicialmente tentar entender e reconhecer os elementos problemáticos básicos, conforme percebidos pelo usuário/cliente. Logo que o problema é identificado, determina-se quais dados serão produzidos e recebidos pelo sistema. Com isso definido, inicia-se a sintetização de uma ou mais soluções a fim de obter modelos do sistema com intuito de compreender melhor o fluxo de dados, controle, processamento funcional, além do conteúdo de informações. O modelo servirá como fundamento para o projeto de software e como base para a levantamento dos requisitos.

2.1.2 Elicitação de requisitos

A elicitação e análise de requisitos, pode ser dividida em duas partes, (i) elicitação e a análise, (ii) negociação de requisitos. A elicitação também conhecida como captura, descoberta, aquisição ou levantamento de requisitos, procura descobrir os requisitos, identificar limites de sistema e os *stakeholders*. Compreender o domínio do aplicativo, necessidades do negócio, restrições do sistema, *stakeholders* e o problema em si é parte

essencial para entender o sistema a ser desenvolvido (BOURQUE, 2014). Segundo Wazlawick (2011), a etapa de elicitação de requisitos visa buscar todas as informações possíveis sobre as funções que o sistema deve executar e as restrições sobre as quais o sistema deve operar.

Enquanto na etapa de viabilidade o foco está no escopo geral do projeto, durante a etapa de elicitação o objetivo principal está na identificação das necessidades do negócio existente, procurando entender de forma detalhada as necessidades dos *stakeholders*, por meio de diferentes abordagens (PRESSMAN, 2011). Na elicitação, os requisitos são identificados através de documentos, *stakeholders* de sistema e especificação de sistema similares. Para Sommerville (2011), a elicitação dos requisitos é um processo de descoberta através da comunicação entre engenheiros de software e os clientes, usuários finais e *stakeholders*.

Elicitar e compreender os requisitos dos *stakeholders* é um processo difícil devido a diversos fatores. Pressman (2011) identifica os tipos de problemas que podem dificultar o levantamento de requisitos:

Problema de escopo - O limite do sistema é mal definido ou o *stakeholder* especifica detalhes técnicos que podem causar confusão dos objetivos.

Problema de entendimento - O *stakeholder* não apresenta de forma clara suas necessidades, e possui pouco entendimento sobre o domínio do problema, com dificuldade de comunicação, e muitas vezes, omitindo informações importantes.

Problema de volatilidade - Mudança de requisitos ao longo do ciclo de vida do software, o que pode dificultar o gerenciamento.

Para reduzir esses problemas, é necessário a aplicação de técnicas de elicitação, a fim de obter uma comunicação sólidas, princípios de análise fundamentais e métodos de análise sistemáticos (PRESSMAN, 2011).

Existem diversas técnicas para realizar a elicitação dos requisitos, tais como:

Entrevista - É um método de obter uma compreensão global sobre o que os *stakeholders* fazem, como podem interagir com o novo sistema e as possíveis dificuldades enfrentadas nos sistemas atuais. Erros e mal-entendidos podem ser identificados e esclarecidos durante a entrevista. Existem dois tipos diferentes de entrevista, **entrevista fechada**, onde o *stakeholder* responde a um conjunto de perguntas pré-definidas. E, **entrevista aberta**, sem perguntas pré-definidas, onde há uma discussão com os *stakeholders* para definir o que estes esperam do sistema (SOMMERVILLE, 2011).

Na verdade, não há uma distinção clara entre esses tipos. Comumente, as questões levam a discussões de outros pontos que podem ser abordados mais abertamente. As entrevistas contribuem para uma obtenção de uma rica coleção de informação. Porém, ela por si só pode deixar escapar informações essenciais, por isso, deve ser usada associada a outras técnicas (SOMMERVILLE, 2011).

Cenários e Casos de uso - Casos de uso descrevem as interações entre usuário e o sistema, com foco nas necessidades do usuário. Essas iterações podem ser descritas como cenários (SOMMERVILLE, 2011). A técnica de caso de uso será apresentada na Capítulo 2.3.

Prototipação - É um processo que capacita o desenvolvedor a criar um modelo do software a ser implementado (PRESSMAN, 2011). Um protótipo é uma versão inicial do sistema que pode assumir uma das três formas:

- Um protótipo em papel ou modelo baseado em PC que retrata a interação homem-máquina
- Um protótipo de trabalho que implementa algum subconjunto de funções do sistema
- Um programa já existente que já possui boa parte ou toda a função desejada, mas possui características a serem melhoradas.

Os protótipos servem idealmente para identificar requisitos de sistema e são um paradigma eficiente da engenharia de software (PRESSMAN, 2011).

Brainstorming - É uma técnica para geração de ideias, ajudando a desenvolver de forma criativa, soluções para os problemas. Brainstorming contém duas fases, a de geração, onda as ideias são coletadas, e a fase de consolidação, onde as ideias coletadas são discutidas. Na primeira fase, as ideias não devem ser discutidas e avaliadas, apenas geradas. Ao final dessa primeira fase, com todas as ideias levantadas, inicia-se a consolidação, onde as ideias são discutidas, revisadas, organizadas e avaliadas. A técnica de Brainstorming leva a uma melhor compreensão do problema e um sentimento de propriedade comum do resultado (PRESSMAN, 2011).

Etnografia - A etnografia, tem o intuito de descobrir requisitos implícitos do sistema que refletem o contexto real do usuário através de uma imersão no ambiente que o sistema será utilizado. A observação do ambiente organizacional contribui para coletar as atividades desempenhadas pelo usuário de forma natural, podendo revelar detalhes críticos de processo que, muitas vezes, são ignorados por outras técnicas de elicitação de

requisitos. Contudo, por ter foco no usuário, esta abordagem pode não levar em consideração os requisitos da organização (SOMMERVILLE, 2011).

2.1.3 Análise e negociação de requisitos

A informações obtidas nas etapas de viabilidade e elicitação são expandidas e refinadas durante a análise e negociação de requisitos. Nesta etapa os requisitos são detalhadamente analisados para identificar inconsistências, relação de dependência e conflitos, identificar se os requisitos são funcionais ou não funcionais e, classificar os requisitos de acordo com as categorias a que pertencem (PRESSMAN, 2011).

Ao finalizar a análise, deve haver uma negociação dos requisitos a fim de solucionar conflitos. Este processo é necessário pois inevitavelmente os diferentes *stakeholders* têm opiniões diferentes sobre a importância e prioridade dos requisitos. Durante o processo, deve-se organizar as negociações regulares dos *stakeholders*, de modo que o os compromissos possam ser cumpridos (KOTONYA e SOMMERVILLE, 1998).

2.1.4 Especificação de requisitos

Posterior a fase de análise e negociação, a especificação de requisitos refere-se a produção de um documento que possa ser sistematicamente revisado, avaliado e aprovado (BOURQUE, 2014). Segundo Pressman (2011) especificação é o produto de trabalho final, podendo ser um documento escrito, um modelo gráfico, um modelo matemático formal, uma coleção de cenários de uso, um protótipo ou qualquer combinação desses elementos.

Segundo a norma ISO/IEC/IEEE 29148:2011, para sistemas complexos envolvendo componentes não-software, ao menos três documentos são produzidos (ISO/IEC/IEEE, 2011):

Documento de especificação de requisitos de partes interessadas - Este documento descreve a motivação da organização para o desenvolvimento do sistema. Destinado aos *stakeholders*, o documento define os requisitos do sistema em alto nível, seus objetivos gerais, o ambiente alvo, suas restrições e pressupostos.

Documento de definição de sistema - Este documento fornece uma descrição do que o sistema deve fazer, em termos de interações com o usuário ou interfaces com o seu meio externo. Este documento apresenta os requisitos do *stakeholder* para os desenvolvedores do sistema.

Documento de especificação de requisitos de software - É uma declaração oficial de o que os desenvolvedores do sistema devem implementar. Deve incluir os requisitos de usuário para um sistema e a especificação detalhada dos requisitos do sistema.

Além disso, a norma também fornece um esboço para a produção desses documentos. Por ser uma norma genérica, esta pode ser adaptada para usos específicos (ISO/IEC/IEEE, 2011).

2.1.5 Validação de requisitos

Esta etapa consiste em verificar se o documento de requisitos produzido na etapa anterior realmente contempla as necessidades do cliente/usuário (SOMMERVILLE, 2011). Esta etapa tem por objetivo detectar faltas de consistência, omissão, ambiguidade ou falta de conformidade nos requisitos já especificados (PRESSMAN, 2011).

A validação de requisitos é de extrema importância pois, erros encontrados nessa etapa tendem a ser menos custosos do que se forem encontrados quando o sistema já estiver desenvolvido. Durante essa fase, diferentes tipos de validação devem ser feitas com os requisitos elicitados no documento, tais como: validade, consistência, completude, realismo, verificabilidade, revisão de requisitos, prototipação, geração de casos de teste (SOMMERVILLE, 2011).

Durante o processo de validação de requisitos raramente serão encontrados todos os problemas de requisitos. Após os ajustes do documento de requisitos, é inevitável a necessidade de mudanças nos requisitos para corrigir omissões e equívocos (SOMMERVILLE, 2011)

2.1.6 Gerenciamento de requisitos

A gestão de requisitos é um processo paralelo às etapas anteriores, e está relacionado ao gerenciamento de mudanças. Segundo Sommerville (2011), as mudanças nos requisitos são inevitáveis, isso porque, os sistemas geralmente são desenvolvidos para resolver problemas que não podem ser completamente definidos. Sendo assim, este processo tem como objetivo de compreender e controlar das mudanças nos requisitos do sistema (SOMMERVILLE, 2011).

Embora mudanças nos requisitos sejam inevitáveis, certos requisitos são mais estáveis do que outros, sendo que estes correspondem aos requisitos relacionados a essência do sistema e o domínio da aplicação. Os demais requisitos, denominados voláteis, podem ser divididos em: requisitos mutáveis - que se altera, em função do ambiente do sistema; requisitos emergentes - que não foram completamente definidos no processo de elicitação; requisitos consequentes - que são resultados de considerações sobre a maneira como o sistema será utilizado; requisitos de compatibilidade - que dependem de outros equipamentos ou processos. Essa identificação prévia dos requisitos voláteis facilita a gestão dos requisitos, pois é possível prever quais os requisitos com mais chance de sofrerem alteração (KOTONYA e SOMMERVILLE, 1998).

O planejamento da gestão de requisitos é uma parte muito importante. Com isso, devem estar definidas desde o início da gestão de requisitos as seguintes políticas (SOMMERVILLE, 2011):

Identificação de requisitos - Identificação única dos requisitos com o intuito de auxiliar na rastreabilidade.

Processo de gestão de mudanças - Conjunto de atividades que permitem avaliar o impacto e o custo da aplicação das mudanças desejadas.

Rastreabilidade - Relação entre os requisitos e os requisitos e o projeto de sistema.

Ferramentas a utilizar - Para grandes sistemas, a quantidade de informações para processar pode ser elevada, sendo aconselhado o uso de alguma ferramenta.

Para manter a consistência entre as inúmeras mudanças, é importante que o processo de gestão de mudanças esteja bem definido. Segundo Kotonya e Sommerville (1998), o processo deve estar dividido em três estágios. No primeiro estágio, deve-se identificar o problema existente nos requisitos, analisá-lo e propor alterações. A viabilidade da mudança proposta deve ser definida no segundo estágio e, deve ser medida em relação ao tempo, aos custos e à forma como essa mudança afetará os demais

requisitos. No último estágio, as mudanças aprovadas devem ser implementadas e os novos requisitos validados (KOTONYA e SOMMERVILLE, 1998).

2.2 MÉTODOS ÁGEIS

Na economia moderna é frequentemente difícil ou impossível prever como um sistema computacional irá evoluir com o tempo. As condições de mercado e as necessidades dos usuários finais mudam constantemente (PRESSMAN, 2011). Processos de desenvolvimento de software que planejam especificar completamente os requisitos e, em seguida projetar, construir e testar não estão adaptados ao desenvolvimento rápido. Por isso, no início da década de 90 acompanhou-se o surgimento de novos métodos de desenvolvimento, com o intuito de auxiliar esse novo formato de produção de software (SOMMERVILLE, 2011).

Essas metodologias passaram a ser chamadas de "leves" por não utilizarem as formalidades que caracterizam os processos tradicionais e por evitarem a burocracia imposta pela utilização excessiva de documentos. Com o tempo, algumas delas ganharam destaque nos ambientes empresarial e acadêmico, gerando grandes debates, principalmente relacionados à confiabilidade dos processos e à qualidade do software (BASSI, 2008)

Após alguns anos de experiência nessas novas metodologias ágeis, em 2001, dezessete renomados desenvolvedores, autores e consultores da área de software, percebendo que seus métodos de trabalho, além de eficazes, eram parecidos, se reuniram para discutir e chegar a um consenso que refletisse suas ideias e conceitos em relação à situação atual do desenvolvimento de software. O resultado dessa reunião foi a criação da *The Agile Alliance*, e a assinatura do "Manifesto ágil de desenvolvimento de software", que se inicia do seguinte modo:

> *Desenvolvendo e ajudando outros a desenvolver softwares, estamos desvendando formas melhores de desenvolvimento. Por meio deste trabalho passamos a valorizar:*
>
> *Indivíduos e interação acima de processos e ferramentas*
> *Software operacional acima de documentação completa*
> *Colaboração dos clientes acima de negociação contratual*
> *Resposta a mudanças acima de seguir um plano*

> *Ou seja, embora haja valor nos itens à direita, valorizamos os da esquerda ainda mais* (BECK et al., 2001).

Os métodos ágeis valorizam os processos, ferramentas, documentação, contratos e planos. Porém, dá mais valor aos indivíduos, interações, software funcionando, colaboração do cliente e resposta a mudança. (WAZLAWICK, 2013).

Métodos ágeis, baseiam-se em uma abordagem incremental para a especificação, desenvolvimento e entrega do produto. Destinam-se a entregar o software rapidamente, funcionando, sendo que podem ser incluídas alterações e novos requisitos a serem desenvolvidos nas iterações posteriores do sistema (SOMMERVILLE, 2011). Apesar dos métodos ágeis possuírem diferentes processos, eles compartilham um conjunto de 12 princípios de agilidade, estabelecidos pelo Agile Alliance.

Segundo Pressman (2011), nem todo o modelo de processo ágil aplicam esses 12 princípios atribuindo pesos iguais, alguns modelos preferem dar maior importância a um ou mais desses princípios. Entretanto, esses 12 princípios definem a agilidade mantida por cada um dos modelos de processo.

Neste capítulo, o Scrum, por ser um dos métodos ágeis mais conhecidos e utilizados atualmente (SOMMERVILLE, 2011), e por ser a metodologia utilizada pela organização em que este estudo de casos foi aplicado, é brevemente apresentado e discutido.

2.2.1 SCRUM

Utilizado para o gerenciamento de projetos de software, o Scrum é um *framework* que tem como base o empirismo, ou seja, o conhecimento é obtido através de experiências anteriores e as decisões são tomadas de acordo com esse conhecimento (SCHWABER, 2002). Usado para resolver problemas complexos e adaptativo, o Scrum auxilia na produção e criatividade e entrega dos produtos com o maior valor possível. Scrum é considerado um framework leve, simples de entender e difícil de dominar (SCHWABER, 2002).

O empirismo é sustentado em três pilares, sendo eles (SCHWABER, 2011):

Transparência - Os principais aspectos do processo devem estar visíveis aos responsáveis pela sua execução. Para isso, é necessária uma definição padrão para que os envolvidos no processo compartilhem um entendimento comum sobre o que está sendo observado.

Inspeção - Os envolvidos no projeto devem frequentemente inspecionar os artefatos e o progresso do projeto para observar se os objetivos estão sendo alcançados. Através dessas inspeções é possível detectar variações indesejadas e, se necessário, aplicar medidas corretivas. Essas inspeções não devem ser feitas com tanta frequência a ponto de atrapalhar o andamento do projeto

Adaptação - Ao observar que um ou mais aspectos de um processo estão fora dos limites aceitáveis, o processo deve ser ajustado. O ajuste deve ser executado o mais brevemente possível, a fim de minimizar o surgimento de mais desvios.

O Scrum possui uma série de valores, conceitos e práticas que tem como objetivo maximizar as chances de sucesso do projeto. A seguir são apresentados seus principais componentes: Eventos, Artefatos e Time Scrum (PRESSMAN, 2011).

2.2.1.1 Eventos

Usados para criar regularidade e minimizar a necessidades de reuniões, os eventos, inspecionam e avaliam o que ocorre durante o projeto. Todos os eventos são *timebox*, ou seja possuem um período determinado de tempo para ocorrer. O Scrum possui os seguintes eventos (SCHWABER, 2011):

Sprint - A Sprint nada mais é do que uma divisão do projeto em iteração. Principal evento do Scrum, a Sprint possui tipicamente uma duração de quatro semanas. Durante esse período é concebida uma versão funcional do produto. Um novo Sprint é iniciado ao final da anterior. Nenhuma mudança que possa afetar o objetivo da Sprint deve ser aceita durante o seu andamento. Cada Sprint pode ser vista como um projeto, sendo assim, possuindo um objetivo final e gerando uma versão funcional do produto. A Figura 3 e 4 representam as características do Sprint.

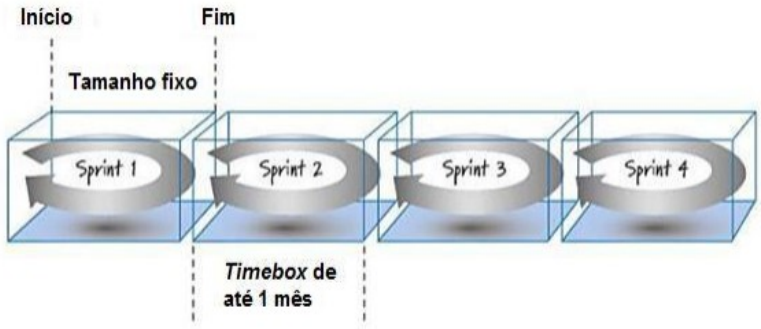


Figura 3 -Sprint no projeto
Fonte: RUBIN, 2012 - Adaptado

Figura 4 - Características de cada Sprint
Fonte: RUBIN, 2012 - Adaptado

Sprint Planning Meeting - É uma reunião para a planejamento da Sprint. Todo o Time Scrum deve participar de forma colaborativa. Com duração de no máximo 8 horas para o planejamento de uma Sprint de um mês de duração. A reunião consiste em basicamente duas partes, cada qual com metade de duração da reunião. Na primeira metade, é definido o que será entregue ao final da Sprint. Na segunda parte, é estimado o esforço de trabalho necessário para ser atingido o objetivo definido na primeira etapa.

Daily Scrum - É um evento, de até no máximo 15 minutos, que ocorre diariamente. E tem por objetivo alinhar as atividades realizadas desde a última *Daily* Scrum, e planejar as atividades a serem realizadas até a próxima *Daily*. Durante cada reunião cada integrante da equipe precisa responder três perguntas: o que foi feito desde a última reunião, o que será feito após a reunião e se existe algum obstáculo sendo

enfrentado. *A Daily Scrum* serve para que a equipe possa medir e avaliar o andamento da Sprint.

Sprint Review Meeting - Realizada ao final de uma Sprint, a *Sprint Review Meeting* é uma reunião que tem por objetivo apresentar o resultado desenvolvido durante a Sprint. O tempo deste evento deve ser proporcional ao número de semana da Sprint. Os problemas e soluções encontrados durante a Sprint devem ser apresentados nessa reunião.

Sprint Retrospective - É uma reunião para que a equipe possa se auto avaliar e planejar melhoria e definir os próximos passos do projeto. O evento deve durar no máximo 3 horas e, durante esse tempo deve ser feito uma avaliação sobre os principais aspectos da Sprint, principalmente no que diz respeito a pessoas, processos e ferramentas. Nesta reunião deve ser levantada também as possíveis melhorias a partir dos aspectos que tiveram sucesso. Por fim, essas melhorias devem ser inseridas em um plano para serem aplicadas pela equipe no próximo Sprint.

2.2.1.2 Artefatos

Os artefatos são gerados a partir dos eventos da Scrum, utilizando como premissa fornecer transparência e permitir inspeção e adaptação. Os eventos geram os seguintes artefatos (SCHWABER, 2011):

Product Backlog - É uma lista ordenada de tudo o que é necessário no produto final. No Scrum, essa é a única fonte de requisitos disponível para consulta. *O Product Backlog* é dinâmico, ou seja, está sempre sendo atualizado. Mudanças nos requisitos de negócio, condições de mercado ou tecnologia provocam mudanças no *Product Backlog*. Este artefato contém requisitos, funcionalidades, funções, melhorias entre outros. Cada item descrito geralmente possui uma descrição, prioridade e estimativa. Para produtos novos, o *Backlog Product* é composto basicamente de requisitos, para produtos já desenvolvidos, o *Backlog* pode conter mudanças, melhorias e correções (RUBIN, 2012). A Figura 5 apresenta um exemplo de organização do *Product Backlog*, com as atividades de criação e refinamento dos itens que irão compor a lista.

Figura 5 - Processo de desenvolvimento do *Product Backlog*
Fonte: RUBIN, 2012 – Adaptado

Normalmente, os requisitos são especificados através de *user stories* e, quanto maior a prioridade do item, mais claro e detalhado esse deve ser. Além disso, para cada item deve ser atribuído uma estimativa de tamanho/complexidade.

Sprint Backlog - Define as funcionalidades e requisitos que a equipe deverá executar durante a Sprint. Ao surgir uma nova atividade, essa deve ser adicionada à lista. Cada nova funcionalidade adicionada a Sprint deve conter a prioridade e estimativa de complexidade.

Burndown Backlog - É um gráfico que possibilita a visualização do progresso das atividades dentro de uma Sprint. Geralmente, o eixo vertical representa a quantidade de atividades restante para completar o Sprint, enquanto o eixo horizontal representa os dias da sprint.

Taskboard - Ferramenta para auxiliar o acompanhamento do Sprint e suas atividades. Deve estar sempre visível e acessível a todos do projeto, agregando transparência e visibilidade ao processo. Deve ser atualizado durante o *Daily Scrum*.

2.2.1.3 O Time Scrum

O Time Scrum é auto organizável e multifuncional, ou seja, sabe como entregar suas tarefas e possuem competência suficiente para realizá-las sem auxílio. Há três perfis que definem um time Scrum (SCHWABER, 2011):

Product Owner - É o responsável por representar o cliente, definindo os requisitos e as prioridades. O *Product Owner* é uma pessoa, e não uma equipe, contudo

pode representar um conjunto de pessoas. Sua responsabilidade é o gerenciamento do *Product Backlog*, desde a elicitação dos requisitos até o surgimento de mudanças durante o projeto. Durante a criação do *Product Backlog* o *Product Owner* deve expressar claramente os itens na lista, ordená-los e garantir que o de time de desenvolvimento entenda os itens no nível necessário.

Time de Desenvolvimento - São, basicamente, todos os envolvidos na entrega de um incremento do projeto. São equipes, dinâmicas, capazes de organizarem e gerenciarem suas próprias atividades. O tamanho de uma equipe pode variar entre 6 e 10 pessoas. Uma equipe muito pequena, pode encontrar dificuldade quanto às habilidades necessárias. Já uma equipe muito grande, pode não conseguir se auto gerenciar.

Scrum Master - É o responsável por auxiliar todos os envolvidos no projeto a compreenderem todas as práticas, regras, teorias e valores do Scrum. Além disso, o Scrum Master deve remover qualquer tipo de impedimento ao progresso do Time de Desenvolvimento e *Product Owner*.

2.3 CASOS DE USO

Casos de uso são uma maneira simples e poderosa de expressar os requisitos funcionais ou comportamentos de um sistema (BITTNER, 2002). São processos de interação com o sistema que tem início e fim em tempo contíguo, ou seja, são executados muito rapidamente (WAZLAWICK, 2011). Segundo Jacobson (2011), caso de uso pode ser visto como uma sequência de ações realizadas pelo sistema para produzir um resultado observável de valor para um determinado usuário. O conjunto de todos os casos de uso definem todas as formas de uso do sistema e o seu valor (JACOBSON et al., 2011).

Os casos de uso são uma técnica de elicitação de requisitos introduzida inicialmente pela metodologia de software OOSSE - *Object-Oriented Software Engineering*, como descrito no livro *Object-Oriented Software Engineering: A Use Case Driven Approach* (JACOBSON et al., 1993 apud SOMMERVILLE, 2011). Tornando-se, posteriormente, uma característica fundamental da linguagem de modelagem unificada (UML - do inglês unified modeling language) (SOMMERVILLE, 2011).

A UML é uma linguagem-padrão para criar estrutura de projetos de software, podendo ser empregada para a visualização, especificação, construção e documentação de artefatos de sistemas complexos de software. A UML é independente do processo,

apesar de ser perfeitamente utilizada em processos orientados a casos de uso, centrado na arquitetura, iterativo e incremental (BOOCH et al, 2000).

A definição de casos de uso, na ótica da UML, é: “uma descrição de um conjunto de sequências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator.” (OMG, 2015). A UML também fornece uma representação gráfica de um caso de uso, conforme Figura 6. Nessa representação, as elipses representam casos de uso, os bonecos representam os atores e o retângulo representa o sistema (BOOCH et al, 2000).

O diagrama pode ser visto apenas como um auxílio visual do resumo das funcionalidades do sistema. Nele, é possível visualizar a interação entre o ator e o caso de uso. Para melhor compreensão, é importante que o diagrama não contenha um número muito grande de elipses. Geralmente, é representado no diagrama apenas os processos que podem ser executados isoladamente (WAZLAWICK, 2011).

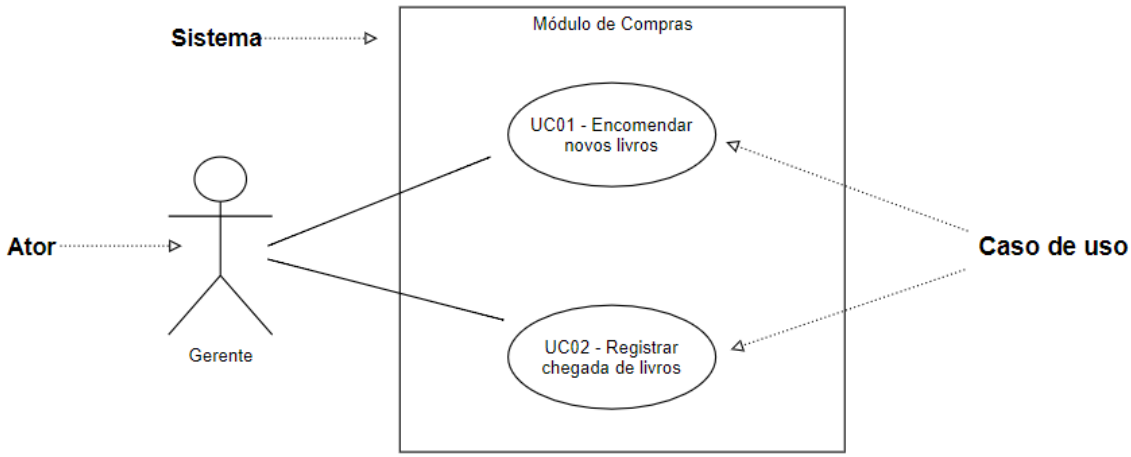


Figura 6 - Diagrama de casos de uso da UML
Fonte: WAZLAWICK, 2011 - Adaptado

Para a UML 2.5 o conceito chave da especificação dos casos de uso são Atores, Casos de Uso e Sistema (OMG, 2015).

Ator - Pode ser representado por pessoas ou outros sistemas e definem as funções que esses reproduzem ao interagir com o sistema. O ator está sempre fora do sistema e, geralmente não pode ser controlado por ele (BITTNER, 2002). A UML define ator como, um tipo de papel desempenhado por uma entidade que interage com o sistema alvo. Os atores podem representar papéis desempenhados por usuários humanos, hardware externo ou outros sistemas. (OMG, 2015).

Caso de Uso - Descreve como um sistema e seus atores colaboram para cumprir, no mínimo, um dos objetivos dos atores. Os casos de uso fornecem uma imagem coerente de como o sistema será utilizado e quais são suas funcionalidades (BITTNER, 2002). Para a UML, caso de uso é um comportamento ou função que o sistema pode executar em colaboração com um ou mais atores. Um caso de uso pode incluir possíveis variações de seu comportamento básico, incluindo comportamento excepcional e tratamento de erros. Casos de uso podem ser utilizados tanto para especificação de requisitos externos sobre um sistema, quanto para especificação de funcionalidades oferecidas por um sistema (OMG, 2015).

Sistema (*Subject*) - Um "sujeito" que pode ser um sistema ou elemento e que é responsável pelo comportamento descrito no caso de uso. É o sujeito que o caso de uso se refere (OMG, 2015).

A modelagem de casos de uso fornece muito mais do que uma simples representação visual da interação entre um sistema e seus atores. Um caso de uso tem como objetivo principal descrever, através de texto em linguagem natural, o que o sistema faz para um ator em particular (BITTNER, 2002).

2.3.1 Narrativa de casos de uso

Todo caso de uso requer uma especificação descrevendo sua funcionalidade. É nessa narrativa, que os detalhes do caso de uso, também conhecido como propriedades do caso de uso, são definidos (BITTNER, 2002). Essa descrição deve ser feita através de um texto em linguagem natural, apresentando o conjunto de comportamentos realizados pelos casos de uso (OMG, 2015). Wazlawick (2011), afirma que para descrever os casos de uso deve-se realizar um exame detalhado do processo envolvido, definindo como ocorre a interação entre os atores e o sistema.

As narrativas dos casos de uso são base para a modelagem de casos de uso. Desde a popularização, em 1992, surgiram inúmeras abordagens para a descrição dos casos de uso. Cada abordagem recomenda um estilo de escrita diferente e níveis de detalhe e de conteúdo (BOOCH et al., 2000). O estilo aqui apresentado será o proposto por Ivar Jacobson em seu livro *Object-oriented software engineering a use case driven approach* e posteriormente adotado pelo RUP. Essa modelagem se concentra nas principais

propriedades dos casos de uso: fluxo de eventos, pré-condição e pós-condição (BITTNER, 2002)

2.3.1.1 Fluxo e eventos

O fluxo de eventos descreve os passos, em uma sequência lógica, desde o seu começo até que os objetivos sejam atingidos (JACOBSON, 2011). Apresenta uma descrição da colaboração do sistema e dos atores para a entrega de valor do caso de uso, acrescentando os impedimentos para que esse valor possa ser alcançado (BITTNER, 2002).

Para facilitar a quantificação, identificação e entrega do valor, é necessário estruturar a narrativa do caso de uso. Inicialmente, deve-se definir a sequência mais simples para alcançar o valor do caso de uso. Em seguida, captura-se todas as formas alternativas de alcançar esse valor e como deve-se lidar com problemas que possam atrapalhar o valor de ser alcançado (JACOBSON, 2011). A sequência mais simples é chamada de fluxo básico. Todas as outras sequências, independentemente se terminam com sucesso ou não, são chamas de fluxo alternativo (BITTNER, 2011).

O fluxo básico é a descrição do caminho normal e esperado. Também referido como caminho feliz, é o caminho que a maioria dos usuários utiliza, na maioria das vezes (JACOBSON, 2011). Já o fluxo alternativo pode ser considerado como os desvios. Ou seja, é a descrição de quaisquer outras formas de usar o sistema para o objetivo (BITTNER, 2011). Abaixo é apresentado um exemplo de descrição de um fluxo para o caso de uso *Retirar dinheiro de um caixa eletrônico*. Sendo a maneira mais simples de alcançar o objetivo está descrito no fluxo básico e, os outros modos de alcançar o objetivo estão descritos nos fluxos alternativos (JACOBSON, 2011).

FB - Fluxo Básico

1. O cliente insere o cartão de crédito
2. O sistema valida o cartão de crédito
3. O cliente seleciona a opção “Retirada em dinheiro”
4. O cliente seleciona a conta para a retirada
5. O sistema verifica a disponibilidade de fundos
6. O sistema retorna o cartão ao cliente

7. O sistema entrega o dinheiro

FA1 - Fluxo Alternativo - Cartão inválido

...

FA2 - Fluxo Alternativo - Saldo insuficiente na conta

…

Esta estrutura de casos de uso facilita a validação da completude do sistema, ao mesmo que tempo que apresenta as diferentes formas de utilizar o sistema que oferecem pouco ou nenhum valor real para o usuário. Esse foco constante no valor do caso de uso, permite que todas as versões do sistema sejam tão pequenas quanto possível, oferecendo valor para os usuários e *stakeholders* (JACOBSON et al, 2011).

2.3.1.2 Pré-condição e Pós-condição

A pré-condição são fatos considerados verdadeiros antes do início do caso de uso (WAZLAWICK, 2011). São o ponto de partida, representados pelo estado do ator e do sistema no momento em que o caso de uso deve ser iniciado. As condições prévias não são uma descrição do evento para que o caso de uso se inicie, mas sim uma declaração das condições necessárias para que o caso de uso seja aplicado. A pré-condição é necessária para que o caso de uso seja iniciado, mas não é suficiente para iniciar o caso de uso (BITTNER, 2002).

As pós-condições define os resultados do caso de uso, ou seja, o que será verdadeiro após sua execução, (WAZLAWICK, 2011), independentemente de quais fluxos alternativos foram executados (BITTNER, 2002). Por isso, podem existir inúmeras pós-condições para um mesmo caso de uso. A definição clara das pós-condições define o estado em que o sistema deve ser deixado quando o fluxo de eventos terminar (BITTNER, 2002).

2.3.2 Artefatos de suporte aos casos de uso

Os casos de uso não são suficientes para especificar completamente os requisitos de um sistema. É necessário uma ou mais documentações adicionais relacionadas aos

requisitos para fornecer uma especificação completa dos requisitos de software (BITTNER, 2002).

Glossários ou Modelos de domínio - Descrição dos conceitos essenciais do domínio do problema e do ambiente abordado. Essas definições auxiliam e dão base para a construção dos casos de uso, compreensão do contexto do projeto, produção de documentação do usuário. As descrições podem ser em três formas: glossário simples e textual, um modelo de domínio formal ou, um glossário textual com modelo de domínio ilustrativo.

Especificações suplementares - São os requisitos não facilmente capturados no modelo de casos de uso. Requisitos como: restrições legais e regulamentares, padrões de desenvolvimento, atributos de usabilidade, confiabilidade e desempenho, entre outros; não se aplicam muito a casos de uso e perdem valor quando forçados a se enquadrarem na sua estrutura narrativa.

Requisitos declarativos e especiais - Diferente dos requisitos capturados nas especificações suplementares, os requisitos especiais são requisitos adicionais que completam e só fazem sentido dentro da descrição dos casos de uso. A forma mais comum para capturar os requisitos especiais é através de declarações, indicando o que o sistema deve fazer.

2.4 USE CASE 2.0

Em dezembro de 2011, Ivar Jacobson, Ian Spence e Kurt Bittner publicaram o conceito de Use Case 2.0 no livro *USE-CASE 2.0 - The Guide Succeeding with Use Case.* Este novo conceito apresenta uma técnica escalável e ágil para desenvolver requisitos e conduzir o desenvolvimento incremental do sistema (JACOBSON et al., 2011).

Segundo Jacobson et al. (2011), o objetivo dessa nova abordagem é impulsionar o desenvolvimento do sistema, inicialmente, auxiliando as equipes a entender como o sistema será usado para, posteriormente, ajudá-las a evoluir e desenvolver o sistema com valor para o usuário.

Use Case 2.0 é uma conexão entre o conceito de Caso de Uso (descrito na seção 2.3) e a abordagem ágil de desenvolvimento (seção 2.2) utilizando o SCRUM (seção 2.2.2). Os conceitos do caso de uso e da abordagem ágil são os mesmos, porém, a combinação inteligente fornece uma visão geral do produto (JACOBSON et al, 2011).

A abordagem Use Case 2.0 não se refere a uma atualização do conceito de caso de uso, mas sim uma mudança na forma como os desenvolvedores de software e os analistas de negócio aplicam os casos de uso. O conceito de caso de uso não é alterado no Use Case 2.0, mas a forma como é apresentado e gerenciado evoluiu para aumentar a eficácia. (JACOBSON et al., 2011)

2.4.1 Os seis princípios para adoção do Use Case 2.0

A abordagem Use Case 2.0, Jacobson et al. (2011), defini 6 princípios básico para que a aplicação dos casos de uso seja bem sucedida:

Mantenha-se simples contando história - Contar histórias é a forma mais simples e eficaz de transmitir conhecimento a uma pessoa. Um conjunto de casos de uso descreve o objetivo dos atores através de requisitos do sistema. Uma maneira de obter todos esses casos de uso é contando diferentes histórias, mas relacionada, de maneira simples e abrangente. Com isso, os requisitos do sistema podem ser facilmente capturados, compartilhados e compreendidos.

Compreenda o "*the big Picture*" - Compreender o sistema como um todo é essencial para tomar as decisões sobre o que deve e o que não deve ser feito e o custo de cada decisão. Como descrito no Capítulo 2.4 sobre casos de uso, o diagrama de casos de uso apresenta, de forma simples, essa visão macro dos requisitos do sistema.

Concentre-se no valor - O valor de um sistema só é gerado quando o sistema é utilizado. Os casos de uso focam no valor, concentrando-se em como o sistema será utilizado para atingir um objetivo específico para um determinado usuário. Isso é feito através da construção do fluxo básico e dos fluxos alternativos do sistema (Capítulo 2.3.1.1).

Construa sistema em *slices* - Os sistemas descritos pelos casos de uso podem ser muito grandes, tornando os casos de uso demasiado grande para serem entregues em uma única vez. Por isso, é necessário dividir esses casos em porções menores. Uma fatia (*slice*) de caso de uso é uma ou mais histórias selecionadas de um caso de uso, formando um item de valor a ser entregue. As fatias de caso de uso são a parte mais importante da abordagem Use Case 2.0, e serão melhor definidas a seguir (Capítulo 2.4.2).

Entrega em incrementos - Muitos sistemas de software são desenvolvidos/evoluem de maneira incremental. Os casos de uso, utilizando fatias e

diagramas de casos de uso, permitem construir incrementos menores para a construção do sistema. Cada incremento fornece uma versão demonstrável e utilizável do sistema.

Adapte-se às necessidades da organização - O Use Case 2.0 não é técnica fechada, devendo ser adaptada a realidade da organização. A equipe deve decidir qual nível de detalhamento da técnica quer utilizar.

A Use Case 2.0 mistura conceitos de abordagem ágil com conceitos de caso de uso. A Figura 7 mostra como esses conceitos estão relacionados entre si e como mudanças e defeitos os influenciam.

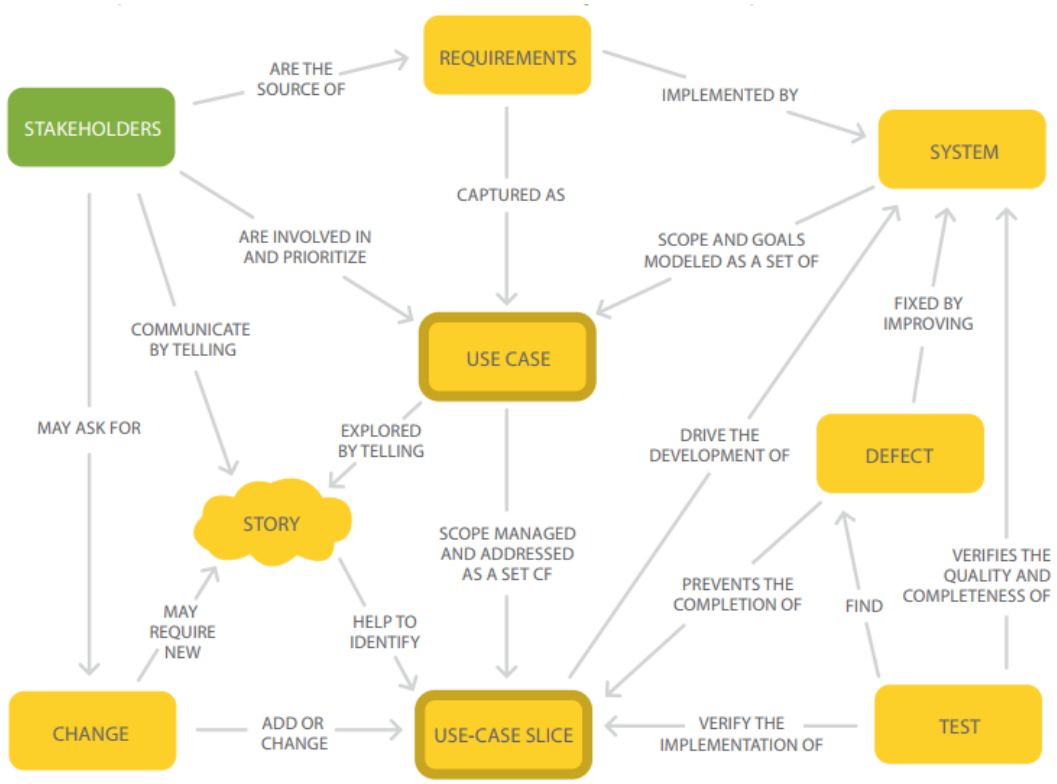


Figura 7 - Mapa conceitual do Use Case 2.0
Fonte: JACOBSON. 2011

A Use Case 2.0 se concentra nos requisitos do usuário, ou seja, no que deve ser desenvolvido para atender os seus objetivos. Os testes comprovam se os objetivos foram ou não atendidos. A base do Use Case 2.0 são os casos de uso, histórias e fatias de caso uso (JACOBSON et al., 2011). Como pode ser observado na Figura 7, os *stakeholders* fornecem os requisitos que são capturados como um conjunto de casos de uso e gerenciados e endereçados como um conjunto de fatias de casos de uso (JACOBSON et al., 2011).

2.4.2 Fatias de casos e uso (*Use Case Slice)*

Grande parte dos sistemas possuem muitos requisitos, sendo que na maior parte das vezes, esses requisitos possuem dependência entres eles (JACOBSON et al., 2011). Segundo Jacobson et al. (2011), um dos grandes erros de desenvolvimento é tentar construir esse tipo de sistema de uma só vez. O sistema deve ser construído em parte, ou fatias, cada uma com um valor claro para o usuário.

O *Use Case Slice*, ou fatias de caso de uso é o agrupamento de uma ou mais histórias selecionadas a partir de um caso de uso para formar um item de trabalho que entrega valor ao cliente. Fatia juntamente com o caso de uso são os conceitos mais importante dos Use Case 2.0. As fatias permitem que os casos de uso sejam divididos em porções menores e entregáveis, fornecendo uma unidade de requisitos adequada ao desenvolvimento e teste. Além disso, as fatias permitem que a equipe se concentre em fornecer um sistema valioso e utilizável o mais brevemente possível, eliminando todos os requisitos desnecessários (JACOBSON et al., 2011).

As fatias de casos de uso, como já dito anteriormente, são um dos conceitos mais importantes do Use Case 2.0, pois são utilizados não apenas como um divisor de narrativa/ histórias, mas também para dirigir o desenvolvimento do sistema. Como mostra a Figura 8, uma fatia de caso de uso possui alguns estágios de mudanças desde a sua identificação inicial até a sua aceitação final. Esses estágios constituem pontos importantes na compreensão, implementação e teste da fatia de caso de uso (JACOBSON et al, 2011):

Figura 8 - O ciclo de vida de uma fatia de caso de uso
Fonte: HAUCK, 2016 - Adaptado

Escopo – Definição do escopo, a extensão da história é esclarecida.

Preparado - Criação das narrativas expandidas. Definição dos casos de teste.

Analisado - Análise dos impactos sobre os componentes de software.

Implementado - Software implementado e pronto para o teste

Verificado - Teste realizado e sistema pronto para ser incluído no release.

Diferente de um sistema em cascata, nessa proposta de fatias de caso de uso, como a fatia é apenas uma parte do sistema a ser implementada, outras atividades de outros ciclos podem estar ocorrendo em paralelo com este (JACOBSON et al., 2011).

2.4.3 Histórias no Use Case 2.0

Histórias são como os casos de uso são explorados com os *stakeholders* do sistema. As histórias são escritas como as narrativas dos casos de uso do sistema, um ou mais fluxos e requisitos especiais, e um ou mais casos de teste. Para descrever as histórias é necessário entender a estrutura das narrativas dos casos de uso. A Figura 9 apresenta a relação entre as histórias e as narrativas de casos de uso (JACOBSON et al, 2011).

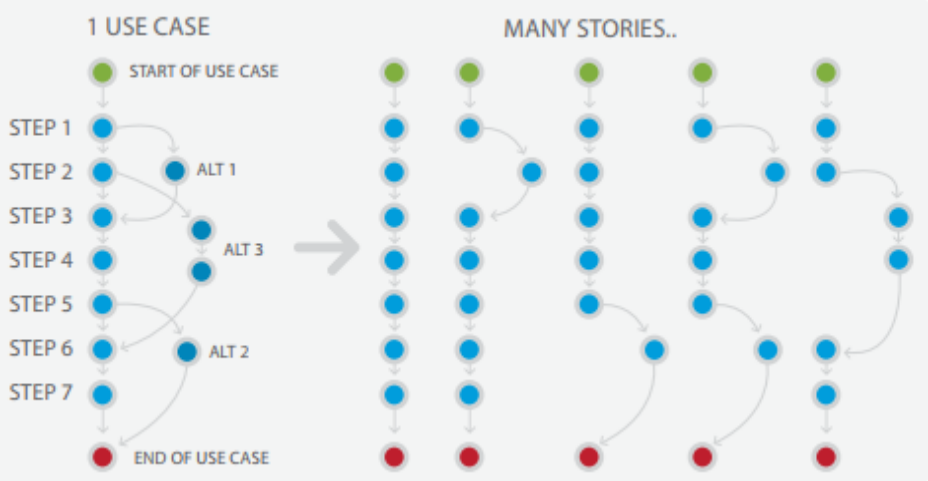


Figura 9 - Fluxo de casos de uso e histórias
Fonte: JACOBSON et al, 2011

Na Figura 9, o fluxo básico é mostrado como uma sequência linear de fatos e os fluxos alternativos, como desvio. Esses fluxos alternativos são vistos sempre como variações sobre o fluxo básico, como mostrado. Cada história percorre um ou mais fluxos, começando sempre pelo início do fluxo básico e terminando no seu final. Essa característica garante que todos os fluxos estão relacionados com o mesmo objetivo (JACOBSON et al, 2016).

As histórias são uma ferramenta para auxiliar a obtenção das fatias de casos de uso e os casos de teste. Cada história possui um ou mais casos de teste, que quando

executados mostram o sistema como é por completo. Os casos de uso e suas fatias impulsionam o desenvolvimento do sistema (JACOBSON, 2016).

2.4.4 Casos de Uso e fatias de casos de uso

Os casos de uso e as fatias de casos de uso precisam ter um rastreamento de estados e um controle de prioridades para que obtenha o maior valor do sistema. Isso pode ser feito de inúmeras maneiras, desde um simples uso de post-its ou planilhas, até ferramentas gerenciais de mudanças e monitoramento de defeitos (JACOBSON et al, 2011).

A Figura 10 apresenta um caso de uso e algumas fatias de casos de uso em um grupo de post-it. O caso de uso apresentado tem como propriedade essencial o seu nome, estado, prioridade e complexidade. Para as fatias de casos de uso, as prioridades essenciais são uma lista de usuário, referência para o caso de uso que define a história, referência aos seus testes e casos de teste e uma estimativa para desenvolver e testar essa fatia de caso de uso (JACOBSON et al, 2011).

Figura 10 - Propriedade de casos de uso e fatias de casos de uso usando *post-it*
Fonte: JACOBSON et al, 2011

Os casos de uso e fatias de casos de uso também devem ser ordenados pelas suas prioridades, de modo que os mais importantes sejam entregues primeiro. A Figura 11 mostra como esses *post- its* podem gerar um *Product Backlog* (JACOBSON et al, 2011).

O primeiro quadro apresenta o diagrama de casos e uso, que contém o escopo do sistema e uma visão completa da primeira versão. O segundo quadro apresenta uma seleção dos casos de uso para a primeira versão de entrega e algumas fatias que foram identificadas mas não ordenadas ou detalhadas. O terceiro quadro mostra a lista de fatias de casos de uso prontas para serem desenvolvidas. E, o último quadro apresenta as fatias desenvolvidas e verificadas com sucesso pela equipe (JACOBSON et al, 2011).

Figura 11 - *Product Backlog* a partir de casos de uso e fatias de casos de uso
Fonte: JACOBSON et al, 2011

2.4.5 Níveis de detalhes do Use Case 2.0

Um dos princípios do Use Case 2.0 relatados na seção 2.4.1, a técnica deve se adaptar à realidade da organização. Seguindo esses preceitos, os casos de uso podem ser definidos em diferentes níveis de detalhe no Use Case 2.0 (HAUCK, 2016).

A Figura 12 apresenta esses níveis de detalhamento definidos para o Use Case 2.0. Como é possível observar os níveis com menor número de detalhes estão no topo da imagem. A quantidade de detalhes vai aumentado à medida que as colunas são melhor definidas e seu conteúdo expandido (JACOBSON et al, 2011).

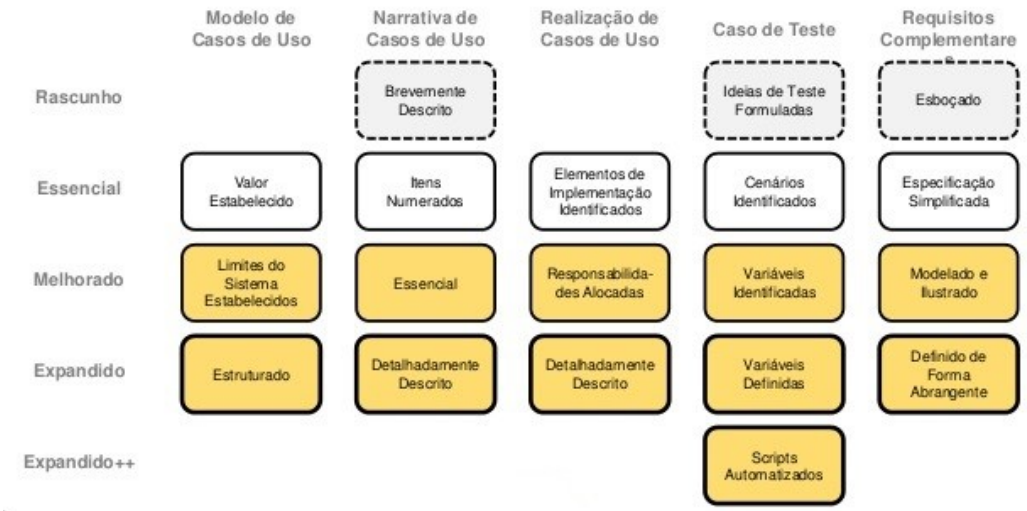


Figura 12 - Níveis de detalhamento dos produtos de trabalho do Use Case 2.0
Fonte: HAUCK, 2016

2.4.6 Use Case 2.0 na prática

O Use Case 2.0 possui algumas atividades essenciais para que os casos de uso forneçam valor ao usuário. A Figura 13 apresenta essas atividades divididas em duas categorias, sendo a primeira responsável por descobrir, ordenar e verificar os requisitos e, a segunda, para formar, implementar e testar fatia por fatia dos casos de uso (JACOBSON et al, 2011).

Encontrar atores e casos de uso - Nesta etapa ocorre a definição de alguns atores e casos de uso que auxiliaram a definir o objetivo do sistema ou definir um novo comportamento, o escopo de liberação do sistema, o valor que o sistema deve agregar, as formas de usar e testar o sistema. A definição dos atores e casos de uso deve ocorrer através de uma modelagem de casos de uso com os *stakeholders* do sistema. O caso de uso deve estar bem delineado para que possa começar o processo de fatia.

Cortar os casos de uso - Após o caso de uso definido, deve-se iniciar o processo de corte. Para isso, é necessário criar itens de tamanho adequado para a equipe de trabalho, limitar-se ao prazo e orçamento, entregar a fatia de maior valor, demonstrar progresso no projeto ou compreensão das necessidades. Os cortes devem ser feitos junto com os *stakeholders* do sistema, a fim de garantir que todos terão valor agregado.

Preparar uma fatia de caso de uso - Ao selecionar uma fatia para ser desenvolvida, é necessário que ela esteja pronta para ser desenvolvida, com as definições de sucesso bem definidas, que os requisitos não funcionais estejam declarados.

Analisar uma fatia de caso de uso - A análise da fatia selecionada e preparada deve ser feita antes do desenvolvimento a fim de compreender quais elementos serão impactados com o desenvolvimento dessa fatia, definir as responsabilidades dos elementos do sistema e quais elementos do sistema interagem com a fatia de caso de uso.

Implementar um caso de uso - Nesta etapa é realizado o projeto, código, teste de unidade, integração dos componentes de software para implementação do sistema.

Teste de uma fatia de caso de uso - Após a implementação, é necessário verificar se a fatia de caso de uso foi implementada com sucesso. Cada fatia precisa ser testada antes de ser considerada como completa e verificada.

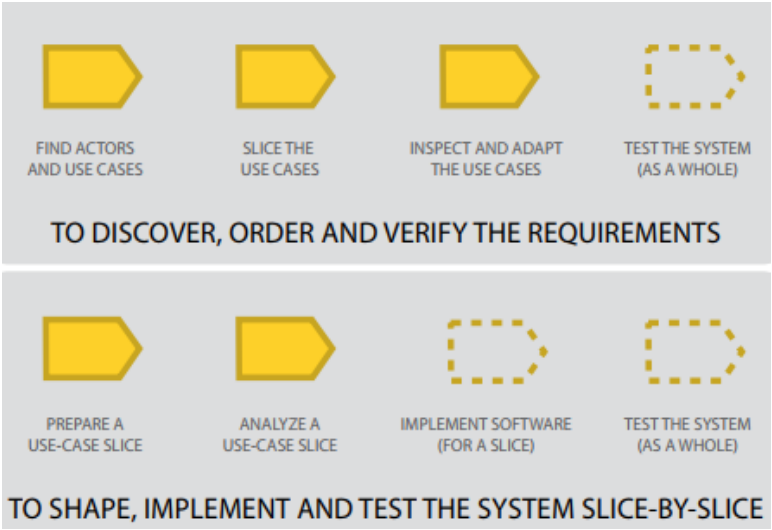


Figura 13 - Atividades do Caso de Uso 2.0
Fonte: JACOBSON et al, 2011

Teste de todo o sistema - Após a fatia de caso de uso ser integrada, é necessário testar para verificar se não ocorreu alguma quebra no sistema.

Inspecionar a adaptar os casos de uso - Ao longo do projeto é necessário revisitar os modelos de casos de uso, casos de uso e fatias de casos de uso para ajudar ou avaliar qualquer mudança ocorrida durante esse período.

3. ESTADO DA ARTE

Neste capítulo é apresentado o estado da arte do tema proposto por este trabalho, através de uma revisão da literatura, seguindo as principais etapas propostas no método definido por Kitchenham (2004), adequando-se a um trabalho de conclusão de curso.

3.1 DEFINIÇÃO DO PROTOCOLO DE REVISÃO

O objetivo desta revisão de literatura é analisar e sintetizar a literatura atual existente sobre a utilização da abordagem Use Case 2.0 para Engenharia de Requisitos, buscando saber quais os resultados observados pelas organizações que utilizam essa abordagem. Portanto, para essa etapa, foi definida a seguinte pergunta:

Quais os resultados na utilização da abordagem Use Case 2.0, para Engenharia de requisitos, aplicada a um ambiente real de desenvolvimento ágil?

Para a tabela de termos de pesquisa (Tabela 1), foram utilizados termos em inglês, devido ao número de publicações em tal idioma nas bases utilizadas.

Tabela 1 - Termos de pesquisa

Termos	**Sinônimos**	**Tradução (inglês)**
Caso de uso 2.0	-	Use case 2.0
Engenharia de Requisitos	Elicitação de requisitos de software, Análise de requisitos de software	Requirements engineering, Software requirements analysis, Software elicitation requirements
Organização	Unidade organizacional, Empresa	Organization, Organizational Unit, Company
Desenvolvimento ágil de software	-	Agile software development

As bases a serem pesquisadas são o Google Scholar[1], por ser de acesso livre e possuir resultados amplos, o portal de pesquisas da CAPES[2], por reunir e disponibilizar à Universidade Federal de Santa Catarina o melhor da produção científica internacional (CAPES/MEC, 2015), e o IEEE Xplore[3], por oferecer as publicações científicas e técnicas mais citadas e referenciadas mundialmente na área de Engenharia Elétrica, Ciências da Computação e Eletrônica (IEEE XPLORE, 2015).

3.1.1 Critérios de inclusão e exclusão

Com o intuito de incluir o maior número de estudos relevantes que corroborem a responder a pergunta de pesquisa, são definidos os seguintes critérios de inclusão:

- O material encontrado deve apresentar as experiências práticas de utilização da abordagem Use Case 2.0.
- O material encontrado deve apresentar a experiência de uma organização com foco em desenvolvimento ágil de software.
- O material encontrado deve apresentar os resultados observados na aplicação da abordagem Use Case 2.0.

Os critérios de exclusão são definidos visando remover publicações irrelevantes ao contexto da pesquisa. Com isso, são definidos os seguintes critérios de exclusão:

- O material encontrado que apresentar a experiência de uma organização com foco em desenvolvimento tradicional.
- O material duplicado.
- O material que não permitir acesso a todo o seu conteúdo.

3.1.2 Critérios de qualidade

Os estudos encontrados nesta revisão de literatura devem atender aos seguintes critérios de qualidade:

- Apresentar o modo como a abordagem Use Case 2.0 foi aplicada.
- Preferencialmente incluir lições aprendidas.
- Preferencialmente incluir a caracterização da organização.

[1] https://scholar.google.com.br
[2] http://www.periodicos.capes.gov.br/
[3] https://ieeexplore.ieee.org/Xplore/home.jsp

- Preferencialmente apresentar as dificuldades encontradas.
- Apresentar os resultados positivos ou negativos na utilização da abordagem Use Case 2.0.

3.2 EXECUÇÃO DA BUSCA

A execução da busca foi realizada em Outubro e Novembro de 2017, utilizando combinações dos termos de buscas presentes na Tabela 1. Foi realizada as buscas com os termos em Inglês (*string* de busca) descritos na Tabela 1. Foram obtidos 86 resultados no total, e todos foram analisado.

Primeiramente, foi executada a busca utilizando a primeira *string* apresentada na Tabela 2. Como não foram encontrados nenhum estudo potencialmente relevante, a *string* foi então modificada para tornar-se mais genérica, a fim de ampliar as buscas para que outros trabalhos fossem apresentados.

A abordagem Use Case 2.0 é muito nova e por isso, acredita-se que não tenham sido encontrados trabalhos que pudessem que se enquadrar nos critérios de inclusão. Sendo assim, foram considerados estudos que utilizassem a abordagem Use Case, que é a base para a abordagem Use Case 2.0.

Foram lidos todos os títulos e resumos de todos os trabalhos retornados e, após a aplicação dos critérios de inclusão, exclusão e qualidade apenas 3 artigos foi considerado relevante ao final da busca

Tabela 2 - *Strings* de busca

String de busca	**Encontrados**	**Analisados**	**Potencialmente relevantes**	**Relevantes**
("Use case 2.0") AND ("Requirements engineering" OR "Software requirements elicitation" OR "Software requirements analysis") AND ("Company" OR "Organization" OR "Organizational Unit") OR ("Agile software development")	1	1	0	0

("Use case 2.0") AND ("Company" OR "Organization" OR "Organizational Unit") OR ("Agile software development")	85	85	16	3

3.3 EXTRAÇÃO DAS INFORMAÇÕES E ANÁLISE DOS RESULTADOS

Utilizando as *strings* e ferramentas de busca definidas, não foram encontrados resultados relevantes que se enquadrassem nos critérios de aceitação. Foram realizadas inúmeras alterações e iterações de busca, com o intuito de encontrar qualquer tipo de material que apresentasse alguma experiência com a abordagem Use Case 2.0, porém, sem resultados.

Acredita-se que isso ocorre pois, o tema proposto é relativamente recente e ainda não foram realizados ou publicados experimentos que utilizassem a abordagem Use Case 2.0. Outro ponto que pode ter contribuído para a falta de material sobre o tema proposta é a limitação no idioma em que foram feitas as buscas, apenas em inglês e português, e também a limitação nas bases de dados utilizadas e na *string* proposta.

3.4 DISCUSSÕES

Apesar de não ter sido encontrado nenhum trabalho relacionados segundo Sommerville (2011), a abordagem de casos de uso é uma das principais técnica de elicitação e descrição de requisitos e isso também foi observado na leitura dos trabalhos, pois inúmeras instituições utilizavam a abordagem, porém, estas não relataram a experiência com a utilização dos casos de uso e não utilizaram a abordagem Use Case 2.0.

3.5 AMEAÇAS À VALIDADE

Após a execução desta revisão sistemática de literatura, notou-se não foram encontradas publicações relevantes à busca que atendessem a todos os critérios de qualidade definidos, acarretando em ameaças a validade desta revisão.

Além disso, a possível imprecisão dos termos de busca utilizados e seus sinônimos pode ter omitido resultados e, assim, impedido a coleta de publicações relevantes à esta revisão de estado da arte.

Assim como, pelo fato de terem sido lidos somente os títulos e resumos dos trabalhos considerados, títulos ou resumos inadequados podem ter ocultado materiais significativos. Por fim, como esta revisão de literatura foi realizada somente pela autora do trabalho, não houve revisão direta dos trabalhos selecionados. Portanto, a seleção dos materiais relevantes coube somente à opinião da autora.

4 PROCESSOS ATUAL

Nesta seção é apresentado o processo atual de desenvolvimento do projeto SISMOB do laboratório Bridge, iniciando pela definição do contexto em que o projeto está inserido. Nesta primeira parte é apresentado também o produto produzido pelo projeto e como ocorre a definição dos produtos a serem desenvolvidos. Em seguida, é apresentado às equipes em que serão realizados o estudo de caso e, é definido o processo atual de desenvolvimento do projeto. Ao final, é apresentado o processo de desenvolvimento de software das equipes do estudo de caso.

4.1 CONTEXTO

O laboratório Bridge atua na pesquisa e desenvolvimento de softwares voltados a gestão da saúde pública. Criado em 2012, foi instituído oficialmente em 2016 e está ligado ao Centro Tecnológico da Universidade Federal de Santa Catarina (BRIDGE, 2017).

Atualmente, o Bridge tem como principal cliente o Ministério da Saúde. O quadro de colaboradores conta com aproximadamente 120 colaboradores divididos entre os projetos desenvolvidos, e os núcleos Administrativo, Gestão de pessoas, Design, Teste exploratório e Teste automatizado, sendo esses núcleos compartilhados entre todos os projetos. Além disso, o laboratório possui um Coordenador geral, Coordenador de projetos e um Gerente de produtos (BRIDGE, 2017).

Os principais projetos desenvolvidos pelo Bridge são (BRIDGE, 2017):

- e-SUS AB - “Informatização do processo de trabalho da Atenção Básica através do Prontuário Eletrônico do Cidadão”.
- SISMOB - “Sistemas de monitoramento de obras. Sistema para a gestão do financiamento e execução das obras financiadas pelo Ministério da Saúde. Tais obras compreendem a construção, reforma e ampliação de estabelecimentos de saúde”.
- RNI - “Registro nacional de implantes. Sistema que auxilia no controle de qualidade dos componentes implantáveis e regulação econômica do mercado, além de realizar os registros dos métodos utilizados para a implantação e a rastreabilidade dos produtos utilizados. ”.

Dentre os projetos citados, o SISMOB foi o escolhido para aplicação desse estudo de caso por ser o projeto em que a autora deste estudo trabalha como analista de software.

O Sistema de monitoramento de obras é um software que tem como objetivo monitorar todas as obras de engenharia e infraestrutura de estabelecimentos de saúde que são financiados com recursos federais, cuja transferência ocorra fundo a fundo, se

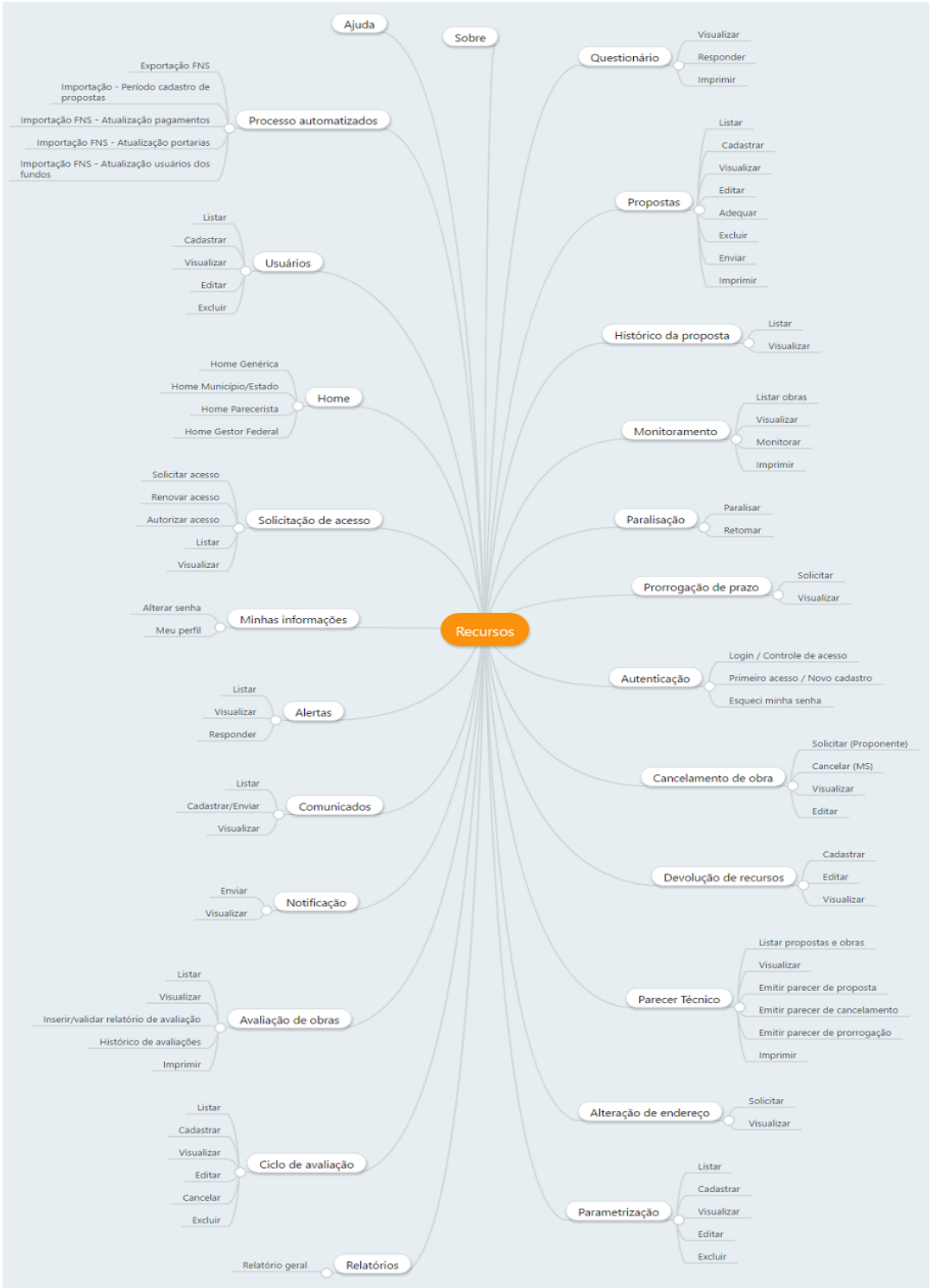


Figura 14 - Mapa mental dos recursos desenvolvidos no projeto SISMOB

tornando uma ferramenta para o gerenciamento de todas as fases da obra. Para atender esse objetivo, o SISMOB possuí inúmeros recurso como mostra a Figura 14.

Atualmente, o projeto conta com 24 funcionários, divididos em 3 times de desenvolvimento, sendo um Gerente de projeto, 4 analistas de sistemas, 9 analistas de teste, 9 programadores e 1 designer. O SISMOB conta também com o apoio de um Grupo de Trabalho (GT), que faz o papel de *stakeholder* do projeto. O GT é formado por 3 colaboradores do Ministério da Saúde e fazem sua atuação em Brasília. Os integrantes do GT têm amplo acesso e conhecimento sobre as necessidades do cliente e, possuem um vasto conhecimento nas regras de negócio do SISMOB.

O desenvolvimento do SISMOB é realizado por meio de termo de execução descentralizada (TED), que serve como instrumento para a realização de projetos governamentais. Nos TED's são definidos os produtos a serem entregues ao cliente. O processo de desenvolvimento desses produtos, gestão e controle de manutenções corretivas e evolutivas do software é mantido através da ferramenta redmine[4]. Utilizada por todos os membros do projeto e pelo *stakeholder*, nesta ferramenta é definido, pelo Gerente de projeto em conjunto com o *stakeholder*, o que será entregue em cada versão e por qual produto cada equipe ficará responsável. Para esse estudo de caso, foram selecionadas duas equipes. A equipe "D" constituída de 6 integrantes no time de desenvolvimento que ocupam os seguintes cargos:

Tabela 3 - Time de desenvolvimento equipe "D"

Cargo	**Formação**	**Experiência**	**Tempo de projeto**
Analista de sistemas	Graduanda em Ciências da Computação	3 anos	3 anos
Desenvolvedor	Bacharel em Ciências da Computação	6 anos	4 anos
Bolsista de desenvolvimento	Graduando em Sistema de Informação	3 anos	2 anos
Bolsista de desenvolvimento	Pós-graduando em Ciências da Computação	4 anos	3 anos

[4] https://redmine.sismob.ufsc.br

Analista de qualidade	Graduando em Sistema de Informação	2 anos	2 anos
Bolsista em análise de qualidade	Graduando em Sistema de Informação	1 ano	1 anos

A equipe “F” constituída de 5 integrantes no time de desenvolvimento que ocupam os seguintes cargos:

Tabela 4 - Time de desenvolvimento equipe "F"

Cargo	**Formação**	**Experiência**	**Tempo de projeto**
Analista de sistemas	Bacharel em Ciências da Computação	8 anos	5 anos
Desenvolvedor	Bacharel em Ciências da Computação	4 anos	3 anos
Bolsista de desenvolvimento	Graduando em Sistema de Informação	3 anos	3 anos
Analista de qualidade	Graduando em Sistema de Informação	2 anos	2 anos
Bolsista em análise de qualidade	Graduando em Sistema de Informação	1 ano	1 anos

Além disso, as equipes compartilham um designer que atende a todo o projeto, um *Product Owner*, papel desempenhado pelo Gerente de projeto e, o papel de *Scrum Master* é desempenhado pelo analista de sistemas de cada equipe.

4.2 PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento executado pelas equipes que realizaram o estudo tem como base os princípios da metodologia SCRUM, descrita no capítulo 2.2.1 deste documento. O Scrum possui uma série de valores, conceitos e práticas que tem como objetivo maximizar as chances de sucesso do projeto (PRESSMAN, 2011). Porém, por ser um framework adaptativo (SCHWABER, 2017) os projetos que utilizam o SCRUM não são obrigados a realizar todos os conceitos e técnicas.

No início de cada sprint, cada time de desenvolvimento recebe um conjunto de tarefas selecionadas pelo gerente de projeto, que este julga serem possíveis de serem durante a sprint. Essas tarefas podem ser de três tipos: (i) novo produto, (ii) manutenção evolutiva e (iii) manutenção corretiva. O time de desenvolvimento avalia essas tarefas e define quais destas realmente são possíveis de serem implementadas, iniciando a etapa de planejamento da sprint.

Para definir e incluir as tarefas na sprint, é realizado uma estimativa de esforço baseando-se no tamanho e complexidade da análise, desenvolvimento e teste da tarefa. Para isso, é utiliza a técnica de *Planning Poker*, que baseada em consenso e histórico da equipe, auxilia na estimativa de esforço para o desenvolvimento de um produto (COHN, 2005). Com base no histórico da quantidade de pontos que foram concluídos em sprints passadas é possível abstrair a velocidade com que o time produz e determinar quantidade de itens que serão incluídos na sprint.

O conjunto de tarefas selecionadas constitui o Backlog da Sprint. Assim que o processo de planejamento é encerrado e o backlog da sprint é construído, inicía-se a etapa de execução da sprint. Diariamente o time de desenvolvimento realiza uma *Daily Scrum*, para verificar se o andamento está seguindo conforme o planejado. No início de cada semana, é realizada também uma reunião de *Scrum of Scrums*, em que um representante de cada equipe, apresenta para o *Product Owner* o que foi feito na sprint e o que ainda falta ser realizado, verificando assim, se o desenvolvimento está de acordo com o planejado.

Ao final da sprint, é realizada então uma homologação por parte dos *stakeholders* sobre os produtos entregues, a fim de verificar se estão de acordo com os requisitos levantados. Também é realizado reunião de *Sprint Retrospective*, na qual o time faz uma análise do decorrer da sprint para determinar aspectos do processo que precisam ser melhorados.

4.2.1 Processo dos times do estudo de caso

O processo do time de desenvolvimento[5] é dividido em duas partes, planejamento e execução (Figura 15). Com duração de duas semanas, ao final da sprint, as tarefas do *Sprint Backlog* devem estar implementadas e testadas pelo time de desenvolvimento e, caso haja tarefas de manutenção evolutiva e novo produto para a próxima sprint, essas devem estar devidamente refinadas e especificadas pelo analista de sistemas do time.

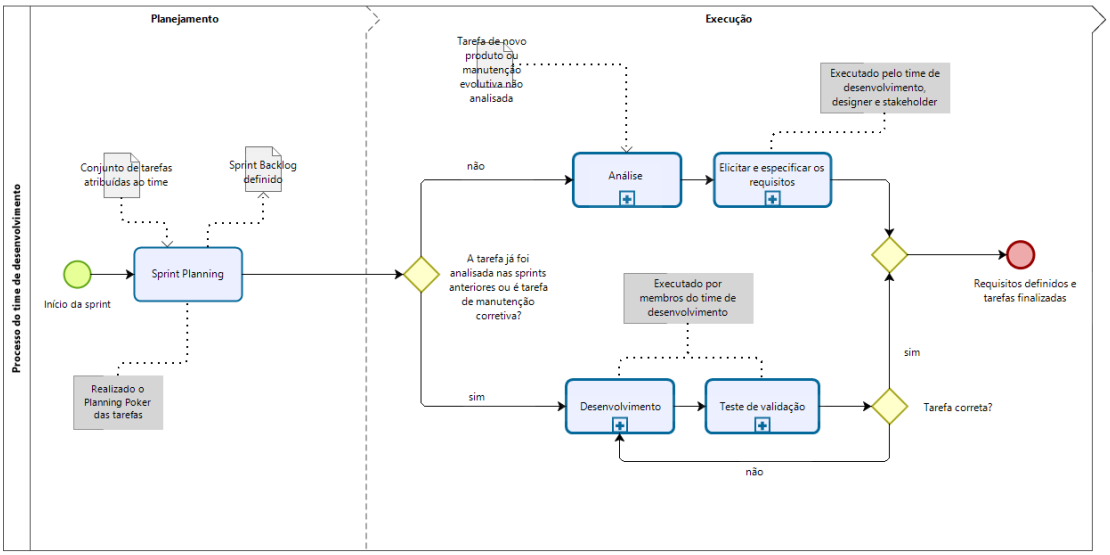


Figura 15 - Processo de desenvolvimento das equipes do estudo

Na etapa de planejamento da sprint, é realizada a *Sprint Planning Meeting,* uma reunião onde são apresentadas as tarefas da sprint atribuídas ao time de desenvolvimento. Para definição e inclusão de cada tarefa na *Sprint Backlog* é realizado uma verificação por tipo de tarefa. Para as tarefas de novo produto e manutenção evolutiva, se já houve a *análise* nas sprints anteriores, essas serão desenvolvidas na sprint que está sendo planejada. As tarefas de manutenção são analisadas durante a reunião de *Sprint Planning* e seu desenvolvimento ocorre na sprint que está sendo sendo planejada. Após a verificação, é realizado o *Planning Poker* de todas as tarefas e, com base na pontuação das sprints anteriores, é definido o *Sprint Backlog*. Com isso, o processo de planejamento

[5] Os processos dos dois times de desenvolvimento selecionados são similares, por isso, houve uma abstração e será feita apenas uma descrição, destacando as diferenças no decorrer do texto.

é encerrado, o *taskboard* (Figura 16) do time é construído e dá-se início ao desenvolvimento dos itens selecionados. Para a construção do *taskboard*, a equipe "D" utiliza a ferramenta Trello[6] e a equipe "F" utiliza a ferramenta Asana[7].

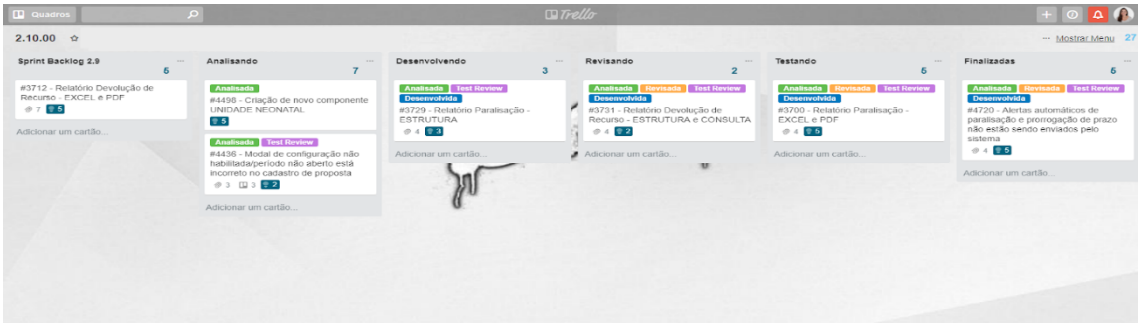


Figura 16 - *Taskboard* da equipe "D" do estudo de caso
Fonte: Trello da equipe "D"

Ao final da etapa de planejamento, o *Sprint Backlog* deve conter dois tipos de tarefas. As tarefas a serem desenvolvidas (tarefas de novo produto e manutenção corretiva analisadas nas sprints anteriores e, tarefas de manutenção corretiva) e, tarefas que devem passar pelo processo de análise. Esses dois processos, análise e desenvolvimento devem ocorrer de forma paralela durante a execução da sprint.

O processo de *análise,* é realizado quando a tarefa é um novo produto/módulo do sistema ou, quando é uma evolução de um produto/módulo já existente. Nesse processo é feito, primeiramente, a análise da tarefa e é realizada a elicitação e especificação dos requisitos pelo analista de sistemas do time de desenvolvimento em conjunto com o *stakeholder*. Em paralelo é realizado também, pelo design do projeto em conjunto do analista de sistemas, a construção dos protótipos de alta fidelidade. Após a definição dos protótipos e especificação dos requisitos é feita a avaliação de impacto no sistema ao desenvolver a tarefa. Essa avaliação é feita em conjunto com todos os membros do time de desenvolvimento e o designer do projeto. Ao final, é construído um documento contendo todas os requisitos funcionais, regras de negócio, protótipos de tela e análise de impacto.

Na primeira parte do processo de análise é feito refinamento da tarefa. Para isso o analista de sistemas entra em contato com o *stakeholder* do projeto para entender melhor o que se espera desse novo produto/módulo. Para isso, é realizado uma entrevista com

[6] https://trello.com/
[7] https://asana.com/

stakeholder a fim de obter um maior entendimento sobre o que se deseja obter com essa nova funcionalidade e elicitar os requisitos funcionais e possíveis regras de negócio. Essa entrevista é feita ou por videoconferência ou presencial.

Com entendimento de todo o escopo do novo produto/módulo ou da sua evolução, é iniciado a especificação dos requisitos. Essa especificação é feita através da definição dos requisitos funcionais e regras de negócio. Os requisitos funcionais são compostos de 5 partes (Figura 17, 18 e 19): (i) campos a serem apresentado no módulo, (ii) as regras desse novo módulo, (iii) um dicionário de dados para os campos apresentados, (iv) protótipos de tela e (v) análise de impacto. Já as regras de negócio são definidas pelo s*takeholder* e contém, entre outras coisas, como deve ocorrer a comunicação entre o sistema e os sistemas externos. Durante a especificação dos requisitos, o analista de sistemas fica em comunicação direta com o *stakeholder*. É durante esse processo, que o analista de sistema inicia a construção de um *mockup* de baixa fidelidade. Para a descrição dos requisitos é utilizada a linguagem Markdown (GRUBER, 2004) e, a criação dos mockups é feita através da ferramenta moqups[8].

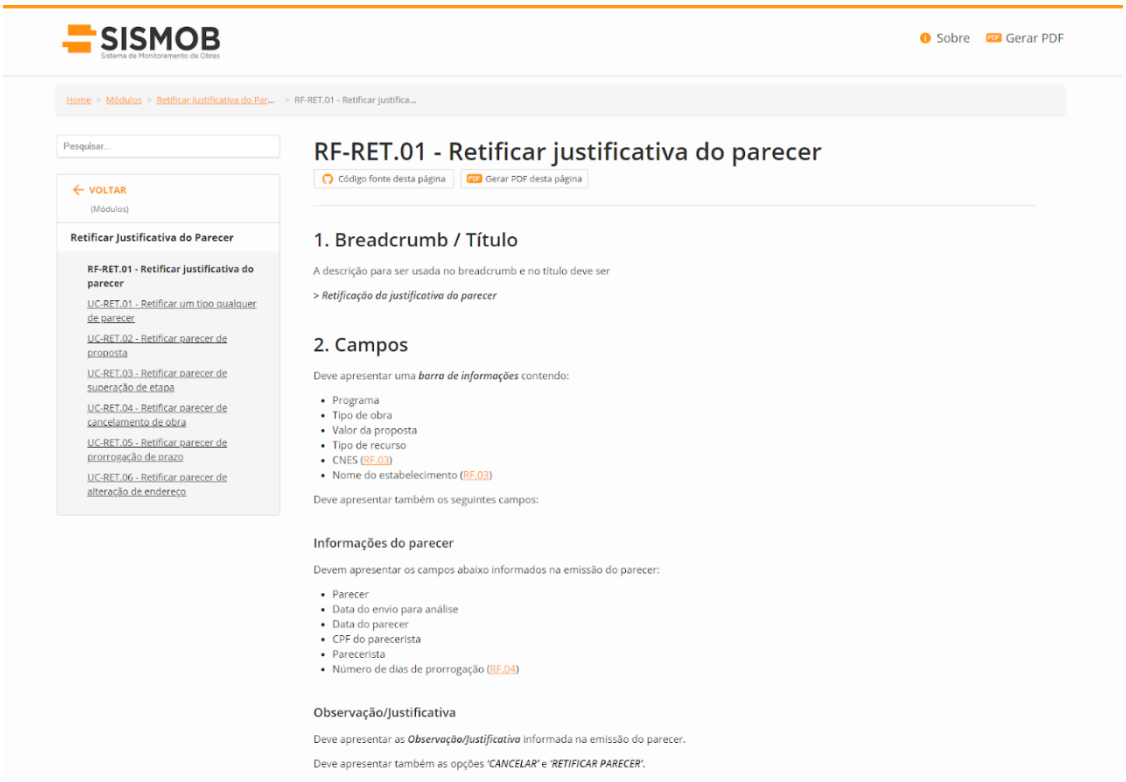


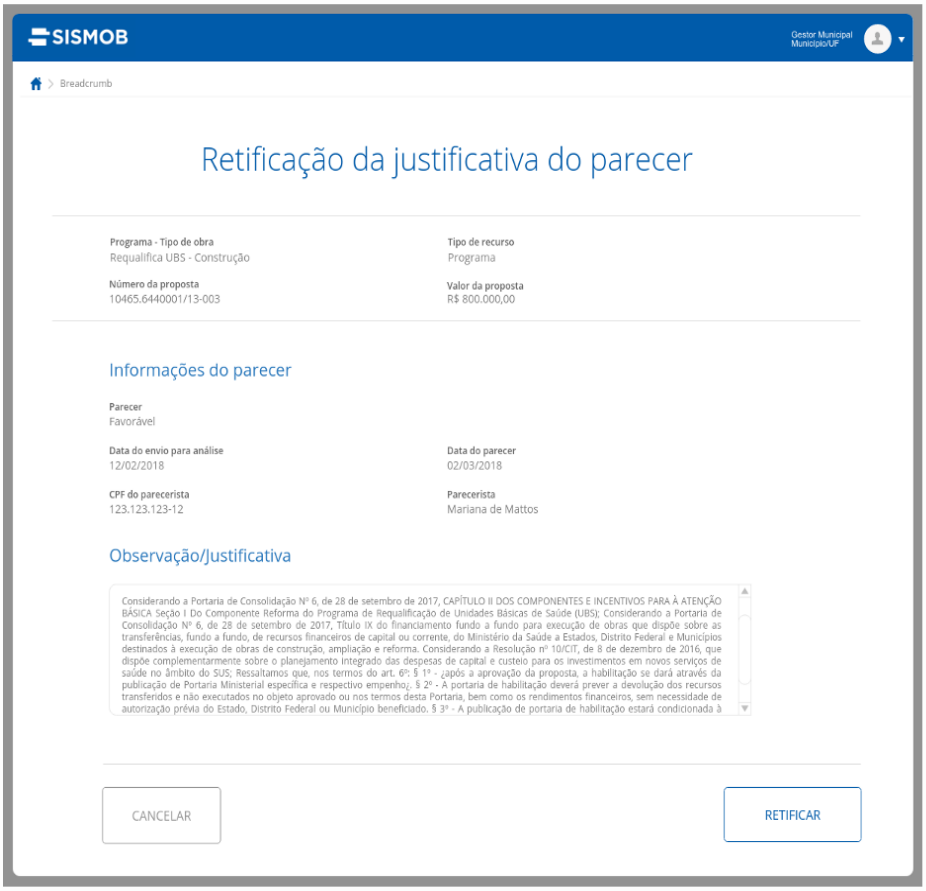
Figura 17 - Requisitos funcionais - Parte 1

[8] https://app.moqups.com/

4. Dicionário de dados

Nome do campo	Obrigatório	Tamanho Mínimo	Tamanho Máximo	Tipo	Observações
Observação/Justificativa	sim	1	4000	Alfanumérico	

5. Protótipo de tela



Protótipo 1 - Retificar parecer

6. Análise de impacto

1. Qual alteração na retificação da justificativa de um parecer impacta na RF-PTE.06 - Visualizar Parecer, RN-FNS.05 - Salvar Parecer (FNS) e RN-HIS.01 - Histórico de ações da proposta.

Figura 19 - Requisitos funcionais - Parte 3, 4 e 5

Ao finalizar a etapa de especificação de requisito e, com o *mockup* de baixa fidelidade definido, o analista de sistemas entra em contato com o designer do projeto para a definição do protótipo de tela em alta fidelidade. Caso o designer do projeto defina um fluxo de tela ou componente diferente do já existentes no sistema, é realizado uma

reunião com o time de desenvolvimento para avaliar a complexidade na construção desse novo componente. Por fim, com a especificação de requisitos finalizada e documentada, a análise de impacto realizada e os protótipos de tela construídos, o documento gerado é enviado ao *stakeholder* para validação e revisão das regras e fluxos de sistema. Neste momento, o analista de teste da equipe inicia a construção dos testes de validação. O processo de análise é finalizado quando o *stakeholder* valida o documento de requisitos.

O processo de desenvolvimento inicia quando o *Sprint Backlog* já está definido. As tarefas são dispostas no *taskboard* e são desenvolvidas de acordo com a ordem definida pelo analista de sistemas. Para as tarefas de manutenção corretiva, o desenvolvedor apenas verifica junto ao analista de sistemas se a correção sugerida é realmente válida e, caso seja, ele inicia o desenvolvimento da mesma. Já para tarefas evolutivas e de novo produto é feita uma divisão entre os desenvolvedores, sem muito critério. É avaliado o documento de requisitos e cada desenvolvedor fica responsável por uma parte. Porém, por não ter um processo definido de divisão das tarefas evolutivas, muitas vezes não fica claro o que já foi desenvolvido e o que ainda falta desenvolver de um novo produto/módulo. Ao finalizar o desenvolvimento de uma tarefa, ou parte de uma tarefa, é feita uma revisão do código por outro desenvolvedor. O processo de desenvolvimento é finalizado quando a tarefa é aceita pelo teste de validação.

O processo de teste de validação é feito quando o desenvolvimento alcançou algum valor para ser testado. Para tarefas de manutenção corretiva, o teste verifica, através de um teste exploratório, apenas o cenário descrito na tarefa. Já para tarefas de novo produto/módulo ou manutenção evolutiva, o analista teste da equipe utiliza o *teste review*, desenvolvido na etapa de análise, e verifica através de um teste exploratório, se todos os critérios foram alcançados. Como, muitas vezes, o que está sendo testado não é toda a tarefa, o analista de teste precisa ficar em comunicação direta com o desenvolvedor para saber o que foi desenvolvida e o que ainda não deve ser testado. Caso exista alguma inconsistência, o analista de teste verifica com o analista de sistemas qual deve ser a regra correta. Caso o erro esteja na documentação de requisitos, a tarefa retorna para o processo de análise, para que possa ser corrigido na documentação de requisito. Porém, caso o erro esteja no sistema, o analista de teste retorna a tarefa para que o desenvolvedor possa fazer as correções.

5 PLANEJAMENTO E DESENHO DO ESTUDO DE CASO

Neste capítulo é realizado um estudo de caso utilizando os passos principais da abordagem para estudos de caso em Engenharia de software proposto por Runeson (2009), em *Guidelines for conducting and reporting case study research in software engineering*. Inicialmente, é realizada a definição do estudo de caso e, logo após, a execução do mesmo.

Para Runeson (2009), a realização de um estudo de caso possui 5 passos principais a serem seguidos, sendo eles: Planejamento, Preparação, Coleta, Análise e Relatório. Na etapa de Planejamento, os objetivos do estudo de caso são definidos e sua execução planejada. Na Preparação, são definidos os procedimentos de coleta de dados. Na etapa de Coleta, o estudo de caso é executado e seus dados são coletados. Por fim, nas etapas de Análise e Relatório, a análise dos dados coletados é feita e um relatório é gerado a partir dos dados coletados (RUNESON, 2009).

5.1 OBJETIVOS DO ESTUDO DE CASO

Como definido no Capítulo 1.1, o objetivo deste trabalho é implantar e avaliar a abordagem Use Case 2.0 no projeto SISMOB do Laboratório Bridge. A partir desse objetivo, a seguinte pergunta de pesquisa para este estudo de caso foi definida: "*Como a aplicação das técnicas de Use Case 2.0 impactam na análise, desenvolvimento e teste de um time de desenvolvimento de software?".* Segundo LINGS e LUNDELL (2005) avaliação de impacto é a análise das consequências de alguma modificação efetuada em algum processo ou organização. Para esse estudo, o impacto foi derivado em termos de: (i) esforço despendido pelas equipes do estudo, (ii) produtividade das equipes em que o estudo está sendo aplicado e (iii) custo-benefício da implantação da abordagem aplicada no estudo.

Seguindo o método de pesquisa definido no capítulo 1.2, após a revisão da literatura e definição do estado da arte conforme descrito nos capítulos 2 e 3, deve ser realizada a aplicação do estudo de caso proposto. Para isso, conforme Figura 20, primeiramente é realizada uma análise do contexto em que a organização está inserida, conforme definido no capítulo 4. Em seguida, foi elaborado o planejamento do estudo de caso utilizando a abordagem GQM - *Goal/Question/Metric* (KOZIOLEK, 2008). Em seguida é definido com as equipes do estudo uma estratégia de implantação da abordagem Use Case 2.0. Então, foi feita a aplicação da abordagem Use Case 2.0 e realizado o acompanhamento das equipes em que a abordagem está sendo aplicada para verificação do impacto e análise dos resultados. Em paralelo, dados são coletados sobre a realização do estudo de caso.

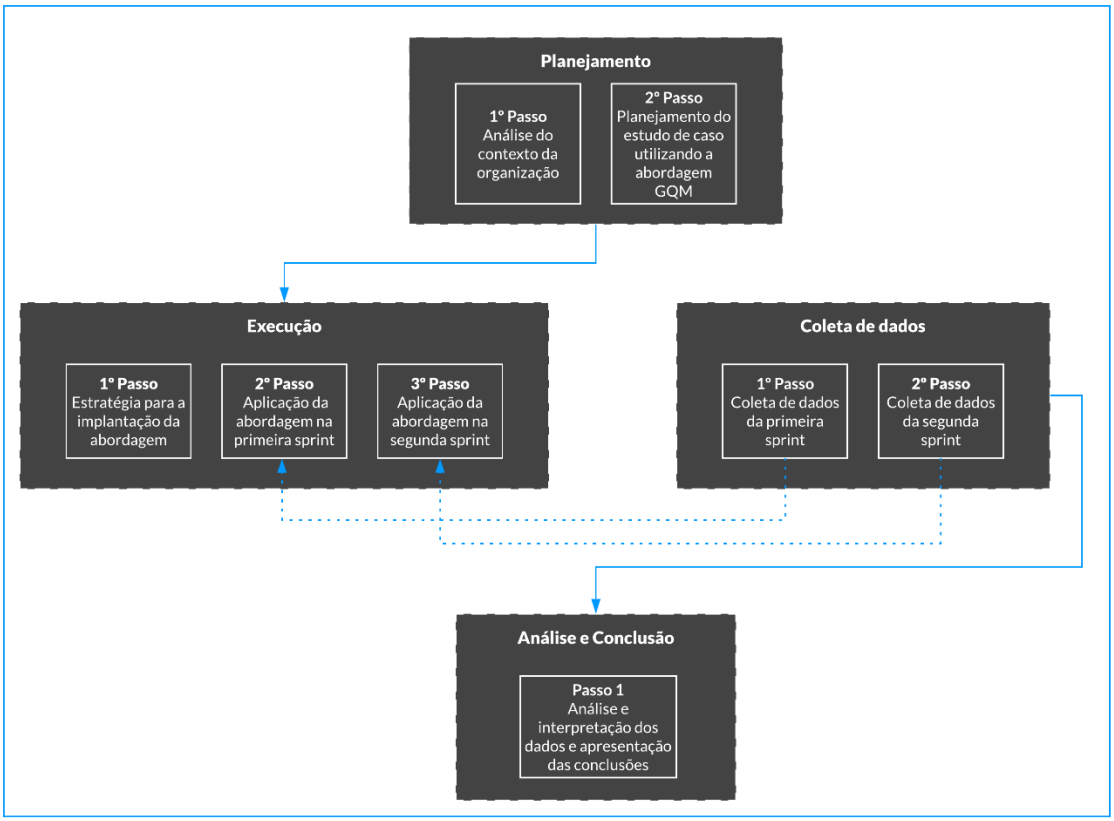


Figura 20 - Passos para realização do estudo de caso

A execução desse estudo de caso se dá no período entre Fevereiro a Maio de 2018. Foram selecionados dois times de desenvolvimento do projeto SISMOB. O time "D" é composto por 6 membros e, o time "F" por 5 membros. Além disso, os times contam com

o suporte de um *Scrum Master*, papel desempenhado pelo analista de sistemas de cada time e, com um *Product Owner*, papel desempenhado pelo gerente de projeto. A contextualização da organização, do projeto e dos times de desenvolvimento estão descritos no capítulo 4.

Para ser feita a avaliação de um estudo de caso, é necessário primeiramente definir seus objetivos de medição. Para a definição dos objetivos de medição deste trabalho, foi utilizada a abordagem GQM (KOZIOLEK, 2008). Essa abordagem auxilia na definição dos objetivos de medição do trabalho, além de contribuir na derivação das medidas a serem coletadas no decorrer do estudo de caso.

Na abordagem GQM (KOZIOLEK, 2008), medidas são definidas de maneira *top-down*, ou seja, primeiro os objetivos são definidos, para que então perguntas sejam estabelecidas. Por último, medidas são definidas de forma a prover dados que satisfaçam as necessidades de informação. Ao definir os objetivos de antemão, apenas medidas relevantes são escolhidas, tornando o procedimento de coleta de dados mais eficiente (KOZIOLEK, 2008; BASILI, 1994).

5.1.1 Definição dos objetivos

Os objetivos descritos a seguir serão estabelecidos tendo em mente o objetivo deste estudo de avaliar o impacto, em termos de esforço, produtividade e custo-benefício, da aplicação da abordagem Use Case 2.0. Para isso, o termo custo-benefício foi derivado em termos de (i) dificuldade para a aplicação da abordagem Use Case 2.0; (ii) aceitação para a aplicação da abordagem Use Case 2.0. Com isso, foram definidos os seguintes objetivos de medição:

Objetivo de Medição 1 - Analisar o esforço e a produtividade da equipe na aplicação das técnicas de Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

Objetivo de Medição 2 - Analisar as dificuldades para aplicação das técnicas da abordagem Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

Objetivo de Medição 3 - Analisar a aceitação das técnicas da abordagem Use Case 2.0 utilizadas sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

5.1.2 Perguntas e medidas

Segundo a abordagem GQM (KOZIOLEK, 2008), após a determinação dos objetivos de medição, deve-se estabelecer perguntas e medidas para cada objetivo. No entanto, embora tenha havido esforço para se quantificar os critérios de medição em termos quantitativos, foi observado que muitos dados a serem coletados a fim de suprir a necessidade na informação são qualitativos. A Tabela 9 apresenta as perguntas e medidas do objetivo 1:

Tabela 5 - Perguntas e medidas do objetivo 1

Objetivo de Medição 1	Analisar o esforço e a produtividade da equipe na aplicação das técnicas de Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.
Pergunta 1.1	Qual o esforço total despendido pelas áreas de análise, desenvolvimento e teste para aplicação da abordagem Use Case 2.0?
Medição 1.1	Esforço total em horas
Pergunta 1.2	Houve aumento na produtividade na equipe ao aplicar a abordagem Use Case 2.0?
Medição 1.2	Diferença percentual entre a estimativa de esforço da equipe ao aplicar a abordagem Use Case 2.0 em relação a estimativa de esforço da mesma equipe antes da aplicação da abordagem.
Pergunta 1.3	Quanto esforço cada área demandou na aplicação da abordagem Use Case 2.0?
Medição 1.3	Proporção de esforço despendido em cada área em relação ao total de horas despendido para a aplicação da abordagem Use Case 2.0.

Na Tabela 10 são apresentadas as perguntas e medidas referentes ao objetivo 2:

Tabela 6 - Perguntas e medidas do objetivo 2

Objetivo de Medição 2	Analisar as dificuldades para aplicação das técnicas da abordagem Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.
Pergunta 2.1	É fácil de compreender e utilizar a abordagem Use Case 2.0?
Medição 2.1	Impressão subjetiva das principais técnicas da abordagem Use Case 2.0
Pergunta 2.2	A experiência da aplicação da abordagem foi benéfica a unidade organizacional?
Medição 2.2	Impressão subjetiva da experiência com a aplicação da abordagem Use Case 2.0
Pergunta 2.3	Quais foram as principais dificuldades encontradas na aplicação da técnica Use Case 2.0?
Medição 2.3	Impressão subjetiva da experiência com a aplicação da abordagem Use Case 2.0

A Tabela 11 apresenta as perguntas e medidas relacionadas ao objetivo 3:

Tabela 7 - Perguntas e medidas do objetivo 3

Objetivo de Medição 3	Analisar a aceitação das técnicas da abordagem Use Case 2.0 utilizadas sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.
Pergunta 3.1	A equipe pretende continuar utilizando a abordagem Use Case 2.0? Se sim, porquê?
Medição 3.1	Impressão subjetiva sobre a aplicação da abordagem na equipe.
Pergunta 3.2	Quais foram as principais técnicas da abordagem Use Case 2.0 utilizada que obtiveram maior aceitação por parte das áreas de análise, desenvolvimento e teste?
Medição 3.2	Impressão subjetiva das principais técnicas de Use Case 2.0 com maior aceitação pelas áreas.
Pergunta 3.3	Quais foram os principais pontos positivos, sob o ponto de vista das áreas, na aplicação da abordagem Use Case 2.0
Medição 3.3	Impressão subjetiva sob os pontos positivos da aplicação da abordagem Use Case 2.0.

A gerência do laboratório Bridge aprovou explicitamente a participação neste estudo de caso. O seu consentimento foi dado através de uma Declaração de Concordância (ANEXO A), assinada pela autora e pelo responsável pelo laboratório.

5.1.3 Planejamento da coleta de dados

Com a definição das questões/perguntas, um questionário foi elaborado utilizando perguntas de múltipla escolha e perguntas discursivas para as medidas de impressão subjetiva. As perguntas de múltipla escolha visam verificar o grau em que o avaliador concorda com as afirmações, dessa forma, essas foram elaboradas no formato de uma escala Likert (LIKERT, 1932) de 1 (discordo totalmente) a 5 (concordo totalmente). Já as perguntas discursivas visam levantar a opinião do avaliador sobre os pontos positivos e negativos da aplicação da abordagem Use Case 2.0. O questionário da avaliação encontra-se disponível no Apêndice A deste trabalho.

Para a coleta de dados quantitativos, foi utilizado as ferramentas de gerência de projetos Trello e Asana já usada pelos times "D" e "F" respectivamente. As ferramentas têm como principal funcionalidade registrar as atividades executadas pela equipe em cada sprint. Além disso, ambas as ferramentas possuem extensão para registrar a estimativa de esforço de desenvolvimento feita pela equipe. Para a coleta do esforço total em horas será utilizada a ferramenta PomoDone[9], técnica pomodoro para gerenciamento de tempo desenvolvido em cada atividade. O PomoDone será integrado ao Trello e Asana. A Tabela 12 apresenta as medidas e os respectivos procedimentos de coleta.

Tabela 8 - Procedimento de coleta das medidas

Medida	Responsável	Algoritmo	Forma de obtenção
Medida 1.1	Time de desenvolvimento	Esforço total em horas	(Trello ou Asana) e PomoDone

[9] https://pomodoneapp.com/

Medida 1.2	Time de desenvolvimento	[(Diferença entre a estimativa de esforço ao aplicar a abordagem e a estimativa de esforço antes da abordagem) / (Estimativa antes da abordagem)] x 100	Trello ou Asana
Medida 1.3	Analista de sistemas, Desenvolvedor e Analista de teste	(Horas despendida em cada área/ Esforço total em horas) *100	(Trello ou Asana) e PomoDone
Medida 2.1	Analista de sistemas, Desenvolvedor e Analista de teste	Perguntas de questionário	Questionário - A (APÊNDICE A)
Medida 2.2	Analista de sistemas, Desenvolvedor e Analista de teste	Perguntas de questionário	Questionário - A (APÊNDICE A)
Medida 2.2	Analista de sistemas, Desenvolvedor e Analista de teste	Perguntas de questionário	Questionário - A (APÊNDICE A)
Medida 3.1	Analista de sistemas, Desenvolvedor e Analista de teste	Perguntas de questionário	Questionário - A (APÊNDICE A)
Medida 3.2	Analista de sistemas,	Perguntas de questionário	Questionário - A (APÊNDICE A)

	Desenvolvedor e Analista de teste		
Medida 3.3	Analista de sistemas, Desenvolvedor e Analista de teste	Perguntas de questionário	Questionário - A (APÊNDICE A)

6 EXECUÇÃO DO ESTUDO DE CASO

Nesta seção é realizada a execução deste estudo de caso, conforme definido no Capítulo 5 em duas partes, conforme apresentado na Figura 21. Inicialmente, criou-se uma **estratégia de implantação da abordagem Use Case 2.0** em ambas as equipes. Nessa estratégia foram definidos os níveis de detalhamento de cada produto de trabalho proposta pela abordagem e as ferramentas utilizadas para desenvolver esses produtos. Na segunda parte, foi realizada a **aplicação da abordagem Use Case 2.0** nas equipes do estudo de caso. Nesta etapa foram realizadas as atividades de análise, desenvolvimento e teste, para a implementação dos produtos.

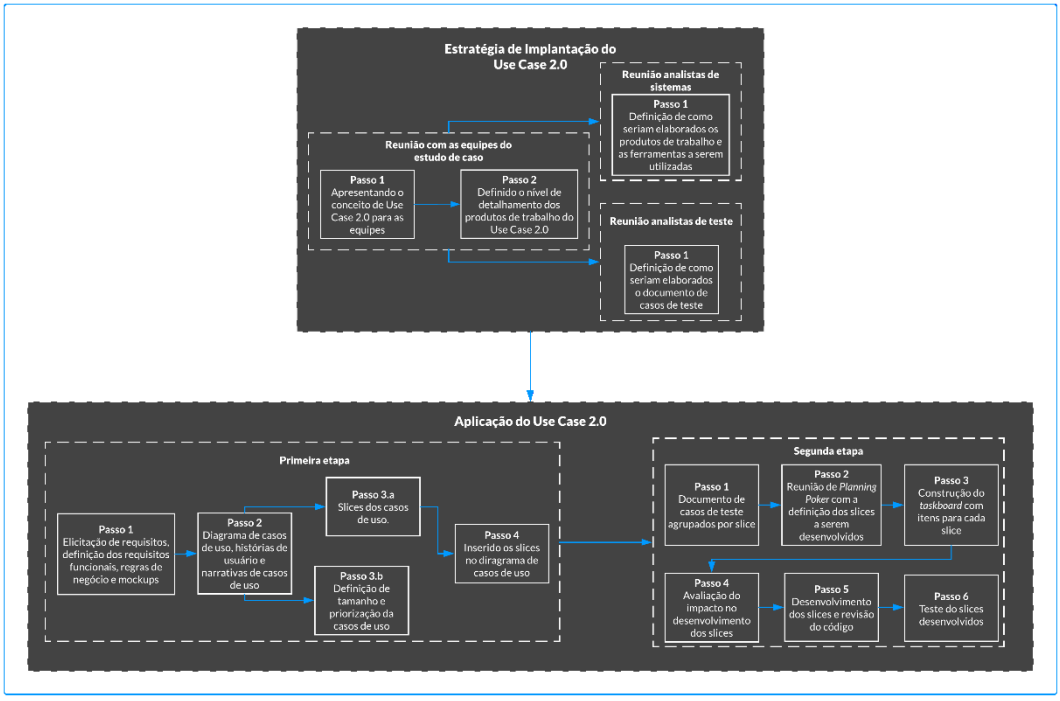


Figura 21 - Execução do estudo de caso

Durante toda a execução do estudo de caso, houve um acompanhamento da aplicação com o intuito de verificar o impacto, coletar os dados definidos no Capítulo 5 e analisar os resultados.

6.1 ESTRATÉGIA DE IMPLANTAÇÃO DA ABORDAGEM USE CASE 2.0 NAS EQUIPES

Foi realizada uma reunião com cada equipe do estudo de caso, com o objetivo de introduzir o conceito da abordagem Use Case 2.0 e definir o nível de detalhamento dos produtos de trabalho para a aplicação da abordagem, conforme descrito no Capítulo 2.4.

A reunião foi dividida em duas partes. Primeiro foram definidos os conceitos de diagrama de caso de uso, descrição de casos de uso e história de usuário para a elicitação de requisitos. Foram apresentadas também as motivações para a utilização do Use Case 2.0, os princípios da abordagem e as técnicas de Use Case Slices e História de usuário.

Ao final da primeira parte da reunião foi questionado a possibilidade da aplicação da abordagem nas equipes. Por ser um time que segue os princípios ágeis, possui um nível alto de descrição dos casos de teste, a abordagem proposta foi muito bem recebida. O foco no valor do produto e o trabalho em incrementos com uma melhor definição através dos slices, foram os pontos de maior aceitação pelas equipes.

6.1.1 Definindo os níveis de detalhamento

A segunda parte da reunião foi utilizada para a definição do nível de detalhamento dos produtos de trabalho a serem desenvolvidos para a aplicação da abordagem (Figura 22). Essa definição ocorreu levando em consideração o valor agregado em cada nível e o que já era implantado em cada um dos times de desenvolvimento, seguindo o princípio do Use Case 2.0 de adaptar-se à realidade da equipe.

Figura 22 -Níveis de detalhamento dos produtos de trabalho da abordagem Use Case 2.0 aplicado pelas equipes do estudo de caso

Ambas as equipes definiram o seguinte nível de detalhamento para os produtos de trabalho:

- **Modelo de caso de uso** - Nível Melhorado - Limite do sistema estabelecido. Conforme Jacobson et al. (2011), o nível de detalhe de limite de sistema estabelecido é útil ao modelar novos sistemas ou novas gerações de sistemas. Como o produto/módulo a ser desenvolvido em ambas as equipes é novo, foi decidido utilizar esse nível de detalhamento.

- **Narrativa dos casos de uso** - Nível Expandido - Detalhadamente descrito. Conforme definido por Jacobson et al. (2011), neste nível dada caso de uso deve ser detalhadamente descrito, apresentado o identificador, nome, atores, fluxos básicos, alternativos e de exceção.

- **Realização dos casos de uso** - Nível Essencial - Elementos de implementação identificados. Segundo Jacobson et al. (2011), este nível de detalhamento é adequado para pequenas equipes em que o produto a ser desenvolvido é simples.

- **Caso de teste** - Nível Expandido - Variáveis definidas. Como em ambas as equipes os casos de teste já são realizados nesse nível de detalhamento, isso foi mantido.

- **Requisitos complementares** - Nível Expandido - Definido de forma abrangente. Como ambas as equipes já produziam um documento de requisitos formal, contendo a especificação de requisito funcional, regras de negócio e protótipo de tela, foi decidido manter também esse suporte, com o intuito de minimizar o impacto nos times de desenvolvimento.

Ao final de reunião, foram marcados outros dois encontros, um com os analistas de sistemas das equipes para definir como seriam elaborados os produtos de trabalho e, quais ferramentas seriam utilizadas. E outro encontro, com os analistas de teste das equipes para definir como seria elaborado o documento de casos de teste para suportar a abordagem Use Case 2.0.

6.1.2 Ferramentas para aplicação da abordagem Use Case 2.0

Para a definição de como seriam implementados os produtos de trabalho da abordagem Use Case 2.0, foi realizado um encontro entre os analistas de sistemas das equipes. Nessa reunião ficou definido que os produtos de trabalho e artefatos produzidos ao aplicar a abordagem Use Case 2.0 seriam incorporados na documentação de requisitos já desenvolvida pela equipe. Nessa reunião também foram definidas a ferramentas para o desenvolvimento de alguns produtos de trabalho. Para a modelagem do diagrama de casos de uso foi definida a utilização do Lucidchart[10], por ser online, simples, gratuita e, por um dos analistas já utiliza-la. Foi definido também que as narrativas de casos de uso seriam descritas utilizando a linguagem Markdown, por ser uma linguagem de marcação leve e simples (GRUBER, 2004), e por já ser utilizada para a descrição dos requisitos funcionais e regras de negócio.

Para a gestão das tarefas e organização do desenvolvimento dos *slices* de casos de uso, foi definido que a equipe "D" e a equipe "F" utilizarão, respectivamente, as ferramentas Trello e Asana, por já serem a ferramenta de gestão de tarefas utilizadas pelas equipes, conforme descrito no Capítulo 5.1.3. Foi definido também, que o controle de tempo de trabalho seria feito utilizando a técnica Pomodoro, um método de gerenciamento de tempo que consiste em dividir o trabalho em períodos de 25 minutos, separados por breves intervalos. (CIRILLO, 2009). Para a aplicação da técnica Pomodoro foi utilizada a ferramenta PomoDone pois, essa possui integração com a ferramenta de gestão de tarefas utilizadas pelas equipes.

Foi realizado também um encontro entre os analistas de teste de cada equipe para definir quais adaptações o documento de casos de teste deveria sofrer para se adaptar a abordagem Use Case 2.0. Optou-se por manter os casos de teste na planilha online do Google, pois assim, o impacto seria muito menor. Porém, observou-se que para uma coesão entre o que seria desenvolvido e os casos de teste, seriam necessários alguns ajustes na planilha. Com isso, foi decidido separar os casos de teste em *slices* a serem desenvolvidos, porém, o fluxo básico do caso de uso seria feito separado, facilitando assim, referenciá-lo dentro de todos os outros *slices* do caso de uso.

6.2 APLICAÇÃO DA ABORDAGEM USE CASE 2.0 NAS EQUIPES

[10] https://www.lucidchart.com

Após todas as definições, as equipes puderam iniciar as atividades necessárias para aplicação da abordagem Use Case 2.0 no desenvolvimento dos produtos. Para isso, foram realizadas as atividades conforme definido no capítulo 2.4.6, sendo divididas em duas etapas. Na primeira, foram feitas as atividades para descobrir, ordenar, dimensionar e verificar os requisitos e definir os *slices*. Já a segunda etapa ficou responsável por complementar, implementar e testar os *slices* de caso de uso. A atividade de elicitação de requisito realizada na primeira etapa, foi feita somente uma vez durante o estudo de caso. Porém, as demais atividades da primeira etapa e a segunda etapa, foram realizadas nas duas sprints definidas para este estudo de caso.

6.2.1 Primeiras etapas

Esta etapa ocorreu, entre 19 de Fevereiro à 11 de Maio de 2018. A primeira atividade, conforme já comentado, foi realizada apenas uma vez, durante o estudo de caso. Esta atividade contemplada a elicitação dos requisitos através de uma entrevista com os *stakeholders* feita pelos analistas de sistemas da equipe. Com isso, foram definidos os objetivos do produto/módulo a ser desenvolvido, os requisitos funcionais, regras de negócio e *mockups* iniciais do produto. Em seguida, iniciou-se a construção do diagrama de casos de uso, as histórias de usuário e narrativas de casos de uso.

A construção desses artefatos foi refinada por meio de várias reuniões com o orientador deste trabalho e com o coordenador geral do laboratório. Em paralelo, foram desenvolvidos, pelo designer do projeto, com suporte do analista de sistemas, os protótipos de alta fidelidade do produto/módulo.

Com o fluxo básico e os fluxos alternativos de casos de uso definidos, foram iniciados os cortes nos casos de uso. Os cortes foram feitos de modo que todos tivessem valor agregado, podendo assim ser testados pelos analistas de teste. Para cortar, foram seguidas as recomendações dadas por Jacobson et al (2011), em que recomenda que o fluxo básico seja sempre o primeiro *slice* a ser desenvolvido e que deve-se criar *slices* de tamanho adequado para a equipe trabalhar. Ao finalizar as definições dos *slices*, foi atualizado o diagrama de casos de uso, onde foram inseridos os *slices* definidos para cada caso. A Figura 23 apresenta um exemplo de um dos diagramas de casos de uso com slice associado, desenvolvidos por uma das equipes do estudo de caso.

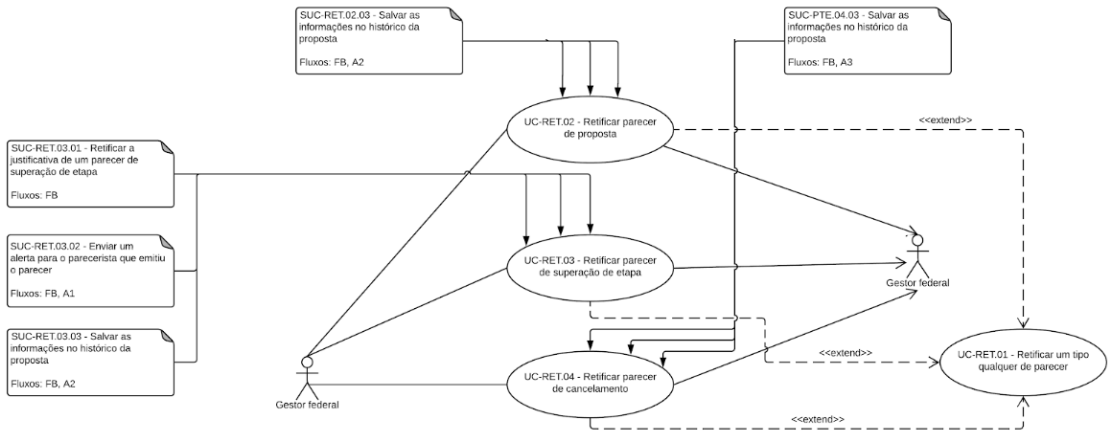


Figura 23 - Diagrama de casos de uso com *slice* associado

Em paralelo à definição de *slices*, foi feita a estimativa de tamanho do caso de uso e a priorização dos casos de usos. A estimativa de tamanho foi feita utilizando o método de Pontos de Caso de Uso - PCU (WAZLAWICK, 2011) e a ferramenta Use Case Calculator[11]. O método PCU fornece uma estimativa de tamanho de software logo na fase inicial, tendo como entrada os casos de uso. Após os casos de uso serem descritos, é possível gerar uma estimativa com a contagem PCU – este é o nome do método e a unidade de medida do método (WAZLAWICK, 2011). Para realizar a definição de tamanho utilizando PCU foram considerados:

- A classificação dos atores em Simples, Médio e Complexo.
- A classificação dos casos de uso em Simples, Médio ou Complexo.
- O fator de complexidade técnica (TCF) para cada caso de uso.
- O fator ambiental (EF) para cada caso de uso.

Além disso, vale ressaltar, que para a classificação dos casos de uso, foram consideradas apenas as trocas de mensagens entre o usuário e sistema, ou seja, a partir da expansão dos casos de uso existe um determinado conjunto de passos que podem ser diferenciados em “obrigatórios”, enquanto os demais são “complementares”. Os passos “obrigatórios” são aqueles que determinam o esforço de desenvolvimento, proporcionalmente ao seu número, enquanto que os passos “complementares” não têm influência sobre o desenvolvimento (WAZLAWICK, 2011).

[11] http://ucp-calculator.appspot.com/

Para a priorização dos casos de uso foi utilizada a técnica MoSCoW, que consiste em classificar os requisitos a serem desenvolvidos em *Must, Should, Could e Won't.* Essa é uma técnica de priorização simples, feita em conjunto com *stakeholder* (OLIVEIRA, 2014). A Tabela 13 apresenta os pontos de caso de uso e a priorização dos casos de uso desenvolvidos durante as duas sprints em que o estudo de caso foi realizado.

Tabela 9 - PCU e Priorização dos casos de uso desenvolvidos

SPRINT 1			
Equipe	**Caso de Uso**	**PCU**	**Priorização**
"F"	UC-WIZ.01 - Cadastrar com wizard	7,41	Must Have
"D"	UC-RET.01 - Retificar um tipo qualquer de parecer	9,57	Must Have
	UC-RET.02 - Retificar parecer de proposta	7,89	Should Have
SPRINT 2			
Equipe	**Caso de Uso**	**PCU**	**Priorização**
"F"	UC-END.01 - Preencher endereço	7,41	Must Have
	UC-AEN.01 - Alteração de endereço	7,41	Must Have
"D"	UC-RET.03 - Retificar parecer de superação de etapa	6,90	Must Have
	UC-RET.04 - Retificar parecer de cancelamento de obra	9,37	Must Have
	UC-RET.05 - Retificar parecer de prorrogação de prazo	5,93	Should Have
	UC-RET.06 - Retificar parecer de alteração de endereço	5,93	Should Have

Ao final de todas as atividades da primeira parte, a documentação de requisitos com os produtos de trabalho da abordagem Use Case 2.0 foi concluída. A Figura 24 apresenta o diagrama de casos de uso a ser desenvolvido incorporado à documentação de requisitos do projeto SISMOB, e a Figura 25 apresenta a narrativa de um caso de uso,

com a definição do fluxo básico e alternativos, a história de usuário, a priorização do caso de uso, versão de entrega, tamanho, definição dos *slices* e requisitos funcionais já definidos. A priorização e a definição dos testes foram feitas na segunda parte. Os demais casos de uso desenvolvidos pela equipe em todas as sprints do estudo de caso estão descritos no Apêndice B.

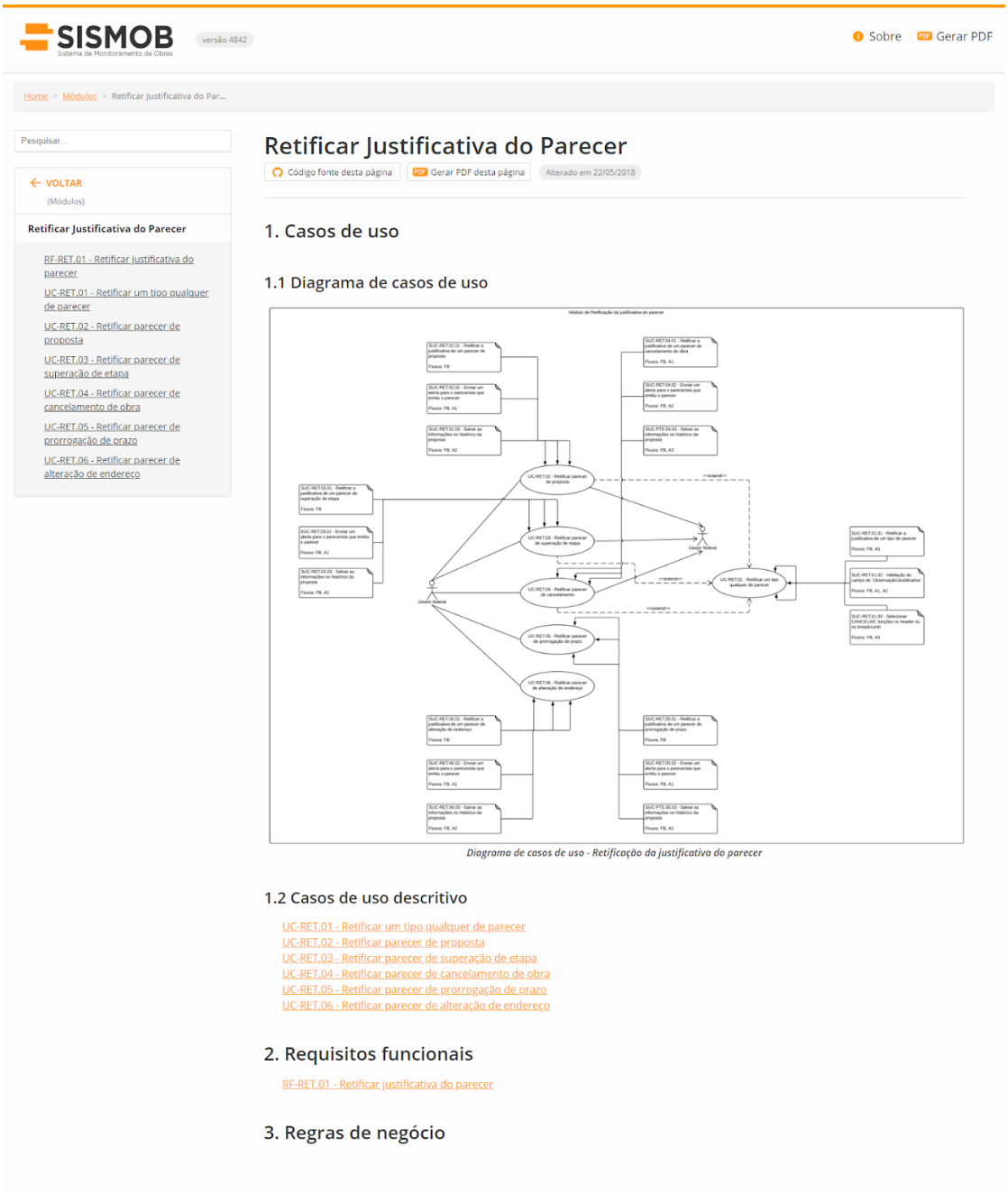


Figura 24 - Diagrama de casos de uso com *slices*

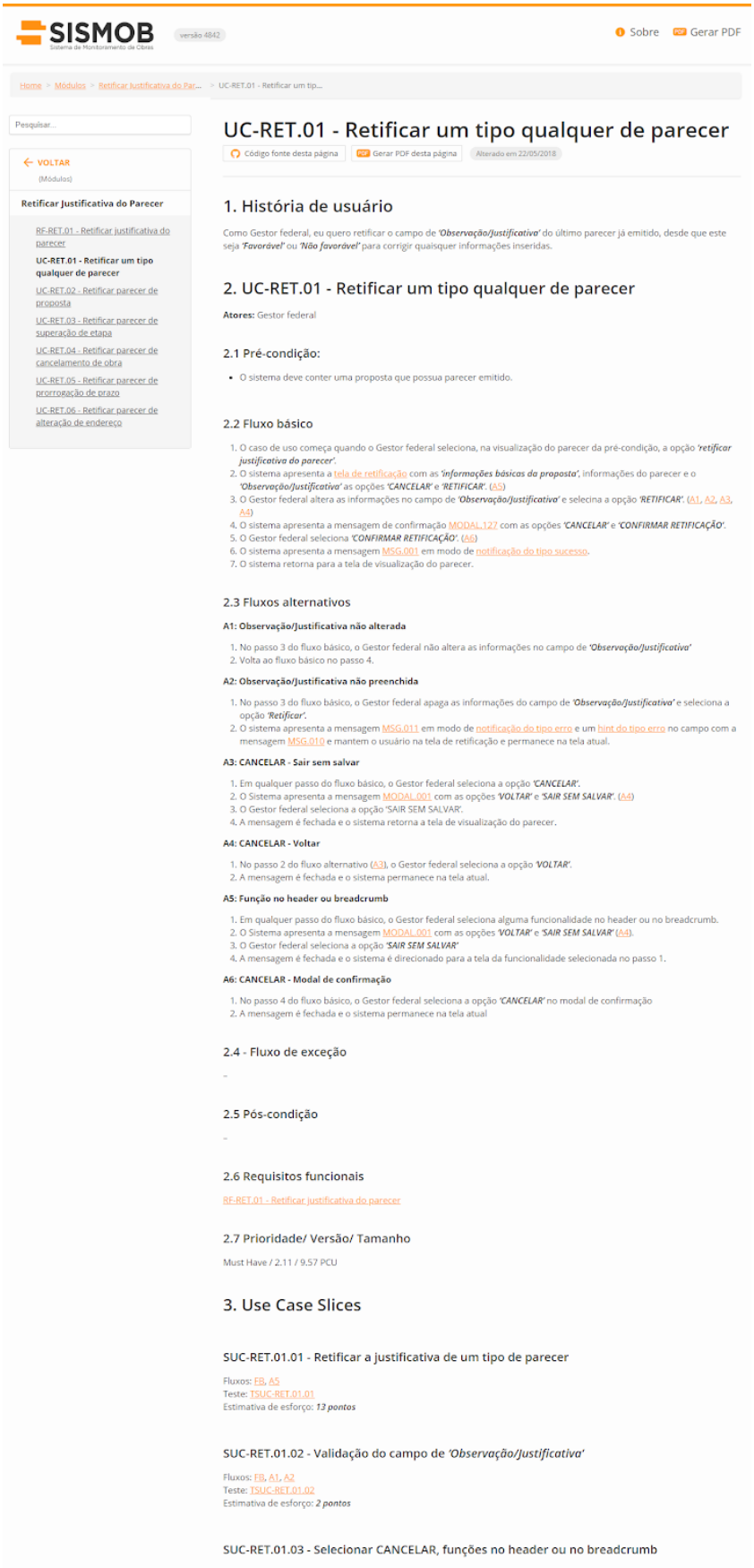
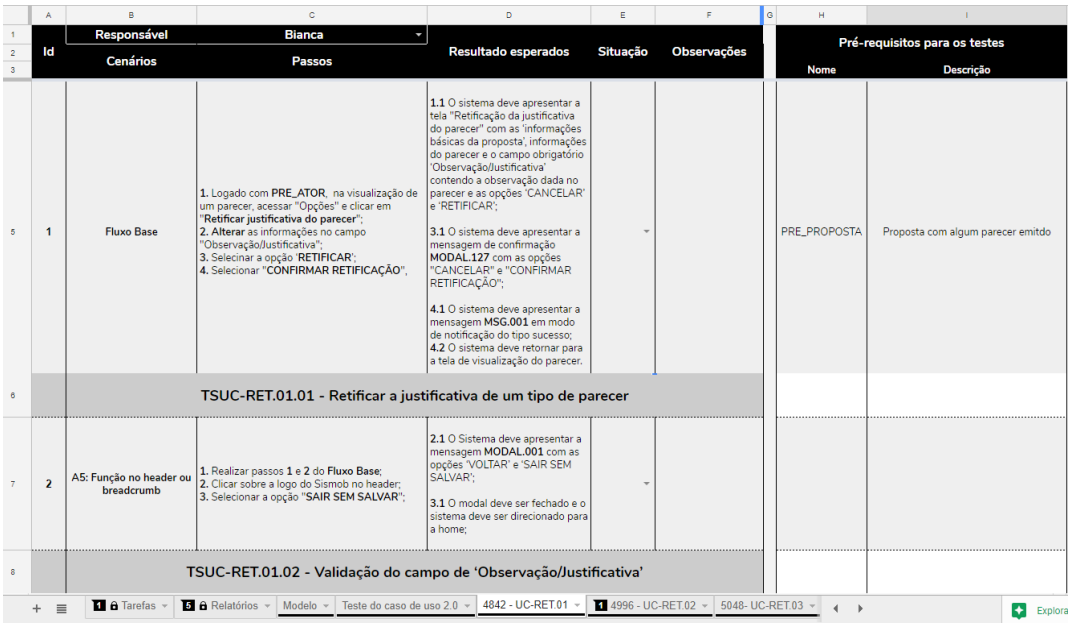


Figura 25- Narrativa de Caso e Uso, História de Usuário, *Slices* e Requisitos funcionais

6.2.2 Segundas etapas

Esta segunda etapa foi realizada durante as duas sprints definidas para esse estudo de caso. A primeira sprint foi realizada no período entre 07 de Maio à 18 de Maio de 2018 e, a segunda sprint ocorreu entre 21 de Maio à 01 de Junho de 2018.

A segunda etapa inicia com a construção do documento de testes realizado pelos analistas de testes das equipes. Essa construção foi feita com base no documento de requisitos e, definindo os cenários de teste, os passos de teste e os resultados esperados para cada cenário. Esses testes foram então agrupados por *slices* como pode ser observado na Figura 26. Foi inserido na documentação de requisitos o link para cada conjunto de casos de teste por *slice*.



Id	Responsável / Cenários	Bianca / Passos	Resultado esperados	Situação	Observações	Pré-requisitos para os testes: Nome	Pré-requisitos para os testes: Descrição
1	Fluxo Base	1. Logado com PRE_ATOR, na visualização de um parecer, acessar "Opções" e clicar em "**Retificar justificativa do parecer**"; 2. **Alterar** as informações no campo "Observação/Justificativa"; 3. Selecinar a opção '**RETIFICAR**'; 4. Selecionar "**CONFIRMAR RETIFICAÇÃO**",	**1.1** O sistema deve apresentar a tela "Retificação da justificativa do parecer" com as 'informações básicas da proposta', informações do parecer e o campo obrigatório 'Observação/Justificativa' contendo a observação dada no parecer e as opções 'CANCELAR' e 'RETIFICAR'; **3.1** O sistema deve apresentar a mensagem de confirmação **MODAL.127** com as opções "CANCELAR" e "CONFIRMAR RETIFICAÇÃO"; **4.1** O sistema deve apresentar a mensagem **MSG.001** em modo de notificação do tipo sucesso; **4.2** O sistema deve retornar para a tela de visualização do parecer.			PRE_PROPOSTA	Proposta com algum parecer emitdo
	TSUC-RET.01.01 - Retificar a justificativa de um tipo de parecer						
2	A5: Função no header ou breadcrumb	1. Realizar passos **1** e **2** do **Fluxo Base**; 2. Clicar sobre a logo do Sismob no header; 3. Selecionar a opção "**SAIR SEM SALVAR**";	**2.1** O Sistema deve apresentar a mensagem **MODAL.001** com as opções 'VOLTAR' e 'SAIR SEM SALVAR'; **3.1** O modal deve ser fechado e o sistema deve ser direcionado para a home;				
	TSUC-RET.01.02 - Validação do campo de 'Observação/Justificativa'						

Tarefas | Relatórios | Modelo | Teste do caso de uso 2.0 | 4842 - UC-RET.01 | 4996 - UC-RET.02 | 5048- UC-RET.03

Figura 26 - Planilha de casos de teste agrupados por *slices*

Com o documento de requisitos e os casos de teste criados, foi realizada a reunião de *Planning Poker* (COHN, 2005) para definir quais casos de uso seriam desenvolvidos na sprint. Nesta reunião foram apresentados os *slices* para cada caso de uso e foram feitas as estimativas de esforço de desenvolvimento para cada *slice*. Como resultado da reunião, foram identificados os *slices* a serem desenvolvidos na sprint e o *taskboard* da equipe [12]para sprint foi definido, conforme Figura 27. Como pode ser observado na Figura 27,

[12] Como o *taskboard* da equipe "D" e "F" possuem os mesmos artefatos, optou-se por apresentar apenas de uma das equipes

cada item do *taskboard* representa um *slice* a ser desenvolvido. O item ainda contém a estimativa de esforço para o desenvolvimento do *slice*. É apresentado também um item expandido, onde pode-se observar as referências a planilha de casos de teste e a

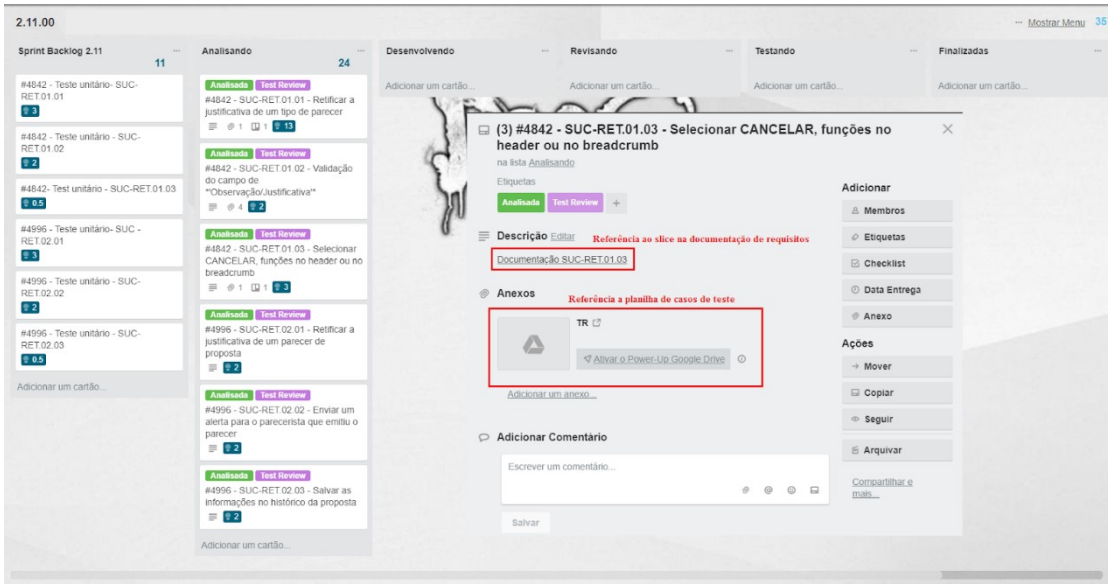


Figura 27 - *Taskboard* da equipe "D" utilizando a abordagem Use Case 2.0 documentação de requisitos.

Após a definição da sprint, foi realizada uma reunião com os desenvolvedores para avaliar o impacto no desenvolvimento dos *slices*. Essa reunião foi feita utilizando a definição dos *slices* e o código do sistema, com o intuito de identificar quais atividades eram necessárias a serem desenvolvidas em cada *slice*. Ao final dessa reunião, foi montado um documento de atividades necessárias a serem realizada em cada *slice*.

Com todas as definições realizadas, foi iniciada a etapa de desenvolvimento dos *slices*. Para isso, foi necessário que o primeiro *slice* de cada caso de uso estivesse pronto, para que os demais pudessem ser desenvolvidos. Ao finalizar o desenvolvimento de um *slice*, foi realizada a revisão por um outro desenvolvedor do código implementado. Após a revisão, o *slice* foi entregue a um dos analistas de sistema da equipe para que pudesse ser testado utilizando os casos de teste previamente desenvolvidos. Por fim, ao finalizar o teste, o *slice* foi integrado ao caso de uso desenvolvido. Ao final da sprint, com os *slices* prontos, os casos de usos foram integrados ao sistema para que pudessem ser entregues e validados pelo *stakeholder*.

Essas duas etapas ocorreram nas duas sprints em que esse estudo foi realizado e, foram feitas por todas as equipes do estudo. Por falta de tempo, a validação do produto desenvolvido com o *stakeholder* ocorreu apenas com a equipe "D" e foi realizada somente na primeira sprint desenvolvida.

6.2.3 Coletas de Dados

O período de coleta de dados iniciou-se em 09 de Abril de 2018, sendo utilizadas, no total, 4 sprints de duas semanas. Durante a execução do estudo de caso as métricas foram coletadas por sprints dos dois times de desenvolvimento. Nas duas primeiras sprints, nenhuma intervenção foi realizada para a coletada de dados e, foram coletadas apenas as estimativas de esforço antes da aplicação da abordagem Use Case, 2.0. Nas duas sprints seguintes foram aplicadas as técnicas da abordagem Use Case 2.0 em ambas as equipes. Os artefatos criados para a realização desse estudo de caso foram elaborados pelos analistas de sistemas e analistas de testes dos times de desenvolvimento e, alguns podem ser visualizados ao final deste trabalho (vide Apêndice B).

Para a coleta de algumas medidas referentes aos objetivos de medição 2 e 3, um questionário (vide Apêndice A) foi elaborado e aplicado aos Analistas de sistemas, Desenvolvedores de software e Analistas de testes das equipes "D" e "F" do projeto SISMOB do Laboratório Bridge. Os restantes das medidas foram coletadas por meio das ferramentas de gerenciamento de projetos Trello utilizada pela equipe "D" e Asana utilizada pela equipe "F". Além disso, ambas as equipes utilizam a ferramenta PomoDone para gerenciamento de tempo de trabalho.

6.3 ANÁLISE DE DADOS

Finalizadas a coletada de dados, esses são analisados para verificar o atendimento dos objetivos estabelecidos. A análise dos dados foi realizada para cada objetivo, agrupando as respostas das perguntas de acordo com as medidas. Cabe salientar que, como a aplicação do questionário de avaliação foi feita perante a autora, algumas questões de escolha simples eram complementadas, enriquecendo assim a coleta de dados. Sendo assim ao analisar os dados levou-se também em consideração esses apontamentos feitos paralelamente às respostas do questionário.

OBJETIVO DE MEDIÇÃO 1 - Analisar o esforço e a produtividade da equipe na aplicação das técnicas de Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

Pergunta 1.1 (Q1)	Qual o esforço total despendido pelas áreas de análise, desenvolvimento e teste para aplicação da abordagem Use Case 2.0?
Medição 1.1	Esforço total em horas para o desenvolvimento total do produto, destacando o que foi despendido no desenvolvimento do produto, planejamento da aplicação da abordagem Use Case 2.0, novas atividades do Use Case 2.0 e atividades já realizadas do Use Case 2.0.

O esforço total despendido para desenvolver os produtos utilizando a abordagem Use Case 2.0 no time de desenvolvimento "D" foi de 292,81 horas e, para o time de desenvolvimento "F" foi de 230,38 horas (Figura 28 e Figura 29). Tal esforço foi executado no período de Fevereiro à Maio de 2018, com duração de 4 meses, compreendendo todas as atividades relacionadas à pesquisa, planejamento e execução da abordagem Use Case 2.0 e, consultoria externa, conforme descrito nos Capítulos 6.1 e 6.2.

Ao considerar somente os produtos de trabalho propostos pela abordagem Use Case 2.0, conforme definido no capítulo 6.1, levando em conta também o esforço aplicado na estratégia de implantação da abordagem e o esforço despendido nas pesquisas e consultorias externas, verifica-se que para o time de desenvolvimento "D" despendeu um esforço total 138,42 horas e para o time de desenvolvimento "F" consumiu um esforço total 109,82 horas. Ou seja, o percentual de horas despendidas no desenvolvimento dos produtos propostos pela abordagem Use Case 2.0, em relação a total de horas utilizadas para a execução de todo o produto foi de 47,27% para o time "D" e 47,67% para o time "F".

Em contrapartida, alguns dos produtos de trabalho propostos pelo Use Case 2.0, já eram realizadas no processo atual de desenvolvimento de produto das equipes conforme descrito no capítulo 4, como os requisitos complementares (requisitos funcionais e regras de negócio) e os casos de teste. Sendo assim, ao somar o esforço total despendido para executar somente as novas atividades propostas pelo Use Case 2.0 e que

ainda não eram realizadas pelos times de desenvolvimento mais o esforço aplicado na estratégia de implantação da abordagem e o esforço despendido nas pesquisas e consultorias externas, verifica-se que para o time de desenvolvimento “D” despendeu um esforço total 93,42 horas e para o time de desenvolvimento “F” consumiu um esforço total 74,83 horas. Ou seja, o percentual de horas despendidas na execução dessas atividades propostas pela abordagem Use Case 2.0 que não eram realizadas no processo atual de desenvolvimento das equipes, em relação a total de horas utilizadas para a execução de todo o produto foi de 31,91% para o time “D” e 32,48% para o time “F”.

A medida acima pode ser dividida em esforço inicial para a aplicação da abordagem, levando em consideração o tempo despendido em pesquisa, planejamento e consultoria externa e, o esforço despendido para o desenvolvimento dos novos produtos de trabalho necessários para a aplicação da abordagem Use Case 2.0, que não eram desenvolvidos pelas equipes do estudo. Sendo assim, o percentual do esforço inicial empregado na estratégia de planejamento para a aplicação da abordagem Use Case 2.0 pela equipe “D” foi de 14,69% (43,00 horas) e para a equipe F foi de 11,72% (27,00 horas). Ou seja, o percentual de esforço necessário para o desenvolvimento dos produtos de trabalho necessário para a aplicação da abordagem que ainda não eram desenvolvidos pelas equipes foi de 17,22% (50,42 horas) para a equipe “D” e 20,76% (47,83 horas) para a equipe “F”.

Figura 28 - Esforço total para o desenvolvimento do produto - Equipe "D"

Figura 29 - Esforço total para o desenvolvimento do produto - Equipe "F"

Pergunta 1.2 (Q2)	Houve aumento na produtividade na equipe ao aplicar a abordagem Use Case 2.0?
Medição 1.2	Diferença percentual entre a estimativa de esforço da equipe ao aplicar a abordagem Use Case 2.0 em relação a estimativa de esforço da mesma equipe antes da aplicação da abordagem.

Para esta pergunta foram considerados os pontos de história estimados pelas equipes utilizando a técnica P*lanning Poker*, que estima o esforço necessário para o desenvolvimento de um produto baseado no consenso e experiência dos participantes da equipe. Para considerar se houve aumento na produtividade, foi levado em consideração os pontos estimados por uma equipe no desenvolvimento de um produto, durante duas sprints, sem a utilização da abordagem Use Case 2.0. Em comparação com os pontos estimados pela mesma equipe para o desenvolvimento de um outro produto, também durante duas sprints, porém utilizando a abordagem Use Case 2.0.

No desenvolvimento de produtos feitos pelas equipes “D” e “F”, durante duas sprints de duas semanas sem a utilização da abordagem Use Case 2.0, foi estimado um esforço de 57,5 pontos para a equipe “D” e de 40, 5 para a equipe “F”.

Comparando a estimativa de estimativa de esforço da equipe ao aplicar a abordagem Use Case 2.0 em relação a estimativa de esforço da mesma equipe antes da aplicação da abordagem, pode-se observar que houve um aumento de produtividade de 42,61% para a equipe “D” e de “50,62%” para a equipe “F”. Sendo assim, podemos concluir que, conforme apontado no questionário, por todos os membros de ambas as equipes, um dos pontos positivos da aplicação da abordagem Use Case 2.0 é a clareza do valor do produto que a história do usuário fornece e, a facilidade de compreender o que deve ser entregue em cada parte na divisão por *slices*, o que acarreta em uma estimativa de esforço mais precisa.

Pergunta 1.3 (Q3)	Quanto esforço cada área demandou no desenvolvimento do produto utilizando a abordagem Use Case 2.0
Medição 1.3	Proporção de esforço despendido em cada área em relação ao total de horas despendido para a aplicação da abordagem Use Case 2.0.

O esforço demandado por cada área do time de desenvolvimento para aplicação da abordagem Use Case 2.0 pode ser observado nas Figuras 30 e 31. Para esta medição foram considerados apenas o esforço despendido pelas áreas para a aplicação dos novos produtos de trabalho, o esforço aplicado na estratégia de implantação da abordagem e o esforço despendido nas pesquisas e consultorias externas. Como pode ser observado na nas Figuras 28 e 29, a área de análise de software foi a que demandou maior parte das horas despendidas para a aplicação da abordagem Use Case 2.0, sendo responsável por aproximadamente 85,91 % (118,91 homens/horas) para o time de desenvolvimento “D” e 85,51% (93,91 homens/hora) para o time de desenvolvimento “F”. Os resultados observados estão de acordo com processo de execução do estudo de caso relatado no Capítulo 5, uma vez que, o maior número de produtos de trabalho e artefatos necessárias para aplicação da abordagem Use Case 2.0 eram de responsabilidade dos analistas de sistemas.

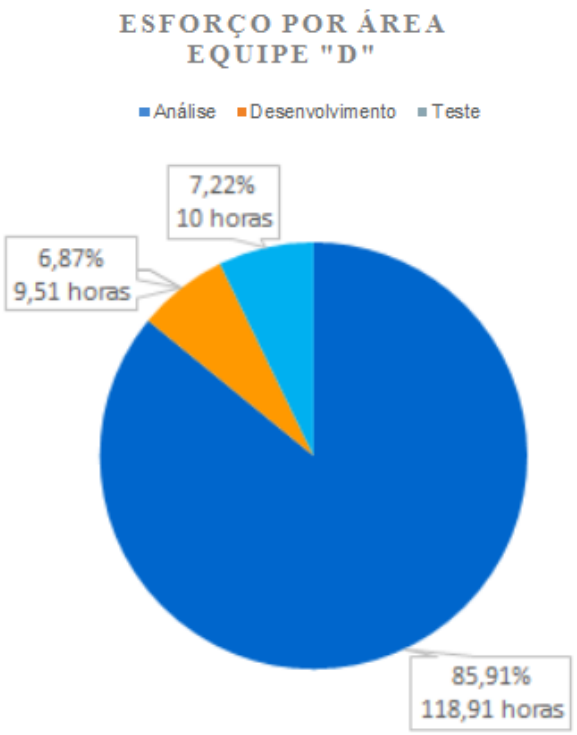


Figura 30 - Proporção do esforço demandado por cada área do time de desenvolvimento “D” para aplicação do Use Case 2.0

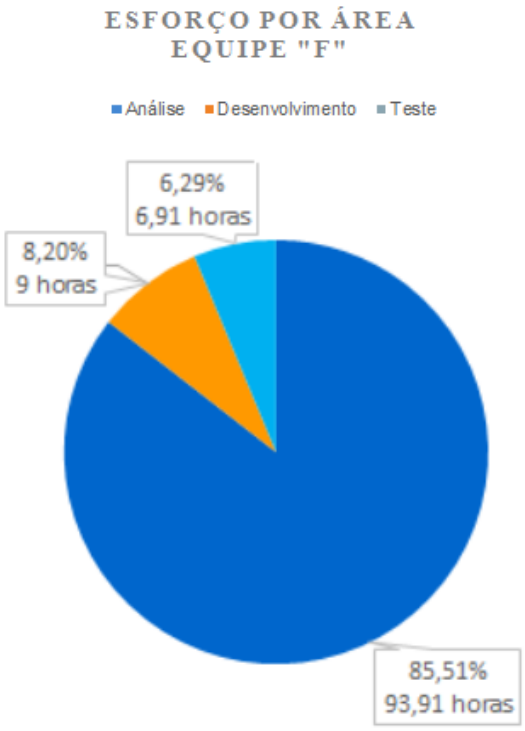


Figura 31 - Proporção do esforço demandado por cada área do time de desenvolvimento "F" para aplicação do Use Case 2.0

OBJETIVO DE MEDIÇÃO 2 - Analisar as dificuldades para aplicação das técnicas da abordagem Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

Pergunta 2.1 (Q4)	Como você classificaria a facilidade de compreender e utilizar a abordagem Use Case 2.0?
Medição 2.1	Impressão subjetiva das principais técnicas da abordagem Use Case 2.0

Para 63,64% dos colaboradores (7) a abordagem Use Case 2.0 é de **fácil** de compreensão e utilização e, 27,27% dos colaboradores (3) afirmaram ser **muito fácil** compreender e utilizar a abordagem**.** Contudo, 9,09% dos colaboradores (1), declarou que a abordagem **não é fácil e nem difícil** de compreender e utilizar (Figura 32). Este afirma que a abordagem é simples de compreender, porém, por ser da área de análise de sistemas,

a produção de alguns artefatos, como por exemplo, a especificação de requisitos com casos de uso e utilização da técnica de pontos de caso de uso para definir o tamanho dos casos de uso foi um pouco difícil de ser realizada. Todavia, o mesmo concorda que esta é uma dificuldade inicial e, se dá apenas por uma falta de experiência na execução de algumas técnicas.

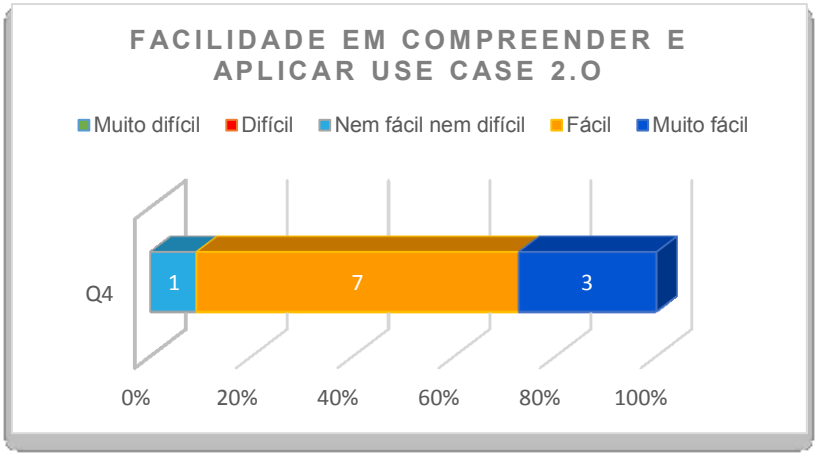


Figura 32 - Proporção da resposta à pergunta Q4

Sobre a facilidade de compreender e utilizar a técnica, 1 colaborador citou que, como já possuía conhecimento sobre a abordagem de casos de uso, compreender a proposta da abordagem Use Case 2.0 foi muito fácil. Outros 3 colaboradores, das áreas de desenvolvimento e teste de software também relataram que a aplicação das técnicas da abordagem Use Case 2.0, em especial, a **história de usuário**, **descrição de caso de uso** e a **divisão por *slices***, fez com objetivo principal do produto ficasse explícito. Isso fez com que a equipes conseguissem focar no que realmente era relevante a ser entregue para o cliente. Este foco no valor era, conforme por eles relatado, umas das dificuldades das equipes ao desenvolver um produto.

No entanto, 2 colaboradores destacaram uma certa dificuldade em compreender o que deveria ser desenvolvido em cada *slice*. Um dos participantes relatou que o desenvolvimento em partes, como proposto pela divisão em *slices*, um pouco diferente do modo como era feita, até então, a divisão para o desenvolvimento do produto, pois esta nova proposta foca no comportamento do sistema para dividir e não nos requisitos funcionais. Porém, o mesmo destacou que isso ocorreu por uma falta de experiência tanto na divisão do caso de uso por *slices*, quanto na experiência em desenvolver o produto em partes valoradas.

Pergunta 2.2 (Q5)	Quão benéfica foi para a unidade organizacional a aplicação da abordagem e, porque?
Medição 2.2	Impressão subjetiva da experiência com a aplicação da abordagem Use Case 2.0

Para 100% dos colaboradores (11) afirmaram que a aplicação da abordagem Use Case 2.0 é **benéfica** a unidade organizacional. Sendo que, desse 54,55% dos colaboradores (6) afirmaram que a aplicação da abordagem Use Case 2.0 é **muito benéfica** para a organização (Figura 33). Os colaboradores (11) relataram que, por apresentar de forma explícita e clara o progresso no desenvolvimento do produto, o repasse dessas informações para o Gerente de projeto foi melhorado. Com isso, os colaboradores acreditam que quaisquer problemas que possam vir a ocorrer durante o andamento de uma sprint, seriam previamente detectados.

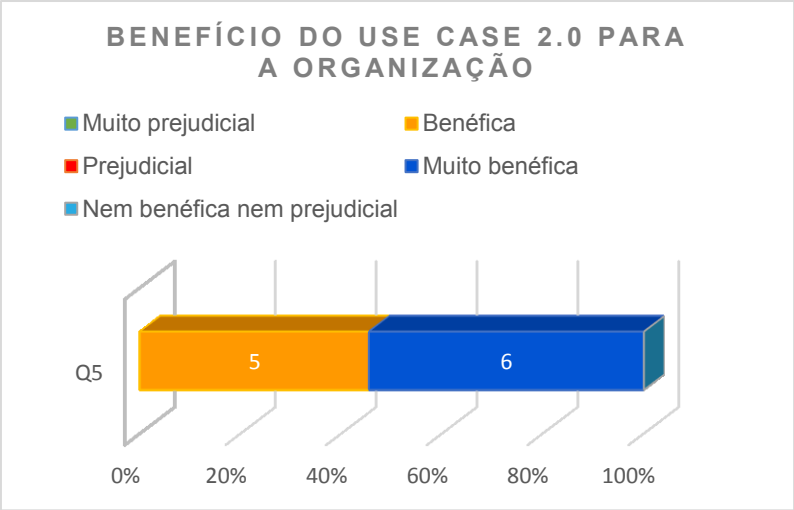


Figura 33 - Proporção da resposta à pergunta Q5

A organização da documentação de requisitos e o *taskboard* das equipes também foi destacado por 2 colaboradores como algo benéfico à organização, pois, qualquer membro da equipe sabe qual o andamento no desenvolvimento do produto. Isso, segundo os colaboradores, era algo que não ocorria antes da aplicação da abordagem.

Outro ponto também levantado por alguns colaboradores (3) das equipes "D", que conseguiu terminar uma parte do produto e homologar com o *stakeholder*, foi que a

entrega do produto em interações trouxe um feedback antecipado do cliente, o que fez com que os ajustes necessários fossem antecipados.

Pergunta 2.3 (Q6)	Quais foram as principais dificuldades encontradas na aplicação abordagem Use Case 2.0?
Medição 2.3	Impressão subjetiva da experiência com a aplicação da abordagem Use Case 2.0

As respostas a essa pergunta foram bastante homogêneas. Contudo, 5 colaboradores não souberam relatar nenhuma dificuldade. As principais dificuldades encontradas na aplicação da abordagem Use Case 2.0 apontadas pelos membros das equipes que participaram da entrevista foram agrupadas pela autora através da busca pelo termo de similaridade destacado e estão descritas abaixo:

- **Especificação de requisitos -** Para aplicar a abordagem Use Case 2.0 foi necessário a construção de alguns artefatos conforme mencionado no Capítulo 6, o que tornou a construção do documento de especificação de requisitos um pouco mais trabalhosa. Outro ponto de dificuldade também citado por um dos analistas de sistemas foi a falta de experiência em especificar descrevendo o comportamento do sistema e, a falta de conhecimento na técnica de PCU para determinar o tamanho de cada caso de uso. O colaborador também destacou a dificuldade de manter a rastreabilidade entre os requisitos funcionais e os casos de uso, uma vez que a ferramenta utilizada para documentar não fornece nenhum suporte.
- **Divisão por *slices* -** Uma dificuldade levantada por 2 colaboradores da área de Desenvolvimento de software, foi a dependência no desenvolvimento do primeiro slice, pois, como esse contém o fluxo básico, para o desenvolvimento dos demais slices, era necessário que esse estivesse implementado. Outros dois desenvolvedores e um analista de teste destacaram certa dificuldade em se ater ao que deveria ser desenvolvido e testado em cada *slice* pois, como antes da aplicação da abordagem não existia uma divisão clara no que deveria ser desenvolvido e testado, houve uma certa resistência em se ater apenas ao que era descrito no *slice*.

- **Abordagem nova -** Por ser uma nova abordagem, com impacto na organização das equipes, um colaborador relatou uma dificuldade inicial de se organizar o *taskboard* para começar o desenvolvimento e teste de cada *slice*.

Objetivo de Medição 3 - Analisar a aceitação das técnicas da abordagem Use Case 2.0 utilizadas sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

Pergunta 3.1 (Q7)	A equipe pretende continuar utilizando a abordagem Use Case 2.0? Porquê?
Medição 3.1	Impressão subjetiva sobre da aplicação da abordagem na equipe.

Todos os colaboradores (11) responderam que pretendem continuar utilizando a abordagem Use Case 2.0. Dentre as justificativas, 8 colaboradores citaram que a abordagem fornece-lhes entendimento mais claro do objetivo do produto a ser desenvolvido. Além disso, 6 colaboradores afirmaram que a abordagem auxiliou na organização do trabalho e, apresentando de forma explícita o andamento do desenvolvimento do produto, ficando claro o que estava sendo desenvolvido e testado e o que ainda faltava desenvolver e testar. Outro ponto comentado por 10 colaboradores foi que a divisão por *slices* facilitou a estimativa de esforço do produto pois, as áreas puderam compreender exatamente o que deveria ser desenvolvido e testado em cada parte.

Um fato importante levantado por 3 colaboradores foi existência de uma documentação descrevendo o comportamento do sistema (Narrativa de casos de uso e história do usuário) e não apenas regras e requisitos. Segundo os mesmos, isso facilitou o entendimento do produto a ser desenvolvido, além de contribuir para a criação dos casos de teste e a realização do teste exploratório.

Em contrapartida, apenas 1 colaborador ressaltou que pretende continuar utilizando a abordagem, porém, para isso, será necessário um amadurecimento na maneira na divisão dos *slices*, a fim de diminuir o impacto dos mesmos no desenvolvimento. Contudo, o mesmo ressaltou que isso se deve pela falta de experiência na aplicação da técnica.

Pergunta 3.2 (Q8)	Quais foram as principais técnicas da abordagem Use Case 2.0 utilizada que obtiveram maior aceitação por parte das áreas de análise, desenvolvimento e teste?
Medição 3.2	Impressão subjetiva das principais técnicas de Use Case 2.0 com maior aceitação pelas áreas.

As respostas a essa pergunta foram bastante homogêneas. Dentre todas as técnicas e artefatos produzidos na abordagem Use Case 2.0 as citadas pelos membros das equipes que participaram da entrevista, agrupadas pela autora por termo de similaridade e ordenado por número de citações:

Tabela 10 - Técnicas e artefatos gerados pela abordagem Use Case 2.0 que obtiveram maior aceitação pelas equipes

#	**Técnicas e artefatos**	**Ocorrências**
1	Divisão dos casos de uso por *slices*	11
2	Descrição dos casos de uso	9
3	Estimativa de esforço por *slice*	7
4	Histórias de usuário	4
5	Diagrama de casos de uso	2

Pergunta 3.3 (Q9)	Quais foram os principais pontos positivos, sob o ponto de vista das áreas, na aplicação da abordagem Use Case 2.0
Medição 3.3	Impressão subjetiva sob os pontos positivos da aplicação da abordagem Use Case 2.0.

Essa pergunta trouxe inúmeras respostas, porém, ao agrupá-las por termo de similaridades obteve-se os seguintes pontos positivos:

- **Organização na documentação de requisitos -** Foi destacado por todos os colaboradores (11), que a abordagem trouxe uma maior organização na documentação de requisitos, trazendo mais claramente o que deveria ser desenvolvido. Outro ponto positivo levantado por 1 colaborador, foi o registro na documentação de requisitos da versão em que o caso de uso foi desenvolvido.
- **Especificação de requisitos focados no comportamento do sistema -** Outro destaque importante, levantado por 10 colaboradores, foi a inclusão de uma especificação de requisitos utilizando casos de uso. Este método de especificação forneceu uma visão mais clara do que o sistema deveria fazer. Além disso, a história do usuário associada ao caso de uso foi destacada por todos os colaboradores (11) como um ponto positivo, pois forneceu uma visão clara do objetivo principal do produto a ser desenvolvido.
- **Rastreabilidade entre o documento de teste e o documento de requisitos -** Um ponto positivo levantado 5 colaboradores foi a rastreabilidade entre o documento de casos de teste e o documento de requisitos construído utilizando a abordagem Use Case 2.0. Segundo os mesmos, havia uma dificuldade em localizar os casos de teste associados aos produtos novos.
- **Reutilização de código -** Os analistas de sistema apontaram que, ao criar o diagrama de casos de uso e ao especificar os requisitos por casos de uso puderam observar vários pontos de reutilização de código.
- **Construção dos casos de teste -** Um ponto positivo levantado pelos os analistas de teste (4) foi a facilidade em descrever os casos de teste utilizando a especificação dos requisitos por casos de uso. Outro ponto também relatado, foi a facilidade de determinar os cenários de casos de teste graças a divisão dos casos de uso em *slices*, pois, conforme relatado pelos mesmos, a descrição dos casos de uso em conjunto com a divisão por *slices*, apresentou, de forma clara, o que deveria ser testado em cada *slice*. Esses pontos também contribuíram, conforme citado pelos analistas de teste, na realização dos testes exploratórios.
- **Compreensão do tamanho e do objetivo do produto a ser desenvolvido -** Todos os colaboradores (11) relataram que a aplicação da abordagem Use Case 2.0, em especial, a história de usuário, descrição de caso de uso e a divisão por *slices*, forneceu-lhes uma maior compreensão do objetivo principal do produto,

conseguindo assim, focar no que realmente era relevante a ser entregue para o cliente.

- **Facilidade na estimativa de esforço -** Outro ponto levantado por todos os colaboradores foi a facilidade em estimar o esforço com a divisão dos casos de uso em *slices*. Pois, pode-se compreender de forma clara e objetiva o que deveria ser desenvolvido e testado em cada *slice*.

6.3.1 Discussão

A partir dos resultados obtidos pode-se observar que foi fácil de compreender e aplicar a abordagem e que seus resultados foram benéficos a organização. A aplicação da abordagem proporcionou aos membros das equipes uma visão mais clara do objetivo do produto a ser desenvolvido (história de usuário) e como esse deveria funcionar (casos de uso). Porém, o destaque da abordagem foi a organização do desenvolvimento do produto em iterações proporcionado pela divisão dos casos de uso em *slices*, deixando claro o que seria entregue em cada parte. Além disso, essa divisão contribuiu para a estimativa de esforço (pontos de história) pois, explicitou o que seria entregue em cada iteração, facilitando a determinação do esforço para entregar cada parte. Essa clareza também proporcionou um aumento nos pontos de história produzidos durante as sprints em que o produto foi desenvolvido, caracterizando um aumentou na produtividade das equipes.

Contudo, observou-se que algumas atividades realizadas na aplicação da abordagem tiveram pouca ou nenhuma relevância a organização, como por exemplo, o PCU, um artefato difícil de se construir e, como a determinação dos produtos a serem desenvolvidos não são negociáveis em nível de produção, não se mostrou muito útil para a equipe.

A aplicação da abordagem demandou um esforço maior, principalmente da área da análise de software, que precisou realizar mais atividades para que abordagem fosse utilizada. Porém, como pode-se observar, grande parte desse esforço foi despendido para o planejamento, pesquisa e consultoria externa necessária inicialmente para a aplicação da abordagem e para realizar atividades que já eram desenvolvidas pelas equipes. Ou seja,

o esforço para a aplicação da abordagem foi pequeno e proporcionou inúmeras vantagens, e por isso, ambas as equipes continuarão a utilizar a abordagem.

6.4 AMEAÇAS À VALIDADE DE AVALIAÇÃO

Os resultados obtidos podem ter sido influenciados por diversos fatores, ameaçando assim, a validade desses resultados. Primeiramente, por ter sido realizado em uma só organização, com somente duas equipes e em um curto período de tempo, as conclusões apresentadas para o estudo não podem ser generalizadas. Além disso, durante a coleta dos resultados das perguntas 2.3, 3.2 e 3.3, a autora agrupou respostas as quais continham termos semelhantes ou sinônimos, com o intuito de apresentar resultados mais concisos para as dificuldades encontradas, técnicas e artefatos que tiveram maior aceitação e pontos positivos na aplicação da abordagem. No entanto, não foi utilizado nenhum método indutivo de mapeamento.

Outro fator que pode ameaçar a validade dos resultados é que colaboradores foram submetidos ao questionário perante a autora. Esse cenário, ao mesmo tempo que enriqueceu os dados coletados, também pode ter influenciado a opinião dos colaboradores sobre a impressão subjetiva da abordagem. Além disso, a autora desse estudo exerce suas funções profissionais dentro de um dos times selecionados no estudo de caso sendo responsável pela produção da documentação dos requisitos de uma das equipes. Por já estar inserida dentro do processo de desenvolvimento do estudo de caso diariamente, seu conhecimento implícito pode ter colaborado para a aplicação da abordagem, porém, isso pode também ter afetado os resultados e conclusões.

7 CONCLUSÃO

O presente trabalho descreva a experiência de duas equipes de desenvolvimento de software do projeto SISMOB, do laboratório Bridge, na aplicação da abordagem Use Case 2.0, avaliando o impacto da aplicação dessa abordagem nas áreas de análise, desenvolvimento e teste das equipes em que a abordagem foi aplicada. Este impacto foi destrinchado em medida de esforço despendido pelas áreas para a aplicação da abordagem

e custo-benefício da aplicação da abordagem. Desta forma, o objetivo é contribuir para a aplicação da abordagem Use Case 2.0 em outras organizações com contexto similares ao projeto SISMOB do laboratório Bridge.

Antes de iniciar a aplicação da abordagem, foi realizada uma análise dos principais conceitos relacionados ao tema desse trabalho, como por exemplo os métodos ágeis, com foco no *framework* SCRUM, especificação de requisitos utilizando casos de uso e abordagem Use Case 2.0, os quais ajudaram na elaboração da estratégia e aplicação da abordagem Use Case 2.0.

Seguindo princípio da abordagem Use Case 2.0 de adaptar-se as necessidades da equipe, foi realizada uma revisão do contexto em que as equipes em que seria aplicada a abordagem estavam inseridas. Foi definido também o processo de desenvolvimento dessas equipes, com o intuito de compreender como seria aplicada a abordagem e quais produtos de trabalho as equipes teriam que produzir.

Com base nas informações levantadas na fundamentação teórica, e a revisão de contexto, a abordagem começou a ser aplicada, iniciando pela sua apresentação as equipes em que essa seria aplicada e, posteriormente, definido em quais níveis os produtos de trabalho seriam desenvolvidos e quais ferramentas seriam utilizadas para o desenvolvimento desses produtos de trabalho.

A abordagem foi então aplicada durantes duas sprints de desenvolvimento das equipes selecionadas para o estudo. Durante o processo de desenvolvimento foram coletados dados de esforço despendido pelas equipes para a aplicação da abordagem. Após a finalização do desenvolvimento, a aplicação da abordagem foi avaliada para verificar qual o impacto causado nas equipes em que a mesma foi aplicada. Para realizar essa avaliação foi utilizado o método GQM e aplicado um questionário as áreas de análise, desenvolvimento e teste das duas equipes em que a abordagem foi aplicada.

O resultado dos esforços despendido pelas áreas e a avaliação apontam que é fácil de se aplicar a abordagem Use Case 2.0 em uma equipe com o contexto similar ao das equipes em que o estudo foi realizado. A divisão dos casos de uso por *slices* é o grande diferencial da abordagem, pois auxilia na organização da equipe explicitando o que será entregue em incremento, o que proporciona uma visão clara do andamento do projeto. Contudo, observa-se também que a aplicação da abordagem demanda um certo esforço,

principalmente da área de análise de software, pois a mesma é responsável pela produção da maioria dos produtos de trabalho necessário para a aplicação da abordagem.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTON, A. I. et al. **Deriving goals from a use-case based requirements specification**.

Requirements Engineering, v. 6, n. 1, p. 63-73, 2001.

BASSI, D. **Desafios de Requisitos em Métodos Ágeis: Uma Revisão Sistemática.** Master's Dissertation. USP, 2008.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 26 de ago. 2017.

BECK, Kent. **Embracing change with extreme programming.** Computer, v. 32, n. 10, p. 70-77, 1999.

BITTNER, Kurt. **Use case modeling**. Addison-Wesley Longman Publishing Co., Inc., 2002.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Elsevier Brasil, 2006.

BOURQUE, Pierre et al. **Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0**. IEEE Computer Society Press, 2014.

BRIDGE, Laboratório. Institucional. Disponível em : <https://bridge.ufsc.br/>. Acesso em: 27 de nov. de 2017.

CIRILLO, F. **The pomodoro technique**. Lulu. com, 2009.

COHN, M. **Agile estimating and planning**. Pearson Education, 2005.

DE MEDEIROS, Juliana Dantas Ribeiro Viana et al. **Engenharia de requisitos em projetos ágeis; uma revisão sistemática da literatura.** Revista Principia, v. 1, n. 28, p. 11-24, 2015.

HASTIE, Shane; WOJEWODA, Stéphane. **Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch.** Retrieved, v. 1, n. 15, p. 2016, 2015.

GRUBER, J. **Markdown.** Disponível em: <https://daringfireball.net/projects/markdown/>. Acesso em: 18 de mai. 2018.

HAUCK, J. **Use Case 2.0 - Incorporando Práticas Ágeis aos Casos de Uso. In: The Developers Conference.** 12 de mai. de 2016. Disponível em <https://pt.slideshare.net/JeanHauck/usecase-20-jean-hauck>. Acesso em: 20 de out. 2017

ISO/IEC/IEEE. **IEEE. 29148: 2011-Systems and software engineering-Requirements engineering**. Technical report, 2011.

JACOBSON, I.; SPENCE, I.; BITTNER, K.. **Use Case 2.0: The guide to succeeding with use cases.** Ivar Jacobson International, 2011.

JACOBSON, I.; SPENCE, I.; BITTNER, K.. **USE-CASE 2.0 - The Hub of Software Development**. ACM Queue, January-February, 2016

JAQUEIRA, A. et al. **Desafios de Requisitos em Métodos Ágeis: Uma Revisão Sistemática.** 3rd WBMA. São Paulo, 2012.

KAMEI, F. K. **Benefícios e Limitações das Metodologias Ágeis no Desenvolvimento de Software.** Master's Dissertation. UFPE, 2012.

KITCHENHAM, B.A. **Procedures for Performing Systematic Reviews**. **Tech. Report TR/SE- 0401**, Keele University. Inglaterra. 2004.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements engineering: processes and techniques**. Wiley Publishing, 1998.

KOZIOLEK, H. **Goal, question, metric. In Dependability metrics**. Springer Berlin Heidelberg. 2008. pp. 39-42.

LIKERT, R. **A technique for the measurement of attitudes**. Archives of Psychology, v. 22, n. 140, p. 55, 1932

LINGS, B.; LUNDELL, B. **On the adaptation of Grounded Theory procedures: insights from the evolution of the 2G method.** Information Technology & People, Vol. 18, n.3, 2005, p.196-211.

LOCONSOLE, A.; BORSTLER, J. **An industrial case study on requirements volatility measures.** In: Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific. IEEE, 2005. p. 8 pp.

OLIVEIRA, R. R. **A técnica de priorização MoSCoW.** Disponível em: <http://www.ronielton.eti.br/publicacoes/artigoprince2moscow2014_mpbr.pdf>. Acesso em: 18 de fev de 2018.

OMG. **Object Management Group. OMG Unified Modeling Language TM (OMG UML),** Version 2.5. Technical report formal/2015-03-01, 2015. Disponível em: <http://www.omg.org/spec/UML/2.5/>. Acesso em: 20 de out de 2017.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional.** 7ª Edição. **Ed: McGraw Hill**, 2011.

RUBIN, K. S. **Essential Scrum: A practical guide to the most popular Agile process**. Addison-Wesley, 2012.

RUNESON, P.; HÖST, M.. **Guidelines for conducting and reporting case study research in software engineering**. Empirical software engineering, v. 14, n. 2, p. 131, 2009.

SANTOS, Nuno et al. **Using Scrum together with UML models: A collaborative University-Industry R&D software project.** In: International Conference on Computational Science and Its Applications. Springer International Publishing, 2016. p. 480-495.

SCHWABER, Ken; BEEDLE, Mike. **Agile software development with Scrum**. Upper Saddle River: Prentice Hall, 2002.

SCHWABER, Ken; SUTHERLAND, Jeff. **The scrum guide.** Scrum Alliance, v. 21, 2011.

SOMMERVILLE, I. **Engenharia de Software.** 9. ed. São Paulo: Pearson Addison Wesley, 2011.

VERSIONONE. **The 10th Annual State of Agile Report. 2015**. Disponível em <https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf>. Acesso em: 26 de ago. de 2017.

WAZLAWICK, Raul S. **Análise e projeto de sistemas de informação orientados a objetos.** 2. ed. Rio de Janeiro: Elsevier, 2011.

WAZLAWICK, Raul S. **Engenharia de software: conceitos e práticas**. Rio de Janeiro: Elsevier, 2013.

YIN, Robert K. **Estudo de Caso : Planejamento e Métodos**. Bookman editora, 2011.

ANEXO A - DECLARAÇÃO DE CONCORDÂNCIA

DECLARAÇÃO DE CONCORDÂNCIA COM AS CONDIÇÕES PARA O DESENVOLVIMENTO DO TCC NA INSTITUIÇÃO

Declaro estar ciente das premissas para a realização de Trabalhos de Conclusão de Curso (TCC) de Ciência da Computação e Sistema de Informações da UFSC, particularmente da necessidade de que se o TCC envolver o desenvolvimento de um software ou produto específico (ex: um protocolo, um método computacional, etc.) o código fonte e/ou documentação completa correspondente deverá ser entregue integralmente, como parte integrante do relatório final do TCC.

Ciente dessa condição básica, declaro estar de acordo com a realização do TCC identificado pelos dados apresentados a seguir.

Instituição	Laboratório Bridge
Nome do responsável	Prof. Dr. Raul Sidnei Wazlawick
Cargo/Função	Coordenador geral
Fone de contato	(48) 3721-5991
Acadêmico(s)	Mariana Aparecida de Mattos
Título do trabalho	Aplicação de um Estudo de Caso com Use Case 2.0
Curso	Ciências da Computação/INE/UFSC

(local e data) Florianópolis, 23/05/18

Assinatura do responsável: 

APÊNDICE A – QUESTIONÁRIO APLICADO AS EQUIPES DO ESTUDO

QUESTIONÁRIO A

Este questionário foi desenvolvido pela acadêmica Mariana Aparecida de Mattos, aluna do curso de Ciências da Computação da Universidade Federal de Santa Catarina, como parte de seu Trabalho de Conclusão de Curso. Tem como proposta implantar e avaliar a abordagem Use Case 2.0 em uma unidade organizacional, por meio de um estudo de caso aplicado ao projeto SISMOB do laboratório Bridge da Universidade Federal de Santa Catarina.

Cargo: **Data:**

1. É fácil de compreender e utilizar a abordagem Use Case 2.0? Por quê?

2. Quais foram as principais dificuldades encontradas na aplicação da técnica Use Case 2.0?

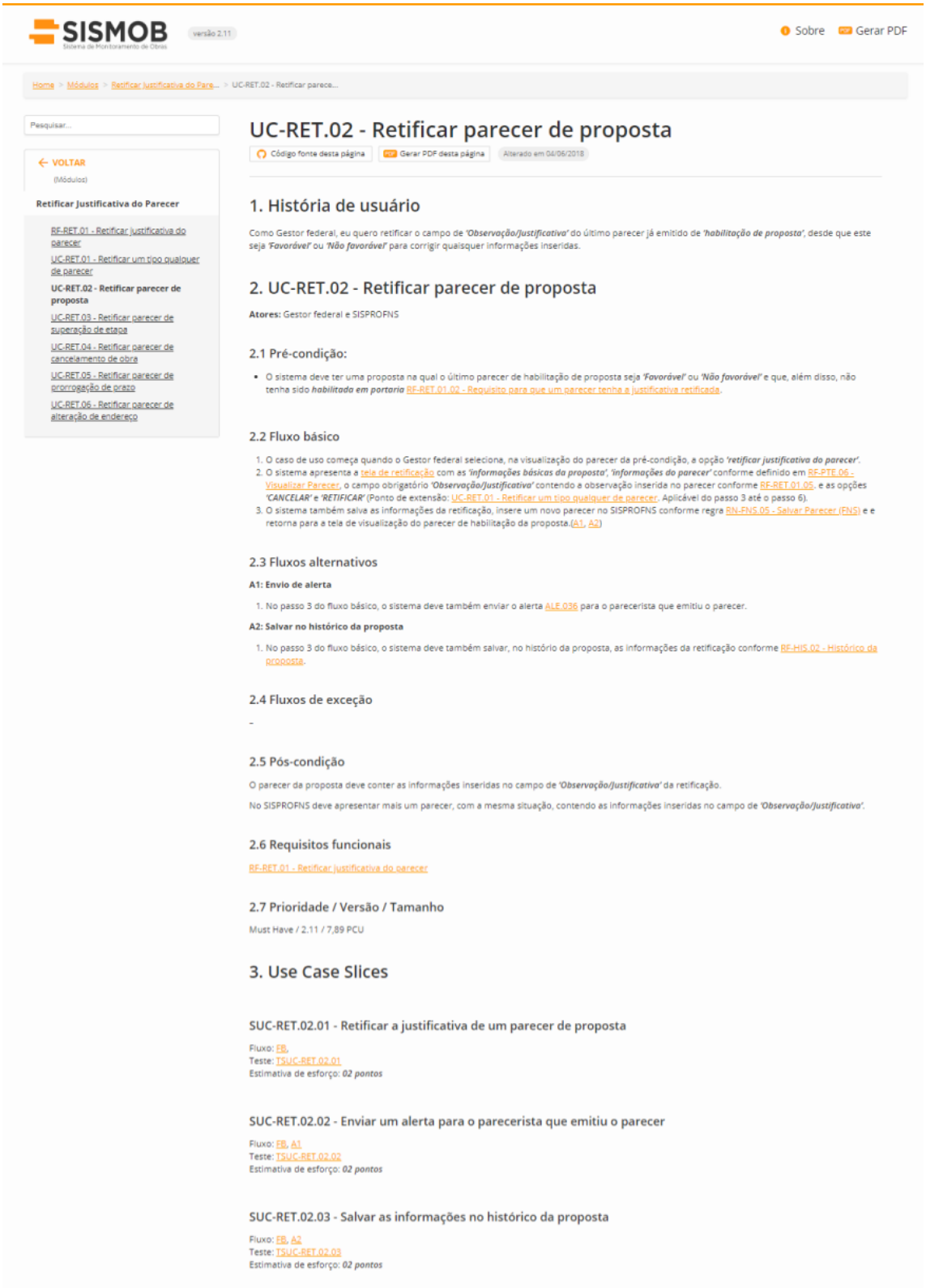
3. Você utilizaria a abordagem Use Case 2.0 em outro projeto? Por quê?

4. Quais foram as principais técnicas da abordagem Use Case 2.0 utilizada que você achou mais relevantes?

5. Quais foram os principais pontos positivos, na aplicação da abordagem Use Case 2.0?

6. Você considera que a experiência da aplicação da abordagem foi benéfica a unidade organizacional? Por quê?

APÊNDICE B – ARTEFATOS PRODUZIDOS NA APLICAÇÃO DA ABORDAGEM USE CASE 2.0



SISMOB versão 2.11 — Sistema de Monitoramento de Obras

Sobre | Gerar PDF

Home > Módulos > Retificar Justificativa do Pare... > UC-RET.02 - Retificar parece...

Pesquisar...

VOLTAR

(Módulos)

Retificar Justificativa do Parecer

- RF-RET.01 - Retificar justificativa do parecer
- UC-RET.01 - Retificar um tipo qualquer de parecer
- **UC-RET.02 - Retificar parecer de proposta**
- UC-RET.03 - Retificar parecer de superação de etapa
- UC-RET.04 - Retificar parecer de cancelamento de obra
- UC-RET.05 - Retificar parecer de prorrogação de prazo
- UC-RET.06 - Retificar parecer de alteração de endereço

UC-RET.02 - Retificar parecer de proposta

Código fonte desta página | Gerar PDF desta página | Alterado em 04/06/2018

1. História de usuário

Como Gestor federal, eu quero retificar o campo de ***'Observação/Justificativa'*** do último parecer já emitido de ***'habilitação de proposta'***, desde que este seja ***'Favorável'*** ou ***'Não favorável'*** para corrigir quaisquer informações inseridas.

2. UC-RET.02 - Retificar parecer de proposta

Atores: Gestor federal e SISPROFNS

2.1 Pré-condição:

- O sistema deve ter uma proposta na qual o último parecer de habilitação de proposta seja ***'Favorável'*** ou ***'Não favorável'*** e que, além disso, não tenha sido ***habilitada em portaria*** RF-RET.01.02 - Requisito para que um parecer tenha a justificativa retificada.

2.2 Fluxo básico

1. O caso de uso começa quando o Gestor federal seleciona, na visualização do parecer da pré-condição, a opção ***'retificar justificativa do parecer'***.
2. O sistema apresenta a tela de retificação com as ***'informações básicas da proposta'***, ***'informações do parecer'*** conforme definido em RF-PTE.06 - Visualizar Parecer, o campo obrigatório ***'Observação/Justificativa'*** contendo a observação inserida no parecer conforme RF-RET.01.05, e as opções ***'CANCELAR'*** e ***'RETIFICAR'*** (Ponto de extensão: UC-RET.01 - Retificar um tipo qualquer de parecer. Aplicável do passo 3 até o passo 6).
3. O sistema também salva as informações da retificação, insere um novo parecer no SISPROFNS conforme regra RN-FNS.05 - Salvar Parecer (FNS) e e retorna para a tela de visualização do parecer de habilitação da proposta.(A1, A2)

2.3 Fluxos alternativos

A1: Envio de alerta

1. No passo 3 do fluxo básico, o sistema deve também enviar o alerta ALE.036 para o parecerista que emitiu o parecer.

A2: Salvar no histórico da proposta

1. No passo 3 do fluxo básico, o sistema deve também salvar, no histório da proposta, as informações da retificação conforme RF-HIS.02 - Histórico da proposta.

2.4 Fluxos de exceção

-

2.5 Pós-condição

O parecer da proposta deve conter as informações inseridas no campo de ***'Observação/Justificativa'*** da retificação.

No SISPROFNS deve apresentar mais um parecer, com a mesma situação, contendo as informações inseridas no campo de ***'Observação/Justificativa'***.

2.6 Requisitos funcionais

RF-RET.01 - Retificar justificativa do parecer

2.7 Prioridade / Versão / Tamanho

Must Have / 2.11 / 7,89 PCU

3. Use Case Slices

SUC-RET.02.01 - Retificar a justificativa de um parecer de proposta

Fluxo: FB,
Teste: TSUC-RET.02.01
Estimativa de esforço: ***02 pontos***

SUC-RET.02.02 - Enviar um alerta para o parecerista que emitiu o parecer

Fluxo: FB, A1
Teste: TSUC-RET.02.02
Estimativa de esforço: ***02 pontos***

SUC-RET.02.03 - Salvar as informações no histórico da proposta

Fluxo: FB, A2
Teste: TSUC-RET.02.03
Estimativa de esforço: ***02 pontos***

Pesquisar...

VOLTAR
(Módulos)

Retificar Justificativa do Parecer

- RF-RET.01 - Retificar justificativa do parecer
- UC-RET.01 - Retificar um tipo qualquer de parecer
- UC-RET.02 - Retificar parecer de proposta
- **UC-RET.03 - Retificar parecer de superação de etapa**
- UC-RET.04 - Retificar parecer de cancelamento de obra
- UC-RET.05 - Retificar parecer de prorrogação de prazo
- UC-RET.06 - Retificar parecer de alteração de endereço

UC-RET.03 - Retificar parecer de superação de etapa

Código fonte desta página | Gerar PDF desta página | Alterado em 26/06/2018

1. História de usuário

Como Gestor federal, eu quero retificar o campo de *'Observação/Justificativa'* do último parecer já emitido de *'superação da etapa'*, desde que este seja *'Favorável'* ou *'Não favorável'* para corrigir quaisquer informações inseridas.

2. UC-RET.03 - Retificar do parecer de superação de etapa

Atores: Gestor federal e SISPROFNS

2.1 Pré-condição:

- O sistema deve ter uma proposta que o último parecer de superação de etapa seja *'Favorável'* ou *'Não favorável'* RF-RET.01 - Requisito para que um parecer tenha a justificativa retificada.

2.2 Fluxo básico

1. O caso de uso começa quando o Gestor federal seleciona, na visualização do parecer da pré-condição, a opção *'retificar justificativa do parecer'*.
2. O sistema apresenta a tela de retificação com as *'informações básicas da proposta'*, *'informações do parecer'* conforme definido em RF-PTE.06 - Visualizar Parecer, o campo obrigatório *'Observação/Justificativa'* contendo a observação inserida no parecer conforme RF-RET.01.05, e as opções *'CANCELAR'* e *'RETIFICAR'* (Ponto de extensão: UC-RET.01 - Retificar um tipo qualquer de parecer. Aplicável do passo 3 até o passo 6).
3. O sistema também salva as informações de retificação do parecer, insere um novo parecer no SISPROFNS conforme regra RN-FNS.05 - Salvar Parecer (FNS) e retorna para a tela de visualização do parecer de superação de etapa. (A1, A2)

2.3 Fluxos alternativos

A1: Envio de alerta

1. No passo 3 do fluxo básico, o sistema deve também enviar o alerta ALE.036 para o parecerista que emitiu o parecer.

A2: Salvar no histórico da proposta

1. No passo 3 do fluxo básico, o sistema deve também salvar, no histório da proposta, as informações da retificação conforme RF-HIS.02 - Histórico da proposta.

2.4 Fluxos de exceção

-

2.5 Pós-condição

O parecer da superação de etapa deve conter as informações inseridas no campo de *'Observação/Justificativa'* da retificação.

No SISPROFNS deve apresentar mais um parecer, com a mesma situação, contendo as informações inseridas no campo de *'Observação/Justificativa'*.

2.6 Requisitos funcionais

RF-RET.01 - Retificar justificativa do parecer

2.7 Prioridade / Versão / Tamanho

Must Have / 2.12 / 6,90 PCU

3. Use Case Slices

SUC-RET.03.01 - Retificar a justificativa de um parecer de superação de etapa

Fluxo: FB
Teste: TSUC-RET.03.01
Estimativa de esforço: ***02 pontos***

SUC-RET.03.02 - Enviar um alerta para o parecerista que emitiu o parecer

Fluxo: FB, A1
Teste: TSUC-RET.03.02
Estimativa de esforço: ***02 pontos***

SUC-RET.03.03 - Salvar as informações no histórico da proposta

Fluxo: FB, A2
<<<< HEAD Teste: TSUC-RET.03.03
Estimativa de esforço: ***02 pontos*** ======= Teste de aceitação:
Estimativa de esforço: ***02 pontos***

673f225d614c7b799f88cc9adaca9948ab595149

Pesquisar...

← VOLTAR
(Módulos)

Retificar Justificativa do Parecer

- RF-RET.01 - Retificar justificativa do parecer
- UC-RET.01 - Retificar um tipo qualquer de parecer
- UC-RET.02 - Retificar parecer de proposta
- UC-RET.03 - Retificar parecer de superação de etapa
- **UC-RET.04 - Retificar parecer de cancelamento de obra**
- UC-RET.05 - Retificar parecer de prorrogação de prazo
- UC-RET.06 - Retificar parecer de alteração de endereço

UC-RET.04 - Retificar parecer de cancelamento de obra

Código fonte desta página | Gerar PDF desta página | Alterado em 18/06/2018

1. História de usuário

Como Gestor federal, eu quero retificar o campo de ***'Observação/Justificativa'*** do último parecer já emitido de ***'alteração de endereço'***, desde que este seja ***'Favorável'*** ou ***'Não favorável'*** para corrigir quaisquer informações inseridas.

2. UC-RET.04 - Retificar parecer de cancelamento de obra

Atores: Gestor federal e SISPROFNS

2.1 Pré-condição:

- O sistema deve ter uma proposta que o último parecer de cancelamento da obra seja ***'Favorável'*** ou ***'Não favorável'***.

RF-RET.01 - Requisito para que um parecer tenha a justificativa retificada.

2.2 Fluxo básico

1. O caso de uso começa quando o Gestor federal seleciona, na visualização do parecer da pré-condição, a opção ***'retificar justificativa do parecer'***.(A1)
2. O sistema apresenta a tela de retificação com as ***'informações básicas da proposta'***, ***'informações do parecer'*** conforme definido em RF-PTE.06 - Visualizar Parecer; o campo obrigatório ***'Observação/Justificativa'*** contendo a observação inserida no parecer conforme RF-RET.01.05, e as opções ***'CANCELAR'*** e ***'RETIFICAR'*** (Ponto de extensão: UC-RET.01 - Retificar um tipo qualquer de parecer. Aplicável do passo 3 até o passo 6).
3. O sistema salva as informações de retificação do parecer e retorna para a tela de visualização do parecer de cancelamento da obra. (A2,A3)

2.3 Fluxos alternativos

A1: Envio de alerta

1. No passo 3 do fluxo básico, o sistema deve também enviar o alerta ALE.036 para o parecerista que emitiu o parecer.

A2: Salvar no histórico da proposta

1. No passo 3 do fluxo básico, o sistema deve também salvar, no histório da proposta, as informações da retificação conforme RF-HIS.02 - Histórico da proposta.

2.4 Fluxos de exceção

-

2.5 Pós-condição

O parecer do cancelamento da obra deve conter as informações inseridas no campo de ***'Observação/Justificativa'*** da retificação.

No histório da proposta, deve ser apresendo as informações da retificação conforme RF-HIS.02 - Histórico da proposta.

2.6 Requisitos funcionais

RF-RET.01 - Retificar justificativa do parecer

2.7 Prioridade/ Versão/ Tamanho

Must Have / 2.12 / 9,37 PCU

3. Use Case Slices

SUC-RET.04.01 - Enviar um alerta para o parecerista que emitiu o parecer

Fluxo: FB, A1
Teste: TSUC-RET.04.02
Estimativa de esforço: ***02 pontos***

SUC-PTE.04.02 - Salvar as informações no histórico da proposta

Fluxo: FB, A2
Teste: TSUC-RET.04.03
Estimativa de esforço: ***02 pontos***

Pesquisar...

VOLTAR
(Módulos)

Retificar Justificativa do Parecer

- RF-RET.01 - Retificar justificativa do parecer
- UC-RET.01 - Retificar um tipo qualquer de parecer
- UC-RET.02 - Retificar parecer de proposta
- UC-RET.03 - Retificar parecer de superação de etapa
- UC-RET.04 - Retificar parecer de cancelamento de obra
- **UC-RET.05 - Retificar parecer de prorrogação de prazo**
- UC-RET.06 - Retificar parecer de alteração de endereço

UC-RET.05 - Retificar parecer de prorrogação de prazo

Código fonte desta página | Gerar PDF desta página | Alterado em 04/06/2018

1. História de usuário

Como Gestor federal, eu quero retificar o campo de ***'Observação/Justificativa'*** do último parecer já emitido de ***'prorrogação de prazo'***, desde que este seja ***'Favorável'*** ou ***'Não favorável'*** para corrigir qualsquer informações inseridas.

2. UC-RET.05- Retificar parecer de prorrogação de prazo

Atores: Gestor federal

2.1 Pré-condição:

- O sistema deve ter uma proposta que o último parecer de alteração de endereço seja ***'Favorável'*** ou ***'Não favorável'*** RF-RET.01 - Requisito para que um parecer tenha a justificativa retificada.

2.2 Fluxo básico

1. O caso de uso começa quando o Gestor federal seleciona, na visualização do parecer da pré-condição, a opção ***'retificar justificativa do parecer'***.
2. O sistema apresenta a tela de retificação com as ***'informações básicas da proposta'***, ***'informações do parecer'*** conforme definido em RF-PTE.06 - Visualizar Parecer, o campo obrigatório ***'Observação/Justificativa'*** contendo a observação inserida no parecer conforme RF-RET.01.05, e as opções ***'CANCELAR'*** e ***'RETIFICAR'*** (Ponto de extensão:UC-RET.01 - Retificar um tipo qualquer de parecer. Aplicável do passo 3 até o passo 6).
3. O sistema salva as informações da retificação do parecer e retorna para a tela de visualização do parecer de prorrogação de prazo. (A1,A2)

2.3 Fluxos alternativos

A1: Envio de alerta

1. No passo 3 do fluxo básico, o sistema deve também enviar o alerta ALE.036 para o parecerista que emitiu o parecer.

A2: Salvar no histórico da proposta

1. No passo 3 do fluxo básico, o sistema deve também salvar, no histórico da proposta, as informações da retificação conforme RF-HIS.02 - Histórico da proposta.

2.4 Fluxos de exceção

-

2.5 Pós-condição

O parecer da prorrogação de prazo deve conter as informações inseridas no campo de ***'Observação/Justificativa'*** da retificação.

2.6 Requisitos funcionais

RF-RET.01 - Retificar justificativa do parecer

2.7 Prioridade/ Versão/ Tamanho

Should Have / 2.12 / 5,93 PCU

3. Use Case Slices

SUC-RET.05.01 - Retificar a justificativa de um parecer de prorrogação de prazo

Fluxo: FB
Teste: TSUC-RET.05.01
Estimativa de esforço: ***01 pontos***

SUC-RET.05.02 - Enviar um alerta para o parecerista que emitiu o parecer

Fluxo: FB, A1
Teste: TSUC-RET.05.02
Estimativa de esforço: ***02 pontos***

SUC-PTE.05.03 - Salvar as informações no histórico da proposta

Fluxo: FB, A2
Teste: TSUC-RET.05.03
Estimativa de esforço: ***02 pontos***

Pesquisar...

VOLTAR
(Módulos)

Retificar Justificativa do Parecer

- RF-RET.01 - Retificar justificativa do parecer
- UC-RET.01 - Retificar um tipo qualquer de parecer
- UC-RET.02 - Retificar parecer de proposta
- UC-RET.03 - Retificar parecer de superação de etapa
- UC-RET.04 - Retificar parecer de cancelamento de obra
- UC-RET.05 - Retificar parecer de prorrogação de prazo
- **UC-RET.06 - Retificar parecer de alteração de endereço**

UC-RET.06 - Retificar parecer de alteração de endereço

Código fonte desta página | Gerar PDF desta página | Alterado em 04/06/2018

1. História de usuário

Como Gestor federal, eu quero retificar o campo de ***'Observação/Justificativa'*** do último parecer já emitido de ***'alteração de endereço'***, desde que este seja ***'Favorável'*** ou ***'Não favorável'*** para corrigir quaisquer informações inseridas.

2. UC.RET.06 - Retificar parecer de alteração de endereço

Atores: Gestor federal

2.1 Pré-condição:

- O sistema deve ter uma proposta que o último parecer de alteração de endereço seja ***'Favorável'*** ou ***'Não favorável'*** RF-RET.01 - Requisito para que um parecer tenha a justificativa retificada.

2.2 Fluxo básico

1. O caso de uso começa quando o Gestor federal seleciona, na visualização do parecer da pré-condição, a opção ***'retificar justificativa do parecer'***.
2. O sistema apresenta a tela de retificação com as ***'informações básicas da proposta'***, ***'informações do parecer'*** conforme definido em RF-PTE.06 - Visualizar Parecer, o campo obrigatório ***'Observação/Justificativa'*** contendo a observação inserida no parecer conforme RF-RET.01.05, e as opções ***'CANCELAR'*** e ***'RETIFICAR'*** (Ponto de extensão: UC-RET.01 - Retificar um tipo qualquer de parecer. Aplicável do passo 3 até o passo 6).
3. O sistema salva as informações da retificação do parecer e retorna para a tela de visualização do parecer de alteração de endereço.(A1,A2)

2.3 Fluxos alternativos

A1: Envio de alerta

1. No passo 3 do fluxo básico, o sistema deve também enviar o alerta ALE.036 para o parecerista que emitiu o parecer.

A2: Salvar no histórico da proposta

1. No passo 3 do fluxo básico, o sistema deve também salvar, no histório da proposta, as informações da retificação conforme RF-HIS.02 - Histórico da proposta.

2.4 - Fluxos de exceção

-

2.5 Pós-condição

O parecer da alteração deve conter as informações inseridas no campo de ***'Observação/Justificativa'*** da retificação.

2.6 Requisitos funcionais

RF-RET.01 - Retificar justificativa do parecer

2.7 Prioridade/ Versão/ Tamanho

Should Have / 2.12 / 5,93 PCU

3. Use Case Slices

SUC-RET.06.01 - Retificar a justificativa de um parecer de alteração de endereço

Fluxo: FB
Teste: TSUC-RET.06.01
Estimativa de esforço: ***01 pontos***

SUC-RET.06.02 - Enviar um alerta para o parecerista que emitiu o parecer

Fluxo: FB, A1
Teste: TSUC-RET.06.02
Estimativa de esforço: ***02 pontos***

SUC-RET.06.03 - Salvar as informações no histórico da proposta

Fluxo: FB, A2
Teste: TSUC-RET.06.03
Estimativa de esforço: ***02 pontos***

Pesquisar...

VOLTAR
(Módulos)

Alteração de Endereço

- RF-AEN.01 - Solicitar Alteração de Endereço
- RF-AEN.02 - Visualizar Solicitação de Alteração de Endereço
- RF-AEN.03 - Alterar Endereço (MS)
- RF-AEN.04 - Visualizar Alteração de Endereço
- UC-AEN.01 - Alteração de endereço

Alteração de Endereço

Código fonte desta página | Gerar PDF desta página | Alterado em 22/05/2018

1. Casos de uso

1.1 Diagrama de casos de uso

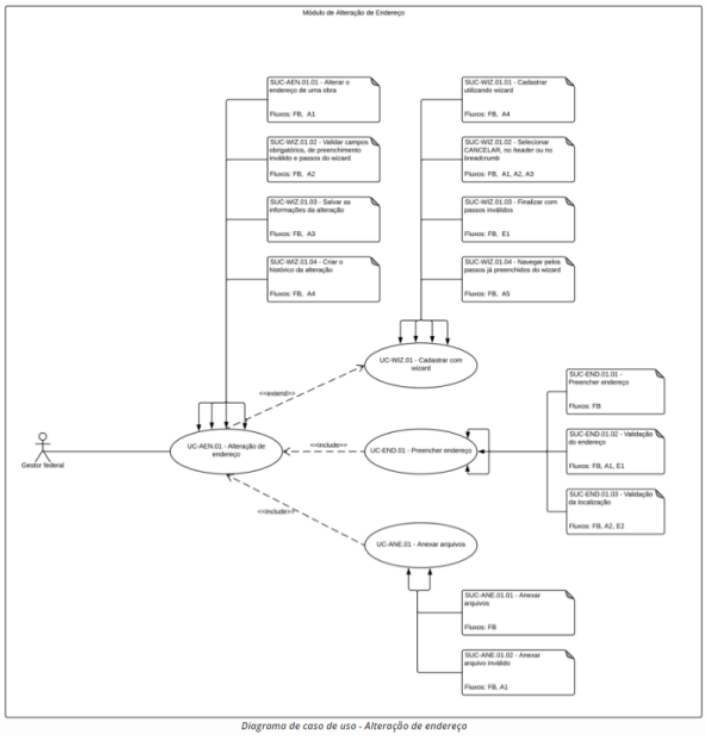


Diagrama de caso de uso - Alteração de endereço

1.2 Casos de uso descritivos

UC-AEN.01 - Alteração de endereço

2. Requisitos funcionais

RF-AEN.01 - Solicitar Alteração de Endereço
RF-AEN.02 - Visualizar Solicitação de Alteração de Endereço
RF-AEN.03 - Alterar Endereço (MS)
RF-AEN.04 - Visualizar Alteração de Endereço

3. Regras de negócio

Pesquisar...

VOLTAR
(Geral)

Casos de Uso

UC-ANE.01 - Anexar arquivos
UC-END.01 - Preencher endereço
UC-WIZ.01 - Cadastrar com wizard

UC-WIZ.01 - Cadastrar com wizard

Código fonte desta página | Gerar PDF desta página | Alterado em 22/05/2018

1. História do usuário

-

2. UC-WIZ.01 - Cadastrar com wizard

O sistema deve permitir cadastros utilizando Wizard.

Este caso de uso está relacionado ao Casos de Uso CRUDL.

Atores: Usuário do sistema

2.1 Pré-condição:

1. O Usuário deve ter feito ***login*** e obtido autorização do sistema
2. O Usuário deve ter selecionado um novo cadastro que utiliza o Wizard

2.2 Fluxo básico

1. O caso de uso começa quando o Usuário seleciona um novo cadastro que utiliza o wizard
2. O Sistema apresenta a tela de cadastro no primeiro passo do wizard com as opções ***'Cancelar'*** e ***'Avançar'***
3. O Usuário preenche as informações do primeiro passo e ***avança*** (A1, A2, A3)
4. O Sistema apresenta os passos seguintes com as respectivas telas do cadastro com as opções ***'Cancelar'***, ***'Voltar'*** e ***Avançar***
5. O Usuário preenche as informações necessárias de cada passo do wizard e ***avança*** (A1, A2, A3, A4, A5)
6. O Sistema apresenta a tela de revisão no último passo com as opções ***Cancelar***, ***Voltar*** e ***Finalizar***
7. O Usuário revisa as informações e ***finaliza*** (A1, A2, A3, A4, A5)
8. O Sistema exibe a mensagem padrão de sucesso e retorna para a tela anterior ao cadastro (A6)

2.3 Fluxos alternativos

A1: CANCELAR - Voltar

1. O Usuário seleciona a opção ***'Cancelar'***, ou em qualquer outra funcionalidade no ***header*** ou no ***breadcrumb***
2. O Sistema apresenta a mensagem MODAL.001 com as opções ***'Voltar'*** e ***'Sair sem salvar'***
3. O Usuário seleciona a opção ***'Voltar'***
4. A mensagem é fechada e o Sistema permanece na tela atual

A2: CANCELAR - Sair sem salvar

1. O Usuário seleciona a opção ***'Cancelar'***
2. O Sistema apresenta a mensagem MODAL.001 com as opções ***'Voltar'*** e ***'Sair sem salvar'***
3. O Usuário seleciona a opção ***'Sair sem salvar'***
4. A mensagem é fechada e o Sistema retorna para a tela anterior ao cadastro

A3: SELECIONAR outra funcionalidade fora do wizard - Sair sem salvar

1. O Usuário seleciona qualquer outra funcionalidade fora do wizard (no ***header*** ou ***breadcrumb*** por exemplo)
2. O Sistema apresenta a mensagem MODAL.001 com as opções ***'Voltar'*** e ***'Sair sem salvar'***
3. O Usuário seleciona a opção ***'Sair sem salvar'***
4. A mensagem é fechada e o Sistema apresenta a tela da funcionalidade selecionada no passo 1 deste fluxo

A4: VOLTAR

1. O Usuário seleciona a opção ***'Voltar'***
2. O Sistema volta para o passo anterior do wizard com todas as informações anteriores preenchidas

A5: NAVEGAR pelos passos já preenchidos do wizard

1. O Usuário seleciona algum passo já preenchido do wizard
2. O Sistema apresenta a tela do passo selecionado com as respectivas opções
3. O Usuário altera as informações necessárias e ***avança***
4. O Sistema invalida os passos relacionados aos campos alterados e apresenta a tela do passo seguinte do wizard (Passo 4 ou 6 do Fluxo básico)

A6: VALIDAR os passos do wizard

1. O Sistema valida os passos do wizard, exibe a mensagem padrão de sucesso e retorna para a tela anterior ao cadastro (E1)

2.4 Fluxos de exceção

E1: FINALIZAR com passos do wizard inválidos

1. O Sistema encontra algum passo inválido do wizard e apresenta a mensagem MODAL.074
2. O Usuário seleciona o passo inválido
3. O Sistema apresenta a tela do passo selecionado e retorna para o passo 5 do Fluxo básico

2.5 Pós-condição

-

2.6 Requisitos funcionais

2.7 Prioridade / Versão / Tamanho

Must Have / 2.10 / 7,41 PCU

3. Use Case Slices

SUC-WIZ.01.01 - Cadastrar utilizando wizard

Fluxo: FB, A4
Teste de aceitação:
Estimativa de trabalho: ***08 pontos***

SUC-WIZ.01.02 - Selecionar CANCELAR, no *header* ou no *breadcrumb*

Fluxo: FB, A1, A2, A3
Teste:
Estimativa de trabalho: ***03 pontos***

SUC-WIZ.01.03 - Finalizar com passos inválidos

Fluxo: FB, E1
Teste:
Estimativa de trabalho: ***03 pontos***

SUC-WIZ.01.04 - Navegar pelos passos já preenchidos do wizard

Fluxo: FB, A5
Teste:
Estimativa de trabalho: ***02 pontos***

Pesquisar...

VOLTAR
(Geral)

Casos de Uso

UC-ANE.01 - Anexar arquivos
UC-END.01 - Preencher endereço
UC-WIZ.01 - Cadastrar com wizard

UC-END.01 - Preencher endereço

Código fonte desta página | Gerar PDF desta página | Alterado em 22/05/2018

1. História do usuário

Como usuário do sistema, eu desejo preencher as informações de endereço e a respectiva localização geográfica, com o intuito de cadastrá-lo no sistema.

2. UC-END.01 - Preencher endereço

Este caso de uso estende o Caso de Uso CRUDL.

Atores: Usuário do sistema

2.1 Pré-condição:

1. O Usuário deve ter feito *login* e obtido autorização do sistema
2. O Usuário acessa a tela de preencher endereço

2.2 Fluxo básico

1. O caso de uso começa quando o Usuário acessa a tela de endereço
2. O Usuário preenche o campo CEP com valor válido (A1)
3. O Sistema consulta e valida o CEP. Preenche os campos município, UF, logradouro, latitude e longitude, e marca o ponto no mapa da localização correspondente ao CEP informado.
4. O Usuário preenche o número e o complemento
5. O Sistema valida as informações inseridas do endereço conforme RF-CAD.01 - Informações de Endereço, atualiza a marcação no mapa e os campos de latitude e longitude de acordo com o número informado (E1)
6. O Usuário finaliza o cadastro (A2)
7. O caso de uso é finalizado

2.3 Fluxos alternativos

A1: VALIDAR endereço

1. O Usuário preenche o campo CEP com valor inválido
2. O Sistema consulta, invalida o CEP e apresenta a respectiva mensagem de validação conforme RF-CAD.01 - Informações de Endereço
3. O Usuário preenche ou altera os campos UF, Município, Logradouro, Número e Complemento correspondentes à faixa do CEP informado
4. O Sistema valida as informações inseridas do endereço conforme RF-CAD.01 - Informações de Endereço, preenche os campos de latitude e longitude e marca o ponto no mapa da localização geográfica (E1)
5. O Sistema retorna ao passo 6 do Fluxo básico

A2: ALTERAR localização geográfica

1. O Usuário altera a marcação no mapa ou altera os campos de latitude e longitude com valor válido em relação ao endereço informado e finaliza o cadastro
2. O Sistema valida a localização geográfica conforme RF-CAD.04 - Localização e retorna ao passo 7 do Fluxo básico (E2)

2.4 Fluxos de exceção

E1: VALIDAR endereço

1. O Sistema invalida as informações inseridas do endereço, apresenta as mensagens de validação conforme RF-CAD.01 - Informações de Endereço e retorna para o passo 1 do fluxo alternativo A1

E2: VALIDAR localização geográfica

1. O Sistema invalida a localização geográfica, apresenta as mensagens de validação conforme RF-CAD.04 - Localização e retorna para o passo 1 do fluxo alternativo A2

2.5 Pós-condição

-

2.6 Requisitos funcionais

RF-CAD.01 - Informações de Endereço

2.7 Prioridade / Versão / Tamanho

Must Have / 2.11 / 7,41 PCU

3. Use Case Slices

SUC-END.01.01 - Preencher endereço

Fluxo: FB
Teste: TSUC-AEN.01.01
Estimativa de trabalho: ***08 pontos***

SUC-END.01.02 - Validação do endereço

Fluxo: FB, A1, E1
Teste: TSUC-END.01.02
Estimativa de trabalho: ***05 pontos***

SUC-END.01.03 - Validação da localização

Fluxo: FB, A2, E2
Teste: SUC-END.01.03 Estimativa de trabalho: ***05 pontos***

Pesquisar...

← VOLTAR
(Módulos)

Alteração de Endereço

RF-AEN.01 - Solicitar Alteração de Endereço
RF-AEN.02 - Visualizar Solicitação de Alteração de Endereço
RF-AEN.03 - Alterar Endereço (MS)
RF-AEN.04 - Visualizar Alteração de Endereço
UC-AEN.01 - Alteração de endereço

UC-AEN.01 - Alteração de endereço

Código fonte desta página | Gerar PDF desta página | Alterado em 22/05/2018

1. História do usuário

Para manter os dados atualizados da obra, como Gestor federal, quero poder alterar em qualquer momento, o endereço de uma obra.
Só poderei atualizar o endereço de obras de ***construção*** em que sua proposta já foi habilitada em portaria.
Deverei informar uma justificativa, o novo endereço com a sua localização geográfica e anexar documentos e fotografias que comprovem o novo endereço informado.
A alteração do endereço de uma obra não poderá ser feita pelo MS caso exista uma solicitação aberta de alteração de endereço enviada pelo proponente.

2. UC-AEN.01 - Alteração de endereço

Este caso de uso estende o Caso de Uso UC-WIZ.01 - Cadastrar com wizard.

Atores: Gestor federal

2.1 Pré-condição:

1. Ter um perfil de Gestor federal
2. Ter uma proposta de construção habilitada em portaria

2.2 Fluxo básico

1. O caso de uso estende o UC-WIZ.01 - Cadastrar com wizard e começa quando o Gestor federal seleciona a opção ***'Alterar endereço'***
2. O Sistema apresenta a tela do passo de justificativa e as opções ***'Cancelar'*** e ***'Avançar'*** (E1)
3. O Gestor federal preenche a justificativa e ***avança*** (A2)
4. O Sistema apresenta a tela do passo de novo endereço e as opções ***'Cancelar'***, ***'Voltar'*** e ***Avançar*** (Inclui UC-END.01 - Preencher endereço)
5. O Gestor federal preenche as informações do endereço e ***avança*** (A2)
6. O Sistema apresenta a tela do passo de anexos e as opções ***'Cancelar'***, ***'Voltar'*** e ***'Avançar'*** (Inclui UC-ANE.01 - Anexar arquivos)
7. O Gestor federal anexa os arquivos e ***avança*** (A2)
8. O Sistema apresenta a tela do passo de revisão e as opções ***'Cancelar'***, ***'Voltar'*** e ***'Finalizar'***
9. O Gestor federal revisa as informações e ***finaliza***
10. O Sistema apresenta a mensagem de confirmação MODAL.121 com as opções ***'Cancelar'*** e ***'Confirmar alteração'***
11. O Gestor federal ***confirma a alteração*** (A1)
12. O Sistema exibe a exibe a mensagem padrão de sucesso e retorna para a tela anterior à alteração de endereço (A3, A4)

2.3 Fluxos alternativos

A1: CANCELAR - Modal de confirmação

1. O Gestor federal seleciona a opção ***'Cancelar'*** do modal de confirmação
2. A mensagem é fechada e o Sistema permanece na tela atual

A2: AVANÇAR sem preencher campos obrigatórios e/ou com campos preenchidos inválidos

1. O Usuário não preenche algum campo obrigatório e/ou preenche algum campo com valor inválido e seleciona a opção ***'Avançar'***
2. O Sistema permanece na tela atual e apresenta a mensagem de validaçao conforme RF-AEN.03 - Alterar Endereço (MS)

A3: SALVAR as informações da alteração

1. O Sistema salva o registro, exibe a mensagem padrão de sucesso e retorna para a tela anterior à alteração de endereço

A4: CRIAR o histórico da alteração

1. O Sistema cria o histórico da alteração de endereço, exibe a mensagem padrão de sucesso e retorna para a tela anterior à alteração de endereço

2.4 Fluxos de exceção

E1: Selecionar a opção de *'Alterar endereço'* e existir uma solicitação aberta

1. O Sistema verifica e encontra uma solicitação de alteração de endereço aberta para a obra
2. O Sistema mostra a mensagem de validação conforme definido em RF-AEN.03 - Alterar Endereço (MS) e retorna para o passo 1 do Fluxo Básico

2.5 Pós-condição

Na visualização e impressão da obra, deve conter o registro da alteração com todas as informações inseridas.

As informações de endereço da proposta, localização, documentos e fotografias devem estar atualizadas com as novas informações inseridas na alteração de endereço (RF-PRO.03 - Visualizar Proposta).

No histórico da proposta, deve conter o registro da alteração de endereço.

2.6 Requisitos funcionais

RF-AEN.03 - Alterar Endereço (MS)

2.7 Prioridade / Versão / Tamanho

Must Have/ 2.11 / 7,41 PCU

3. Use Case Slices

SUC-AEN.01.01 - Alterar o endereço de uma obra

Fluxo: FB, A1
Teste de aceitação: TSUC-AEN.01.01
Estimativa de esforço: ***05 pontos***

SUC-WIZ.01.02 - Validar campos obrigatórios, de preenchimento inválido e passos do wizard

Fluxo: FB, A2
Teste de aceitação: TSUC-AEN.01.02
Estimativa de trabalho: ***03 pontos***

SUC-WIZ.01.03 - Salvar as informações da alteração

Fluxo: FB, A3
Teste de aceitação: TSUC-WIZ.01.03
Estimativa de esforço: ***08 pontos***

SUC-WIZ.01.04 - Criar o histórico da alteração

Fluxo: FB, A4
Teste de aceitação: TSUC-WIZ.01.04
Estimativa de esforço: ***08 pontos***

APÊNDICE C – ARTIGO DA MONOGRAFIA

Estudo de Caso com Use Case 2.0

Mariana Aparecida de Mattos[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC - Brasil

[2]Laboratório Bridge – Universidade Federal de Santa Catarina (UFSC) – Florianopolis, SC - Brasil

mariana.mattos@grad.ufsc.br

Abstract. *The methodology is increasingly conquering the space within software development. A recent survey by VersionOne, a massive growth in methodologies methodology. , A Standish Group report, only 29% of the projects in 2015 were successfully delivered. Among the factors for success, there are no reports, the main problems are related to lack of data and lack of customer participation. Despite the importance of eliciting requirements for the success of software software, such as methodologies for bureaucratic activities, making the project less agile. In order to solve this problem, in 2011, a Use Case 2.0 approach, a version of the use-case technique, emerged as an agile and scalable practice for capturing requirements and development aids. Dought from this scenario, this work aims to deploy and evaluate a Use Case 2.0 approach in an organizational unit. The research is done through a case study applied in the laboratory of the Federal University of Santa Catarina. To the first, the revision is the fundamental problems to the subject already was the review of literature. From the data acquired, an evaluation of the parameter definition method for the organization of the SISMOB project of the laboratory is carried out. Next, the techniques of approach of Use Case 2.0 in a team of the SISMOB project are applied and the evaluations of the obtained results are made.*

Resumo. *A metodologia está conquistando cada vez mais o espaço dentro do desenvolvimento de software. Uma pesquisa realizada recentemente pela VersionOne, um crescimento massivo na metodologia de metodologias. , Um relatório do Grupo Standish, somente 29% dos projetos em 2015 foram entregues com sucesso. Dentre os fatores para o sucesso, não há relatórios, os principais problemas são relacionados à falta de dados e a falta de participação do cliente. Apesar da importância da elicitação de requisitos para o sucesso do software de software, como metodologias para as atividades burocráticas, tornando-se o projeto menos ágil. Com o intuito de resolver esse problema, em 2011, surgiu uma abordagem de Caso de Uso 2.0, uma versão da técnica de casos de uso, que vem como uma prática ágil e escalável para capturar requisitos e auxiliares de desenvolvimento. Dought desse scenario, this work tem o objetivo de implantar e avaliar uma abordagem Use Case 2.0 em uma unidade organizacional. A pesquisa é feita através de um estudo de caso*

aplicado no laboratório da Universidade Federal de Santa Catarina. To the first, the revision is the fundamental problems to the theme already was a review of literature. A partir dos dados adquiridos, é realizada uma avaliação do método de definição de parâmetros para a organização do projeto SISMOB do laboratório. Em seguida, são aplicadas as técnicas de abordagem do Caso de Uso 2.0 em uma equipe do projeto SISMOB e, são feitas as avaliações dos resultados obtidos.

1. Introdução

Os métodos ágeis vêm ganhando cada vez mais espaço no desenvolvimento de software, o que desperta interesse tanto de profissionais da área como de pesquisadores (JAQUEIRA et al., 2012). Uma pesquisa realizada entre julho e novembro de 2015 pela VersionOne envolvendo mais de 3.000 profissionais da área de TI, revelou um crescimento massivo na adoção de práticas ágeis nas organizações, onde 95% dos entrevistados revelaram utilizar abordagens ágeis na maioria dos seus projetos (VERSIONONE, 2015).

Apesar de mais de uma década ter passado desde o manifesto, estudos ainda apontam problemas em projetos que usam abordagens ágeis, como expectativa do cliente não atendidas e dificuldade em estimar prazos e orçamentos (KAMEI, 2012; READ; BRIGGS, 2012, apud MEDEIROS et al., 2015).

Mesmo sabendo da importância da elicitação e documentação de requisitos no sucesso do desenvolvimento do software e na minimização dos riscos do projeto, estas atividades são vistas pela metodologia ágil como burocráticas, tornando o processo menos ágil. Sendo assim, a falta de documentação é um dos seus principais desafios (JAQUEIRA et al., 2012).

Com o intuito de solucionar o problema da engenharia de requisitos e os métodos ágeis, novas técnicas surgiram para auxiliar a análise de requisitos em um ambiente ágil. Neste cenário, a abordagem Use Case 2.0, vem como uma prática escalável e ágil que usa casos de uso para capturar um conjunto de requisitos e impulsionar o desenvolvimento incremental de um sistema. Criado por Ivar Jacobson, Ian Spence e Kurt Bittner, o *Use-Case 2.0 - The Guide to Succeeding with Use Cases*, nada mais é do que uma evolução da técnica de casos de uso. Entre as atualizações propostas, as principais são a utilização de história de usuário para a definição do escopo do caso de uso, e o *slice,* ou corte vertical dos casos de uso, que definem um conjunto de fatias do caso de uso (JACOBSON et al., 2011).

Dentro deste contexto, este artigo apresenta um estudo de caso da abordagem Use Case 2.0 para especificação de requisitos no projeto SISMOB do laboratório Bridge da Universidade Federal de Santa Catarina. Será apresentada toda a experiência relacionada a aplicação da abordagem e seus resultados, visando verificar o impacto da abordagem na organização. Por fim, direções futuras para o trabalho serão sugeridas.

2. Fundamentação Teórica

2.1. Engenharia de requisitos

Os requisitos são as descrições das funcionalidades de um sistema, seus serviços e restrições de funcionamento, refletindo a necessidade dos usuários para um sistema que serve a uma determinada finalidade (SOMMERVILLE, 2011). Na literatura existem várias propostas de procedimentos de análise e gerenciamento da engenharia de requisitos (KOTONYA e SOMMERVILLE, 1998). Este trabalho baseia-se nos fundamentos do modelo proposto por Sommerville (2011) que descreve a ER como sendo composta por quatro atividades de alto nível, para o levantamento de requisitos, sendo estas, **estudo de viabilidade** - busca avaliar se um sistema é útil para a empresa; **elicitação e análise de requisitos** - visa descobrir os requisitos; **especificação de requisitos** - converte os requisitos levantados em uma forma padrão; **validação de requisitos** - realiza a validação dos requisitos junto ao cliente.

As atividades no processo de requisitos não são feitas em apenas uma sequência. Na prática, a engenharia de requisitos é um processo iterativo, inter-relacionados, com retroalimentação, em que as atividades são organizadas em forma de espiral, conforme Figura 1 e, são repetidas até a aceitação de um documento de requisitos. A quantidade de esforço dedicado a cada atividade em cada iteração depende do estágio do processo e de que tipo de sistema está sendo desenvolvido. Caso algum problema seja encontrado ao longo do desenvolvimento do sistema, as etapas de elicitação, especificação, validação e documentação entram novamente na espiral. Isso ocorre tantas vezes quantas forem necessárias, até que não haja mais inconsistências nos requisitos. Nesse momento, o documento final de requisitos é elaborado e inicia-se o processo de gerenciamento de requisitos que visa compreender e controlar as mudanças nos requisitos de um sistema (SOMMERVILLE, 2011).

A etapa de estudo de viabilidade, também conhecida como etapa de concepção, serve como ponto de partida para as demais atividades da engenharia de requisitos. Seu objetivo é um entendimento do problema a ser tratado pelo software a ser desenvolvido, quais são os *stakeholders* e a natureza da solução desejada (PRESSMAN, 2011). Nesta etapa, é realizada uma estimativa acerca das possibilidades de se satisfazerem as necessidades do usuário identificado, usando as tecnologias atuais, a fim de definir se haverá um avanço do projeto. (SOMMERVILLE, 2011).

A elicitação também conhecida como captura, descoberta, aquisição ou levantamento de requisitos, procura descobrir os requisitos, identificar limites de sistema e os *stakeholders*. Compreender o domínio do aplicativo, necessidades do negócio, restrições do sistema, *stakeholders* e o problema em si é parte essencial para entender o sistema a ser desenvolvido (BOURQUE, 2014). Segundo Wazlawick (2011), a etapa de elicitação de requisitos visa buscar todas as informações possíveis sobre as funções que o sistema deve executar e as restrições sobre as quais o sistema deve operar.

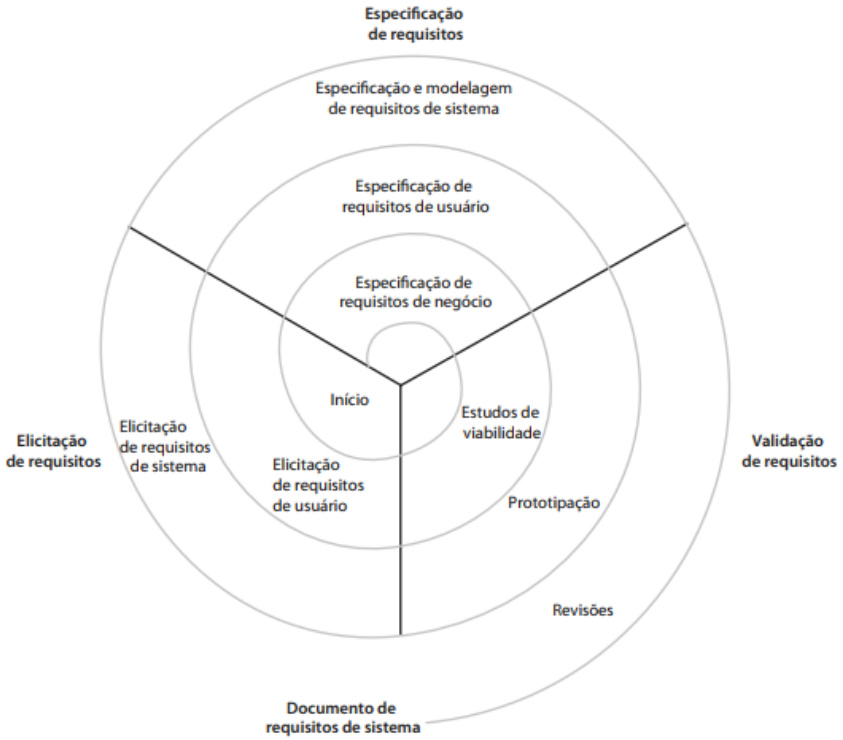


Figure 1 - Espiral do processo de Engenharia de Requisitos
Fonte: SOMMERVILLE, 2011

A informações obtidas nas etapas de viabilidade e elicitação são expandidas e refinadas durante a análise e negociação de requisitos. Nesta etapa os requisitos são detalhadamente analisados para identificar inconsistências, relação de dependência e conflitos, identificar se os requisitos são funcionais ou não funcionais e, classificar os requisitos de acordo com as categorias a que pertencem (PRESSMAN, 2011).

Posterior a fase de análise e negociação, a especificação de requisitos refere-se a produção de um documento que possa ser sistematicamente revisado, avaliado e aprovado (BOURQUE, 2014). Segundo Pressman (2011) especificação é o produto de trabalho final, podendo ser um documento escrito, um modelo gráfico, um modelo matemático formal, uma coleção de cenários de uso, um protótipo ou qualquer combinação desses elementos.

A etapa de validação consiste em verificar se o documento de requisitos produzido na etapa anterior realmente contempla as necessidades do cliente/usuário (SOMMERVILLE, 2011). Esta etapa tem por objetivo detectar faltas de consistência, omissão, ambiguidade ou falta de conformidade nos requisitos já especificados (PRESSMAN, 2011).

A gestão de requisitos é um processo paralelo às etapas anteriores, e está relacionado ao gerenciamento de mudanças. Segundo Sommerville (2011), as mudanças nos requisitos são inevitáveis, isso porque, os sistemas geralmente são desenvolvidos para

resolver problemas que não podem ser completamente definidos. Sendo assim, este processo tem como objetivo de compreender e controlar das mudanças nos requisitos do sistema (SOMMERVILLE, 2011).

2.2. Métodos ágeis

Métodos ágeis, baseiam-se em uma abordagem incremental para a especificação, desenvolvimento e entrega do produto. Destinam-se a entregar o software rapidamente, funcionando, sendo que podem ser incluídas alterações e novos requisitos a serem desenvolvidos nas iterações posteriores do sistema (SOMMERVILLE, 2011). Apesar dos métodos ágeis possuírem diferentes processos, eles compartilham um conjunto de 12 princípios de agilidade, estabelecidos pelo Agile Alliance.

Segundo Pressman (2011), nem todo o modelo de processo ágil aplicam esses 12 princípios atribuindo pesos iguais, alguns modelos preferem dar maior importância a um ou mais desses princípios. Entretanto, esses 12 princípios definem a agilidade mantida por cada um dos modelos de processo.

Dentre os métodos ágeis existentes, o SCRUM é um dos dos métodos ágeis mais conhecidos e utilizados atualmente (SOMMERVILLE, 2011), sendo um *framework* que tem como base o empirismo, ou seja, o conhecimento é obtido através de experiências anteriores e as decisões são tomadas de acordo com esse conhecimento (SCHWABER, 2002). Usado para resolver problemas complexos e adaptativo, o Scrum auxilia na produção e criatividade e entrega dos produtos com o maior valor possível. Scrum é considerado um framework leve, simples de entender e difícil de dominar (SCHWABER, 2002).

Segundo Pressman, 2011, o Scrum possui uma série de valores, conceitos e práticas que tem como objetivo maximizar as chances de sucesso do projeto. A seguir são apresentados, conforme Figura 2, seus principais componentes:

- **Eventos** - Sprint, Sprint Planning Meeting, Daily Scrum, Sprint Review Meeting e Sprint Retrospective
- **Artefatos -** Product Backlog, Sprint Backlog, Burndown Backlog, Taskboard
- **Time Scrum -** Product Owner, Scrum Master, Time de Desenvolvimento.

Figure 2 - Características de cada Sprint
Fonte: RUBIN, 2012 - Adaptado

2.3. Use Case e Use Case 2.0

Casos de uso são uma maneira simples e poderosa de expressar os requisitos funcionais ou comportamentos de um sistema (BITTNER, 2002). São processos de interação com o sistema que tem início e fim em tempo contíguo, ou seja, são executados muito rapidamente (WAZLAWICK, 2011). Segundo Jacobson (2011), caso de uso pode ser visto como uma sequência de ações realizadas pelo sistema para produzir um resultado observável de valor para um determinado usuário. O conjunto de todos os casos de uso definem todas as formas de uso do sistema e o seu valor (JACOBSON et al., 2011).

Os casos de uso são uma técnica de elicitação de requisitos introduzida inicialmente pela metodologia de software OOSSE - *Object-Oriented Software Engineering*, como descrito no livro *Object-Oriented Software Engineering: A Use Case Driven Approach* (JACOBSON et al., 1993 apud SOMMERVILLE, 2011). Tornando-se, posteriormente, uma característica fundamental da linguagem de modelagem unificada (UML - do inglês unified modeling language) (SOMMERVILLE, 2011).

A definição de casos de uso, na ótica da UML, é: “uma descrição de um conjunto de sequências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator.” (OMG, 2015). A UML também fornece uma representação gráfica de um caso de uso, conforme Figura 3. Nessa representação, as elipses representam casos de uso, os bonecos representam os atores e o retângulo representa o sistema (BOOCH et al, 2000).

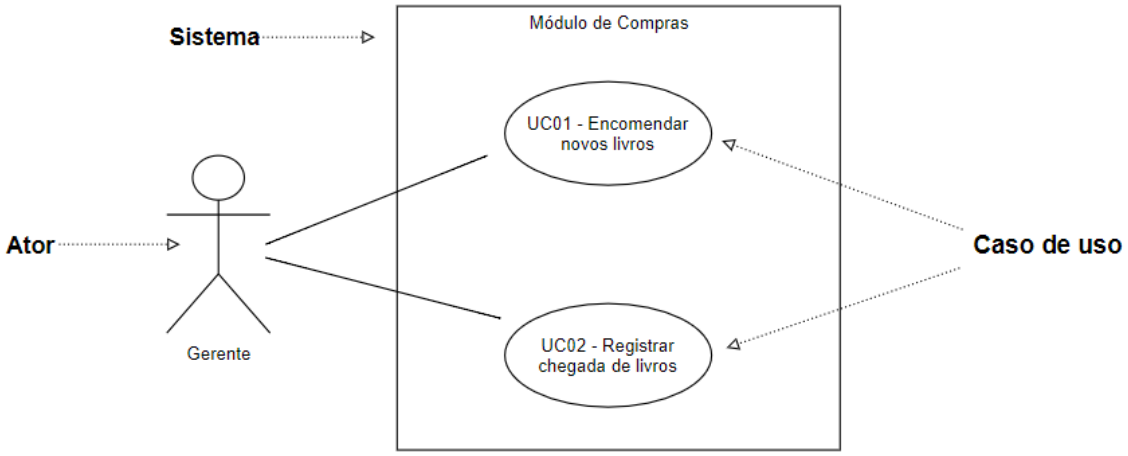


Figure 3 - Diagrama de casos de uso da UML
Fonte: WAZLAWICK, 2011 - Adaptado

Em dezembro de 2011, Ivar Jacobson, Ian Spence e Kurt Bittner publicaram o conceito de Use Case 2.0 no livro *USE-CASE 2.0 - The Guide Succeeding with Use Case.* Este novo conceito apresenta uma técnica escalável e ágil para desenvolver requisitos e conduzir o desenvolvimento incremental do sistema (JACOBSON et al., 2011).

Use Case 2.0 é uma conexão entre o conceito de Caso de Uso (descrito na seção 2.3) e a abordagem ágil de desenvolvimento (seção 2.2) utilizando o SCRUM (seção 2.2.2). Os conceitos do caso de uso e da abordagem ágil são os mesmos, porém, a combinação inteligente fornece uma visão geral do produto (JACOBSON et al, 2011).

A abordagem Use Case 2.0 não se refere a uma atualização do conceito de caso de uso, mas sim uma mudança na forma como os desenvolvedores de software e os analistas de negócio aplicam os casos de uso. O conceito de caso de uso não é alterado no Use Case 2.0, mas a forma como é apresentado e gerenciado evoluiu para aumentar a eficácia. (JACOBSON et al., 2011)

A abordagem Use Case 2.0, Jacobson et al. (2011), defini 6 princípios básico para que a aplicação dos casos de uso seja bem sucedida, sendo eles, (i) Mantenha-se simples contando história; (ii) Compreenda o "*the big Picture*"; (iii) Concentre-se no valor; (iv) Construa sistema em *slices;* (v) Entrega em incrementos; (vi) Adapte-se às necessidades da organização. A Use Case 2.0 mistura conceitos de abordagem ágil com conceitos de caso de uso. A Figura 4 mostra como esses conceitos estão relacionados entre si e como mudanças e defeitos os influenciam.

A Use Case 2.0 se concentra nos requisitos do usuário, ou seja, no que deve ser desenvolvido para atender os seus objetivos. Os testes comprovam se os objetivos foram ou não atendidos. A base do Use Case 2.0 são os casos de uso, histórias e fatias de caso uso (JACOBSON et al., 2011). Como pode ser observado na Figura 4, os *stakeholders* fornecem os requisitos que são capturados como um conjunto de casos de uso e

gerenciados e endereçados como um conjunto de fatias de casos de uso (JACOBSON et al., 2011).

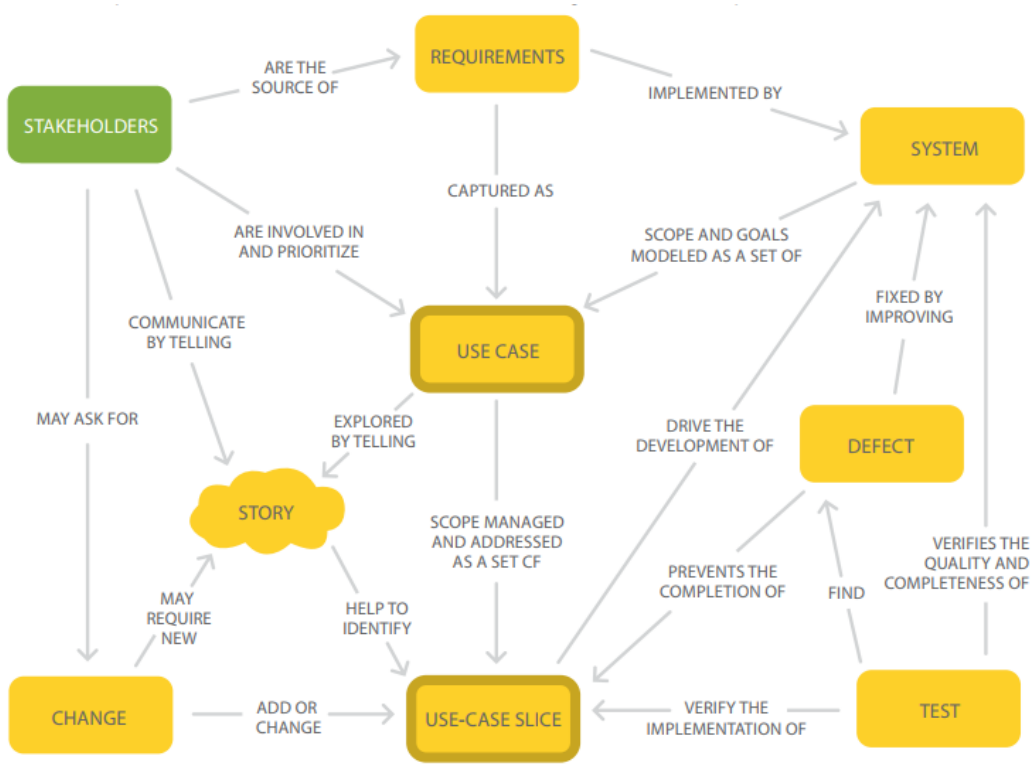


Figure 4 - Mapa conceitual do Use Case 2.0
Fonte: JACOBSON, 2011

3. Mapeamento da literatura

Com o objetivo de levantar o estado da sobre a utilização da abordagem Use Case 2.0 para Engenharia de Requisitos, buscando saber quais os resultados observados pelas organizações que utilizam essa abordagem. Portanto, para essa etapa, foi definida a seguinte pergunta: *"Quais os resultados na utilização da abordagem Use Case 2.0, para Engenharia de requisitos, aplicada a um ambiente real de desenvolvimento ágil?"*.

Critérios de inclusão/exclusão: Somente são incluídos materiais que apresentem experiências práticas de utilização da abordagem Use Case 2.0, em que a organizações tenham sido realizadas as experiências utilizem métodos ágeis de desenvolvimento e que os materiais contenham os resultados observados na aplicação da abordagem. Não são considerados materiais em que a organizações utilizem métodos tradicionais de desenvolvimento ou que não utilizem a abordagem Use Case 2.0.

Ferramentas utilizadas: A bases a serem pesquisadas são o Google Scholar[13], o portal de pesquisas da CAPES[14] e o IEEE Xplore[15].

Idioma de busca: Inglês e Português.

String de busca	Encontrados	Analisados	Potencialmente relevantes	Relevantes
("Use case 2.0") AND ("Requirements engineering" OR "Software requirements elicitation" OR "Software requirements analysis") AND ("Company" OR "Organization" OR "Organizational Unit") OR ("Agile software development")	1	1	0	0
("Use case 2.0") AND ("Company" OR "Organization" OR "Organizational Unit") OR ("Agile software development")	85	85	16	0

Utilizando as *strings* e ferramentas de busca definidas acima, não foram encontrados resultados relevantes que se enquadrassem nos critérios de aceitação. Acredita-se que isso ocorre pois, o tema proposto é relativamente recente. Além disso, há uma limitação no idioma em que foram feitas as buscas, nas ferramentas utilizadas e na *string* proposta.

4. Estudo de Caso

O objetivo deste trabalho é implantar e avaliar a abordagem Use Case 2.0 no projeto SISMOB do Laboratório Bridge. A partir desse objetivo, a seguinte pergunta de pesquisa para este estudo de caso foi definida: "*Como a aplicação das técnicas de Use Case 2.0 impactam na análise, desenvolvimento e teste de um time de desenvolvimento de software?".* Segundo LINGS e LUNDELL (2005) avaliação de impacto é a análise

[13] https://scholar.google.com.br
[14] http://www.periodicos.capes.gov.br/
[15] https://ieeexplore.ieee.org/Xplore/home.jsp

das consequências de alguma modificação efetuada em algum processo ou organização. Para esse estudo, o impacto foi derivado em termos de: (i) esforço despendido pelas equipes do estudo, (ii) produtividade das equipes em que o estudo está sendo aplicado e (iii) custo-benefício da implantação da abordagem aplicada no estudo.

Primeiramente foi realizada uma análise de contexto do contexto da unidade organizacional. Em seguida, foi elaborado o planejamento do estudo de caso utilizando a abordagem GQM - *Goal/Question/Metric* (KOZIOLEK, 2008). Posteriormente, foi definido com as equipes do estudo uma estratégia de implantação da abordagem Use Case 2.0. Então, foi feita a aplicação da abordagem Use Case 2.0 e realizado o acompanhamento das equipes em que a abordagem está sendo aplicada para verificação do impacto e análise dos resultados. Em paralelo, dados são coletados sobre a realização do estudo de caso.

A execução desse estudo de caso se dá no período entre Fevereiro a Maio de 2018. Foram selecionados dois times de desenvolvimento do projeto SISMOB. O time "D" é composto por 6 membros e, o time "F" por 5 membros.

Para a avaliação deste estudo, foram definidos os seguintes objetivos de medição, utilizando a abordagem GQM:

- **Objetivo de Medição 1 -** Analisar o esforço e a produtividade da equipe na aplicação das técnicas de Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.
- **Objetivo de Medição 2 -** Analisar as dificuldades para aplicação das técnicas da abordagem Use Case 2.0, sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.
- **Objetivo de Medição 3 -** Analisar a aceitação das técnicas da abordagem Use Case 2.0 utilizadas sob o ponto de vista das áreas de análise, desenvolvimento e teste de software no contexto do projeto SISMOB do laboratório Bridge.

A partir de cada objetivo de medição foram derivadas perguntas e medidas de forma a satisfazer as necessidades de informações deste estudo. Para o objetivo 1, por exemplo, foi derivada a pergunta "Quanto esforço cada área demandou na aplicação da abordagem Use Case 2.0?", com a respectiva medida "Proporção de esforço despendido em cada área em relação ao total de horas despendido para a aplicação da abordagem Use Case 2.0". Todas as outras perguntas e medidas foram derivadas dos objetivos de medição da mesma forma.

4.1. Contextualização

O laboratório Bridge atua na pesquisa e desenvolvimento de softwares voltados a gestão da saúde pública. Criado em 2012, foi instituído oficialmente em 2016 e está ligado ao Centro Tecnológico da Universidade Federal de Santa Catarina (BRIDGE, 2017). Dentro os projetos realizados no Bridge, existe o SISMOB (Sistema de Monitoramento de Obras), um software que tem como objetivo monitorar todas as obras

de engenharia e infraestrutura de estabelecimentos de saúde que são financiados com recursos federais, cuja transferência ocorra fundo a fundo, tornando-se uma ferramenta para o gerenciamento de todas as fases da obra. Para atender esse objetivo, o SISMOB possuí inúmeros recurso.

Neste estudo de caso, foi realizada a aplicação da abordagem Use Case 2.0 em dois times de desenvolvimento, "D" e "F" constituído por analistas de sistemas, desenvolvedores de software e analistas de teste. Além disso, as equipes compartilham um designer que atende a todo o projeto, um *Product Owner*, papel desempenhado pelo Gerente de projeto e, o papel de *Scrum Master* é desempenhado pelo analista de sistemas de cada equipe.

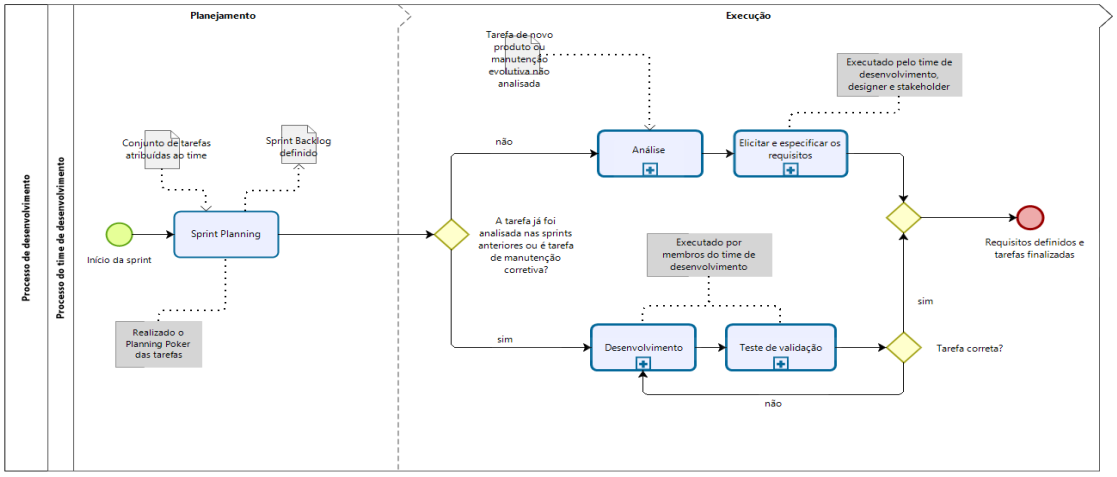


Figure 5 - Processo de desenvolvimento das equipes do estudo

O processo do time de desenvolvimento[16] é dividido em duas partes, planejamento e execução (Figura 5). Com duração de duas semanas, ao final da sprint, as tarefas do *Sprint Backlog* devem estar implementadas e testadas pelo time de desenvolvimento e, caso haja tarefas de manutenção evolutiva e novo produto para a próxima sprint, essas devem estar devidamente refinadas e especificadas pelo analista de sistemas do time.

4.2. Execução do estudo de caso

Para a execução deste estudo de caso, inicialmente, criou-se uma **estratégia de implantação da abordagem Use Case 2.0** em ambas as equipes. Nessa estratégia foram definidos os níveis de detalhamento de cada produto de trabalho proposta pela abordagem e as ferramentas utilizadas para desenvolver esses produtos. Essa definição ocorreu levando em consideração o valor agregado em cada nível e o que já era implantado em cada um dos times de desenvolvimento, seguindo o princípio do Use Case 2.0 de adaptar-se à realidade da equipe. Ambas as equipes definiram o nível de detalhamento para os produtos de trabalho conforme Figura 6.

[16] Os processos dos dois times de desenvolvimento selecionados são similares, por isso, houve uma abstração e será feita apenas uma descrição, destacando as diferenças no decorrer do texto.

Na segunda parte, foi realizada a **aplicação da abordagem Use Case 2.0** nas equipes do estudo de caso. Esta parte foi dividida em duas etapas, sendo a primeira realizada entre 19 de fevereiro à 11 de maio. Nesta primeira etapa foram realizadas as seguintes atividades

1. Elicitação de requisitos, definição dos requisitos funcionais, regras de negócio e definição do mockups.
2. Construção do diagrama de casos de uso, definição da história de usuário e descrição da narrativa de casos de uso.

Figure 6 - Níveis de detalhamento dos produtos de trabalho da abordagem Use Case 2.0 aplicado pelas equipes do estudo de caso

3. Divisão dos *slices* para cada caso de uso
4. Definição do tamanho e priorização dos casos de uso.

Ao final dessas atividades, foram inseridos também, os diagramas de casos de uso foram revisitados para que os *slices* fossem inseridos em cada caso de uso.

Na segunda etapa foram realizadas as seguintes atividades:

1. Construção do documento de casos de teste agrupados por *slices.*
2. Reunião para a definição dos *slices* a serem desenvolvidos.
3. Construção do *taskboard* com itens para cada *slice.*
4. Avaliação do impacto do sistema para o desenvolvimento dos *slices.*
5. Desenvolvimento dos *slices* e revisão do código implementado.
6. Teste dos *slices* desenvolvidos.

Ao final de todas essas atividades foram produzidos, além do sistema, alguns artefatos como por exemplo, o diagrama de casos de uso com *slices* associados, Figura 7.

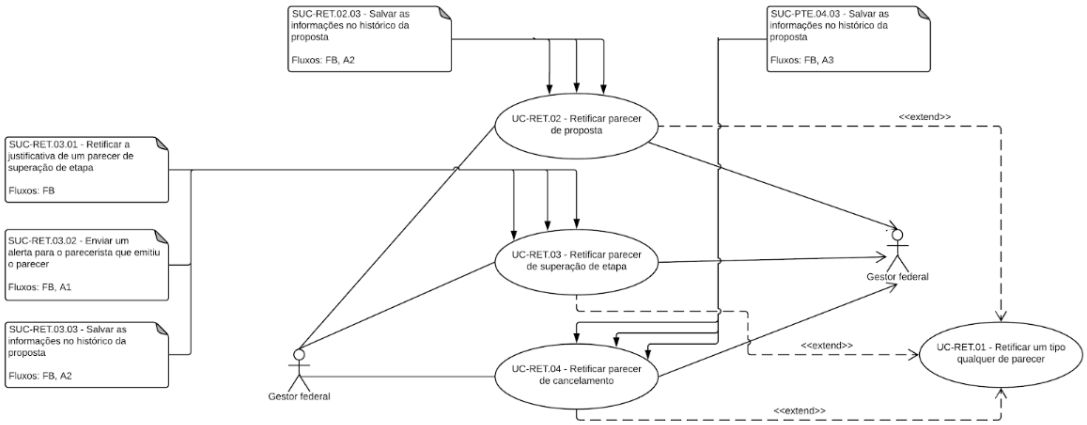


Figure 7 - Diagrama de casos de uso com *slice* associado

4.3 Análise e coleta de dados

O período de coleta de dados iniciou-se em 09 de Abril de 2018, sendo utilizadas, no total, 4 sprints de duas semanas. Durante a execução do estudo de caso as métricas foram coletadas por sprints dos dois times de desenvolvimento. Nas duas primeiras sprints, nenhuma intervenção foi realizada para a coletada de dados e, foram coletadas apenas as estimativas de esforço antes da aplicação da abordagem Use Case, 2.0. Nas duas sprints seguintes foram aplicadas as técnicas da abordagem Use Case 2.0 em ambas as equipes.

O esforço total despendido para a para desenvolver os produtos durante utilizando a abordagem Use Case 2.0 no time de desenvolvimento “D” foi de 292,81 horas e, para o time de desenvolvimento “F” foi de 230,38 horas (Figura 8). Tal esforço foi executado no período de Fevereiro à Maio de 2018, com duração de 4 meses, compreendendo todas as atividades relacionadas à pesquisa, planejamento e execução da abordagem Use Case 2.0 e, consultoria externa.

Ao considerar somente os produtos de trabalho propostos pela abordagem Use Case 2.0, o esforço aplicado na estratégia de implantação da abordagem e o esforço despendido nas pesquisas e consultorias externas. Verifica-se que para o time de desenvolvimento “D” despendeu um esforço total 138,42 horas e para o time de desenvolvimento “F” consumiu um esforço total 109,74 horas. Ou seja, o percentual de horas despendidas no desenvolvimento dos produtos propostos pela abordagem Use Case 2.0, em relação a total de horas utilizadas para a execução de todo o produto foi de 47,27% para o time “D” e 47,67% para o time “F”.

Em contrapartida, várias dos produtos de trabalho propostos pelo Use Case 2.0, já eram realizadas no processo atual de desenvolvimento de produto das equipes conforme descrito, sendo assim o percentual de esforço necessário para o desenvolvimento dos produtos de trabalho necessário para a aplicação da abordagem que ainda não eram

desenvolvidos pelas equipes foi de 17,22% (50,42 horas) para a equipe “D” e 20,76% (47,83 horas) para a equipe “F”.

Figure 8 - Esforço total para o desenvolvimento
Equipe "D"

O esforço demandado por cada área do time de desenvolvimento para aplicação da abordagem Use Case 2.0 pode ser observado nas Figuras 9. Para esta medição foram considerados apenas o esforço despendido pelas áreas para a aplicação dos novos produtos de trabalho, o esforço aplicado na estratégia de implantação da abordagem e o esforço despendido nas pesquisas e consultorias externas. Como pode ser observado na nas Figuras 28 e 29, a área de análise de software foi a que demandou maior parte das horas despendidas para a aplicação da abordagem Use Case 2.0, sendo responsável por aproximadamente 85,91 % (118,91 homens/horas) para o time de desenvolvimento “D” e 85,51% (93,91 homens/hora) para o time de desenvolvimento “F”.

Para 63,64% dos colaboradores (7) a abordagem Use Case 2.0 é de **fácil** de compreensão e utilização e, 27,27% dos colaboradores (3) afirmaram ser **muito fácil** compreender e utilizar a abordagem**.** Contudo, 9,09% dos colaboradores (1), declarou que a abordagem **não é fácil e nem difícil** de compreender e utilizar. Além disso, 100% dos colaboradores (11) afirmaram que a aplicação da abordagem Use Case 2.0 é **benéfica** a unidade organizacional. Sendo que, desse 54,55% dos colaboradores (6) afirmaram que a aplicação da abordagem Use Case 2.0 é **muito benéfica** para a organização.

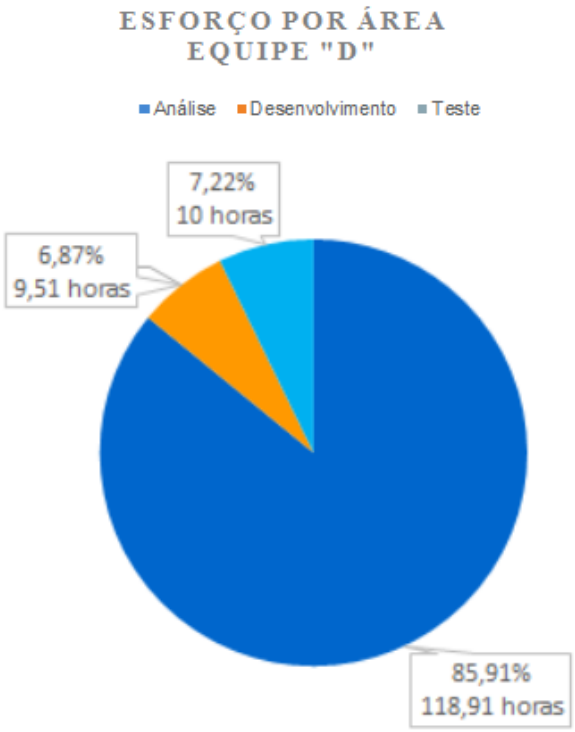


Figure 9 - Proporção do esforço demandado por cada área do time de desenvolvimento "D" para aplicação do Use Case 2.0

Entre os pontos levantados como dificuldade para implantação da abordagem, destaca-se os três fatores: (i) Aumento nas atividades de especificação de requisitos; (ii) dificuldades na divisão por slices (iii) Dificuldade na organização no taskboard da equipe.

Dentre todas as técnicas e artefatos produzidos na abordagem Use Case 2.0 as citadas pelos membros das equipes que participaram da entrevista, agrupadas pela autora por termo de similaridade e ordenado por número de citações:

#	**Técnicas e artefatos**	**Ocorrências**
1	Divisão dos casos de uso por *slices*	11
2	Descrição dos casos de uso	9
3	Estimativa de esforço por *slice*	7
4	Histórias de usuário	4
5	Diagrama de casos de uso	2

Os principais pontos positivos relatados pelos entrevistados foram:

- Organização na documentação de requisites.
- Especificação de requisitos focados no comportamento do sistema.
- Rastreabilidade entre o documento de teste e o documento de requisitos.
- Reutilização de código explicitada pelo diagrama de atividades.
- Construção dos casos de teste facilitada pela descrição de casos de uso.
- Compreensão clara do tamanho e do objetivo do produto a ser desenvolvido.
- Facilidade na estimativa de esforço.

4.3. Discussão

A partir dos resultados obtidos pode-se observar que foi fácil de compreender e aplicar a abordagem e que seus resultados foram benéficos a organização. A aplicação da abordagem proporcionou aos membros das equipes uma visão mais clara do objetivo do produto a ser desenvolvido (história de usuário) e como esse deveria funcionar (casos de uso). Porém, o destaque da abordagem foi a organização do desenvolvimento do produto em iterações proporcionado pela divisão dos casos de uso em *slices*, deixando claro o que seria entregue em cada parte. Além disso, essa divisão contribuiu para a estimativa de esforço (pontos de história) pois, explicitou o que seria entregue em cada iteração, facilitando a determinação do esforço para entregar cada parte. Essa clareza também proporcionou um aumentou a pontos de história produzidos durante as sprints em que o produto foi desenvolvido, caracterizando um aumentou na produtividade das equipes.

Contudo, observou-se que algumas atividades realizadas na aplicação da abordagem tiveram pouca ou nenhuma relevância a organização, como por exemplo, o PCU, um artefato difícil de se construir e, como a determinação dos produtos a serem desenvolvidos não são negociáveis em nível de produção, não se mostrou muito útil para a equipe.

A aplicação da abordagem demandou um esforço maior, principalmente da área da análise de software, que precisou realizar mais atividades para que abordagem fosse utilizada. Porém, como pode-se observar, grande parte desse esforço foi despendido para o planejamento, pesquisa e consultoria externa necessária inicialmente para a aplicação da abordagem e para realizar atividades que já eram desenvolvidos pelas equipes. Ou seja, o esforço para a aplicação da abordagem foi pequeno e proporcionou inúmeras vantagens, e por isso, ambas as equipes continuarão a utilizar a abordagem.

5. Conclusão

O presente trabalho descreva a experiência de duas equipes de desenvolvimento de software do projeto SISMOB, do laboratório Bridge, na aplicação da abordagem Use Case 2.0, avaliando o impacto da aplicação dessa abordagem nas áreas de análise, desenvolvimento e teste das equipes em que a abordagem foi aplicada. Este impacto foi destrinchado em medida de esforço despendido pelas áreas para a aplicação da abordagem e custo-benefício da aplicação da abordagem. Desta forma, o objetivo é contribuir para a

aplicação da abordagem Use Case 2.0 em outras organizações com contexto similares ao projeto SISMOB do laboratório Bridge.

O resultado dos esforços despendido pelas áreas e a avaliação apontam que é fácil de se aplicar a abordagem Use Case 2.0 em uma equipe com o contexto similar ao das equipes em que o estudo foi realizado. A divisão dos casos de uso por *slices* é o grande diferencial da abordagem, pois auxilia na organização da equipe explicitando o que será entregue incremento, o que proporciona uma visão clara do andamento do projeto. Contudo, observa-se também que a aplicação da abordagem demanda um certo esforço, principalmente da área de análise de software, pois a mesma é responsável pela produção da maioria dos produtos de trabalho necessário para a aplicação da abordagem.

Referências

ANTON, A. I. et al. **Deriving goals from a use-case based requirements specification**. Requirements Engineering, v. 6, n. 1, p. 63-73, 2001.

BASSI, D. **Desafios de Requisitos em Métodos Ágeis: Uma Revisão Sistemática.** Master's Dissertation. USP, 2008.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 26 de ago. 2017.

BECK, Kent. **Embracing change with extreme programming.** Computer, v. 32, n. 10, p. 70-77, 1999.

BITTNER, Kurt. **Use case modeling**. Addison-Wesley Longman Publishing Co., Inc., 2002.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Elsevier Brasil, 2006.

BOURQUE, Pierre et al. **Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0**. IEEE Computer Society Press, 2014.

BRIDGE, Laboratório. Institucional. Disponível em : <https://bridge.ufsc.br/>. Acesso em: 27 de nov. de 2017.

CIRILLO, F. **The pomodoro technique**. Lulu. com, 2009.

COHN, M. **Agile estimating and planning**. Pearson Education, 2005.

DE MEDEIROS, Juliana Dantas Ribeiro Viana et al. **Engenharia de requisitos em projetos ágeis; uma revisão sistemática da literatura.** Revista Principia, v. 1, n. 28, p. 11-24, 2015.

HASTIE, Shane; WOJEWODA, Stéphane. **Standish Group 2015 Chaos Report-Q&A with Jennifer Lynch.** Retrieved, v. 1, n. 15, p. 2016, 2015.

GRUBER, J. **Markdown.** Disponível em: <https://daringfireball.net/projects/markdown/>. Acesso em: 18 de mai. 2018.

HAUCK, J. **Use Case 2.0 - Incorporando Práticas Ágeis aos Casos de Uso. In: The Developers Conference.** 12 de mai. de 2016. Disponível em <https://pt.slideshare.net/JeanHauck/usecase-20-jean-hauck>. Acesso em: 20 de out. 2017

ISO/IEC/IEEE. **IEEE. 29148: 2011-Systems and software engineering-Requirements engineering**. Technical report, 2011.

JACOBSON, I.; SPENCE, I.; BITTNER, K.. **Use Case 2.0: The guide to succeeding with use cases.** Ivar Jacobson International, 2011.

JACOBSON, I.; SPENCE, I.; BITTNER, K.. **USE-CASE 2.0 - The Hub of Software Development**. ACM Queue, January-February, 2016

JAQUEIRA, A. et al. **Desafios de Requisitos em Métodos Ágeis: Uma Revisão Sistemática.** 3rd WBMA. São Paulo, 2012.

KAMEI, F. K. **Benefícios e Limitações das Metodologias Ágeis no Desenvolvimento de Software.** Master's Dissertation. UFPE, 2012.

KITCHENHAM, B.A. **Procedures for Performing Systematic Reviews**. **Tech. Report TR/SE- 0401**, Keele University. Inglaterra. 2004.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements engineering: processes and techniques**. Wiley Publishing, 1998.

KOZIOLEK, H. **Goal, question, metric. In Dependability metrics**. Springer Berlin Heidelberg. 2008. pp. 39-42.

LIKERT, R. **A technique for the measurement of attitudes**. Archives of Psychology, v. 22, n. 140, p. 55, 1932

LINGS, B.; LUNDELL, B. **On the adaptation of Grounded Theory procedures: insights from the evolution of the 2G method.** Information Technology & People, Vol. 18, n.3, 2005, p.196-211.

LOCONSOLE, A.; BORSTLER, J. **An industrial case study on requirements volatility measures.** In: Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific. IEEE, 2005. p. 8 pp.

OLIVEIRA, R. R. **A técnica de priorização MoSCoW.** Disponível em: <http://www.ronielton.eti.br/publicacoes/artigoprince2moscow2014_mpbr.pdf>. Acesso em: 18 de fev de 2018.

OMG. **Object Management Group. OMG Unified Modeling Language TM (OMG UML),** Version 2.5. Technical report formal/2015-03-01, 2015. Disponível em: <http://www.omg.org/spec/UML/2.5/>. Acesso em: 20 de out de 2017.

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional.** 7ª Edição. **Ed: McGraw Hill**, 2011.

RUBIN, K. S. **Essential Scrum: A practical guide to the most popular Agile process**. Addison-Wesley, 2012.

RUNESON, P.; HÖST, M.. **Guidelines for conducting and reporting case study research in software engineering**. Empirical software engineering, v. 14, n. 2, p. 131, 2009.

SANTOS, Nuno et al. **Using Scrum together with UML models: A collaborative University-Industry R&D software project.** In: International Conference on Computational Science and Its Applications. Springer International Publishing, 2016. p. 480-495.

SCHWABER, Ken; BEEDLE, Mike. **Agile software development with Scrum**. Upper Saddle River: Prentice Hall, 2002.

SCHWABER, Ken; SUTHERLAND, Jeff. **The scrum guide.** Scrum Alliance, v. 21, 2011.

SOMMERVILLE, I. **Engenharia de Software.** 9. ed. São Paulo: Pearson Addison Wesley, 2011.

VERSIONONE. **The 10th Annual State of Agile Report. 2015**. Disponível em <https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf>. Acesso em: 26 de ago. de 2017.

WAZLAWICK, Raul S. **Análise e projeto de sistemas de informação orientados a objetos.** 2. ed. Rio de Janeiro: Elsevier, 2011.

WAZLAWICK, Raul S. **Engenharia de software: conceitos e práticas**. Rio de Janeiro: Elsevier, 2013.

YIN, Robert K. **Estudo de Caso : Planejamento e Métodos**. Bookman editora, 2011.