UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE CRATEÚS

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

IGOR CLAUDINO DE FRANÇA COSTA

ALCANÇANDO MATURIDADE ÁGIL EM UM AMBIENTE ACADÊMICO DE DESENVOLVIMENTO DE SOFTWARE

CRATEÚS

2019

IGOR CLAUDINO DE FRANÇA COSTA

ALCANÇANDO MATURIDADE ÁGIL EM UM AMBIENTE ACADÊMICO DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará
Campus de Crateús, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Msc. André Meireles de Andrade

CRATEÚS

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C872a Costa, Igor Claudino de França.
Alcançando maturidade ágil em um ambiente acadêmico de desenvolvimento de software / Igor Claudino de França Costa. – 2019.
59 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Sistemas de Informação, Crateús, 2019.
Orientação: Prof. André Meireles de Andrade.

1. Metodologia Ágil. 2. Maturidade Ágil. 3. Processo de Software. I. Título.

CDD 005

IGOR CLAUDINO DE FRANÇA COSTA

ALCANÇANDO MATURIDADE ÁGIL EM UM AMBIENTE ACADÊMICO DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará
Campus de Crateús, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em: 05 de Julho de 2019

BANCA EXAMINADORA

Prof. Msc. André Meireles de Andrade (Orientador)
Universidade Federal do Ceará - UFC

Prof. Msc. Allysson Allex de Paula Araújo
Universidade Federal do Ceará - UFC

Prof. Dr. Rodrigo de Almeida Vilar de Miranda
Universidade Federal da Paraíba - UFPB

À minha mãe, por sempre acreditar e me apoiar nas escolhas. Aos meus amigos que estão juntos comigo nos momentos bons e ruins desta caminhada.

AGRADECIMENTOS

À todos os envolvidos, destacando minha mãe, Claudina por ter me dado a educação necessária para que meu caráter fosse formado corretamente e aos meus amigos que nunca me deixaram fraquejar.

Ao Prof. Me. André Meireles de Andrade por me orientar neste trabalho científico na área que aprendi a gostar na graduação e que foi incentivada por ele por meio de suas experiências.

A Prof. Me. Lisieux Marie Marinho dos Santos Andrade por ministrar tão bem a disciplina de Projeto de Pesquisa Científica e Tecnológica, demonstrando total zelo e atenção nas sugestões de escrita deste trabalho.

A minha grande amiga da vida, Luiza Ananda por estar comigo nos momentos bons e ruins, ajudando a passar por eles com a melhor companhia.

A minha grande parceira de todos os momentos, Agatha Martins por estar ao meu lado nas glórias e dificuldades da vida e por me ajudar na minha caminhada até este momento.

A todos os meus amigos de estágio por compartilhar, junto comigo, desta luta de conclusão de graduação.

“Cante alto seus sonhos.”

(Gustavo Bertoni)

RESUMO

O crescimento da necessidade de profissionais bem instruídos em desenvolvimento de *software* é notório no mercado da Tecnologia da Informação na atualidade. Assim, é indispensável que as Instituições de Ensino Superior que dispõem de cursos de graduação na área de Tecnologia da Informação forneçam uma formação que instrua corretamente seus graduandos por meio de uma experiência mais próxima do mercado de trabalho. Muitas das Instituições utilizam de ambientes que simulam empresas da indústria de *software* como forma de preparar os alunos com as mais avançadas metodologias de desenvolvimento de sistemas. As metodologias ágeis estão em grande foco na atualidade e tem como objetivo principal a entrega rápida de software funcionando e maior adaptação à mudanças. Porém, não é fácil a aplicação de tais metodologias nas organizações e principalmente em ambientes de simulação criados dentro das Instituições de Ensino Superior, pois seus alunos ainda não dispõe de uma maturidade avançada. Assim, faz-se necessário que um processo ágil seja definido, executado e avaliado para que o nível de maturidade ágil do ambiente seja satisfatório para o mercado. Este trabalho apresenta um estudo de caso para definição, aplicação, execução e medição de um processo ágil em um ambiente de simulação de mercado de software em uma Instituição de Ensino Superior para qualificar de maneira mais enfática o aluno de graduação em cursos de Tecnologia da Informação.

Palavras-chave: Metodologia ágil. Maturidade ágil. Desenvolvimento de *software*. Ambiente de simulação de mercado de *software*.

ABSTRACT

The growing need for well-educated software development professionals is notorious in today's information technology market. Therefore, it is indipensable that the institutions of higher education that have undergraduate courses in the area of Information Technology provide a training that instructs their graduates correctly through an experience closer to the job market. Many of the institutions use environments that simulate companies in the software industry as a way to prepare students with the most advanced systems development methodologies. The agile methodologies are in great focus at the present time and its main objective is the fast delivery of working software and greater adaptation to changes. However, it is not easy to apply such methodologies in organizations and especially in simulation environments created within the Institutions of Higher Education, because their students still do not have an advanced maturity. Therefore, it is necessary that an agile process is defined, executed and evaluated so that the level of agile maturity of the environment is satisfactory for the market. This paper presents a case study for the definition, application, execution and measurement of an agile process in a software market simulation environment in a Higher Education Institution to more emphatically qualify the undergraduate student in Information Technology courses.

Keywords: Agile Methodology. Agile maturity. Development of software. Software market simulation environment.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

No decorrer dos últimos anos, com o aumento da demanda de profissionais de Tecnologia da Informação (TI) em desenvolvimento de *softwares* de qualidade, as organizações estão cada vez mais rigorosas na análise do profissional a ser contratado. Dessa forma, é indispensável uma melhor formação prática dos egressos dos cursos da área de tecnologia pelas universidades. Porém, tal formação é um desafio cada vez maior para as Instituições de Ensino Superior (IES). Essa prática "diz respeito a municiar, motivar e qualificar o futuro profissional com as boas práticas de análise e desenvolvimento de *software*"(BEGOSSO *et al.*, 2011). Com este intuito, professores e demais acadêmicos procuram desenvolver técnicas de preparação teórica/prática dos alunos ainda em processo de graduação para que os mesmos saiam com experiências de ambientes reais para o mercado de trabalho. Na Faculdade de Tecnologia Bandeirantes, por exemplo, criou-se a disciplina obrigatória de Escritório de Projetos (BARTH *et al.*, 2012) que engloba a "formação de habilidades pessoais (i.e., capacidade de trabalhar em grupo, gerenciar suas próprias tarefas, negociar) e habilidades organizacionais (i.e., orientar negócios, pensar em projetos, analisar viabilidades, desenvolver ferramentas)".

Dentre as principais práticas de desenvolvimento de *software* existentes, as práticas ágeis ganham força nos últimos tempos, tais práticas surgem a partir do Manifesto de Desenvolvimento de *Software* Ágil (BECK *et al.*, 2001). Este manifesto foi elaborado por pesquisadores de desenvolvimento de *software* para padronizar e definir os princípios de desenvolvimento ágil. Ele tem como objetivos principais: indivíduos e interações mais que processos e ferramentas, *software* em funcionamento mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos e responder a mudanças mais que seguir um plano (BECK *et al.*, 2001). Rao *et al.* (2011) afirmam que os princípios ágeis de desenvolvimento de *software* "surgiram dos princípios tradicionais de desenvolvimento de *software* e de várias experiências baseadas nos sucessos e fracassos dos projetos de *software*". A aplicação desses princípios tem aumentado a produtividade no desenvolvimento de sistemas de informação, no entanto, aplicá-los na prática não é uma tarefa fácil. Segundo Qumer *et al.* (2007), um dos problemas identificados na aplicação de métodos ágeis é a incapacidade das organizações de construir, executar e gerenciar processos ágeis de desenvolvimento de *software*. Consequentemente, os ambientes criados pelas IESs para simular os ambientes de projeto de *software* das organizações

que atuam no mercado sofrem da mesma dificuldade.

Fundamentado nisso, diversas pesquisas são desenvolvidas para elaboração de técnicas que facilitam a aplicação de métodos ágeis nas organizações de desenvolvimento de *software*. Dentre elas, destacam-se: o *tayloring*, na qual se adapta o processo não ágil já existente na organização para um método ágil; e a adoção de metodologia ágil, na qual se constroi o processo de desenvolvimento desde o começo de acordo com a necessidade da organização. Qumer e Henderson-Sellers (2008) apresentam o *Agile Software Solution Framework* (ASSF), uma estrutura completa para auxiliar os gerentes na avaliação do grau de agilidade de que precisam e como identificar maneiras apropriadas de introduzir essa agilidade em sua organização, ilustrada com alguns estudos de caso da indústria. Nos dois estudos de caso apresentados, eles obtiveram resultados satisfatórios em relação a introdução de práticas ágeis na indústria.

Dessa forma, o presente trabalho visa apresentar a definição e implantação de um processo ágil no contexto do Núcleo de Práticas em Desenvolvimento de *Software* [1] (NPDS) da Universidade Federal do Ceará (UFC) - *Campus* Crateús, um ambiente voltado para a iniciação profissional dos alunos dos cursos de Ciência da Computação e Sistemas de Informação, através da vivência em projetos de *software* e da aplicação de boas práticas da engenharia de *software*.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem com objetivo principal estabelecer e implantar um processo de desenvolvimento ágil de *software* para o Núcleo de Práticas de Desenvolvimento de Sistemas.

1.2.2 Objetivos Específicos

Para melhor compreender como os resultados alcançados atingem o objetivo geral do trabalho, definiu-se os seguintes objetivos específicos:

- Definir um processo de desenvolvimento para o NPDS;
- Garantir um nível de maturidade ágil alto do processo definido;
- Implantar o processo de desenvolvimento definido em projetos do Núcleo;

[1] http://npds.crateus.ufc.br

2 FUNDAMENTAÇÃO TEÓRICA

No cenário atual torna-se indispensável o uso de *softwares* para automatizar serviços e processos que até então se davam de forma manual. Assim, é notório que o papel da Engenharia de *Software* é essencial no mundo moderno visto a complexidade do entendimento na construção de um *software*. Sommerville (2003) define Engenharia de *Software* como

> uma disciplina de engenharia cujo foco está em todos os aspectos da produção de *software*, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado.

Diante dessa notoriedade, é perceptível que exista uma dificuldade no controle da capacidade de entendimento dos sistemas desenvolvidos. Sommerville (2003) afirma que pelo fato de sistemas de *software* serem intangíveis e abstratos, sua complexidade aumenta rapidamente, tornando-os difíceis de compreender e com alto custo para se efetuar uma mudança. Portanto, faz-se necessário a elaboração de metodologias para implementação de sistemas de *software* no qual obtenha-se um controle no processo de desenvolvimento em prol de se entregar um produto final de qualidade. Mas para a criação de tais metodologias, é importante entender como se dá o processo de desenvolvimento de *software*.

2.1 Engenharia de *Software* e seus processos de desenvolvimento

Segundo Pressman (2011), processo é "um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho (*work product*)". No contexto do desenvolvimento de sistemas, este processo se dá por um grupo de atividades que se relacionam para a entrega de um produto de *software*. Tal processo surge a partir da necessidade de um padrão adaptável para o surgimento de novos modelos. Para Sommerville (2003), processos de *software* devem incluir quatro atividades principais: especificação, projeto e implementação, validação e evolução de *software*. Analisando o contexto histórico de desenvolvimento de *software*, nota-se a presença dessas atividades em modelos desenvolvidos durante o decorrer dos anos. Os processos dividem-se em dois principais tipos: Processos dirigidos a planos e Processos Ágeis, que serão abordados a seguir.

2.1.1 Processos dirigidos a planos

No princípio, os processos dirigidos a planos foram elaborados com o intuito de "trazer ordem ao caos existente na área de desenvolvimento de *software*"(PRESSMAN, 2011).

Estes processos tem como característica fundamental a identificação de "estágios distintos do processo de *software* com saídas associadas a cada estágio"(SOMMERVILLE, 2003)

Dentre os muitos processos dirigidos a planos, um exemplo explícito para se destacar é o modelo em cascata (*Waterfall Model*), elaborado por Royce (1987), que será descrito a seguir.

2.1.1.1 Modelo em cascata (Waterfall Model)

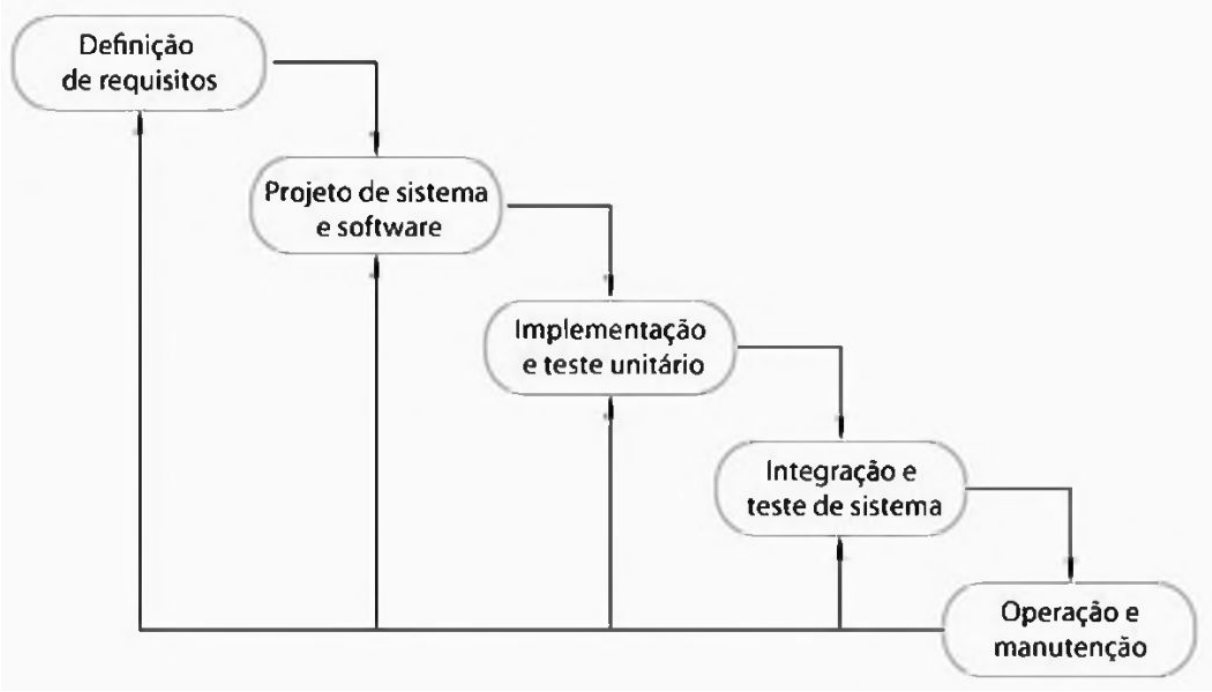
O modelo em cascata é um exemplo de um processo dirigido a planos. A característica mais destacável do modelo em cascata é a necessidade do cumprimento do passo anterior para o avanço de fase, como destaca Sommerville (2003):"Em princípio, você deve planejar e programar todas as atividades do processo antes de começar a trabalhar nelas."

O modelo cascata distribui as atividades definidas por Sommerville (2003) em estágios. A Especificação de *software* é mapeada para o estágio de Análise e Definição dos requisitos, onde deve-se extrair dos possíveis usuários as funcionalidades do sistema a ser desenvolvido. O Projeto e Implementação de *software*, é mapeado para dois estágios: Projeto de Sistema e *Software*, no qual os requisitos gerados no estágio anterior dão suporte a definição da arquitetura do projeto, determinando suas abstrações e relacionamentos, e Implementação e Teste Unitário, fase em que é iniciada a construção do sistema. A Validação foi mapeada para a fase de Integração e Teste de Sistema, na qual integra-se as partes do sistema e valida-se o seu funcionamento geral. Finalmente, para a Evolução tem-se o estágio de Operação e Manutenção, em que o sistema é posto em produção e os erros identificados são corrigidos no decorrer da fase. A Figura 1 denota as atividades principais do desenvolvimento em cascata, explicitando que, para passar para a próxima fase, faz-se necessário o cumprimento da fase anterior.

Cada estágio tem como saída um documento aprovado para que se possa avançar no processo. Royce (1987) deixa bem claro que este modelo de processo foi definido a partir de experiências para aplicação em sistemas de grande porte, pois eles exigem que os requisitos definidos sejam maximamente compreendidos. Sommerville (2003) alega que o maior problema deste modelo "é a divisão inflexível do projeto em estágios distintos", o que dificulta o atendimento a mudanças solicitadas pelo cliente.

Em um cenário atual onde as mudanças são recorrentes e inevitáveis, faz-se necessária a elaboração de modelos que atendam projetos de pequeno e médio porte e ainda que atendam facilmente as alterações solicitadas pelo usuário. Para isso, surgem os métodos ágeis, baseados no conceito de entrega incremental, no qual os "incrementos desenvolvidos são entregues ao

Figura 1 – Fases do modelo em cascata



Fonte: (SOMMERVILLE, 2003)

cliente e implantados para uso em um ambiente operacional"(SOMMERVILLE, 2003). Os Métodos Ágeis possuem grande influência no mercado atual e que serão abordados nas duas seções a seguir.

2.2 Metodologias Ágeis

Para lidar com os obstáculos associados à dificuldade de atender a mudanças, atrasos na entrega de sistemas funcionando e o alto custo de desenvolvimento, um grupo inicial de dezessete pesquisadores formaram a *Agile Software Development Alliance* e definiram o Manifesto Ágil, citando quais os princípios devem ser seguidos para um melhor desenvolvimento de *software* (BECK *et al.*, 2001). O Manifesto Ágil estabelece os fundamentos que devem ser seguidos por modelos ágeis ao definir suas práticas.

2.2.1 Manifesto Ágil

O Manifesto Ágil é estabelecido com base em quatro valores que devem ser encarados como preferências. São eles: indivíduos e interações mais que processos e ferramentas; *software* em funcionamento mais que documentação abrangente; colaboração com o cliente mais que negociação de contratos; e responder a mudanças mais que seguir um plano (BECK *et al.*, 2001).

Ao considerar indivíduos e interações mais que processos e ferramentas, os fatores mais importantes a serem considerados são as pessoas e como elas funcionam trabalhando em conjunto.

Já no *software* em funcionamento mais que documentação abrangente, nota-se que o cliente prefere o sistema em funcionamento muito mais do que um manual que descreva o que o sistema irá fazer. Logo, torna-se prioritário o desenvolvimento da aplicação ao esforço da construção de uma documentação.

Em colaboração com o cliente mais que negociação de contratos, é indubitável que apenas o cliente sabe o que quer. Dessa maneira, é indispensável uma comunicação mais frequente com o mesmo. Além disso, ele estará participando ativamente do desenvolvimento e qualidade do produto final.

Por fim, responder a mudanças mais que seguir um plano é importante pelo fato de que as prioridades dos clientes mudam constantemente. Por isso, estar sujeito a mudanças durante o processo é primordial para satisfazer as necessidades que surgem durante o projeto.

Além dos valores que devem servir de guia para modelos ágeis, o manifesto define doze princípios ágeis (BECK *et al.*, 2001):

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de *software* com valor agregado;
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente;
3. Entregar frequentemente *software* funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo;
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho;
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face;
7. *Software* funcionando é a medida primária de progresso;
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade;

10. Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial;
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis;
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Embasado no que o manifesto descreve, modelos de processos ágeis de *software* foram definidos ao decorrer dos anos. Dentre modelos definidos, iremos abordar o modelo SCRUM (SCHWABER, 1997), um dos processos já consolidados nas implementações atuais.

2.2.2 SCRUM

O termo SCRUM vem de uma forte formação de atacantes que se unem em posições específicas no esporte *rugby*.

O modelo de processo SCRUM pode ser definido como uma abordagem que trata grandes partes do desenvolvimento de sistemas como uma caixa preta controlada (SCHWABER, 1997). O autor utiliza de conceitos de controle de processos na indústria para aplicação no desenvolvimento de sistemas, baseando-se no nível de controle do processo, podendo ser teórico (totalmente definido) ou empírico (caixa preta). Tal modelo é um aprimoramento da abordagem iterativa e incremental, o que é contextualizado pelo terceiro princípio definido no Manifesto Ágil. Assim, mostra-se a adaptação a possíveis mudanças durante a execução do projeto, a evolução que favorece aqueles que operam com exposição máxima à mudança ambiental e a otimização da adaptação flexível à mudança (SCHWABER, 1997).

Schwaber (1997) dividiu o modelo de processo em cerimônias e atividades. Relacionando com as atividades citadas por Sommerville (2003) o SCRUM apresenta, para a Especificação de *software*, o *Planning* (Planejamento), na qual ocorre, dentre seus procedimentos, a elaboração de uma lista abrangente de *backlog*[1], a definição da(s) equipe(s) do projeto para a construção do produto e a avaliação e controle de riscos.

Para a atividade de Projeto e Implementação, o SCRUM define o *Design* de Alto Nível e o Desenvolvimento (*Sprint*)[2] que consistem em um ciclo iterativo de trabalho de desenvolvimento. Esse ciclo se baseia nos procedimentos de desenvolver, empacotar, revisar e ajustar (SCHWABER, 1997).

[1] Requisitos de funcionalidade do produto. *Bugs*, defeitos, aprimoramentos solicitados pelos clientes, funcionalidade competitiva do produto, funcionalidade de vantagem competitiva e atualizações tecnológicas são itens de *backlog* (SCHWABER, 1997).

[2] Uma *Sprint* é um conjunto de atividades de desenvolvimento conduzidas durante um período pré-definido, geralmente de uma a quatro semanas. (SCHWABER, 1997).

As duas últimas atividades (Validação e Evolução) se unem na cerimônia de *Closure* (Encerramento), quando o produto desenvolvido é preparado para lançamento. Integração, testes do sistema, documentação do usuário, preparação do material de treinamento e preparação do material de *marketing* estão entre as tarefas de encerramento.

2.2.3 Personalização e adoção de métodos ágeis

Com o avanço da tecnologia e a implementação de novas metodologias no desenvolvimento ágil para maior produtividade e qualidade, as organizações de desenvolvimento de *software* estão empenhadas na adaptação de seus processos a tais metodologias. Porém, não é simples implantá-las em um ambiente onde já se encontram enraizadas as práticas de modelos dirigidos a planos, como Qumer *et al.* (2007) mostram. Dois problemas fundamentais encontrados por eles são: em primeiro lugar, as organizações não conseguem construir, executar e gerenciar processos ágeis de desenvolvimento de *software* e, em segundo lugar, a falta de um modelo para guiar a adoção e a melhoria ágil. Qumer e Henderson-Sellers (2008), ao utilizarem do *framework* desenvolvido por eles para adoção de métodos ágeis, explicam em um de seus casos na indústria que a transição de uma abordagem e mentalidade tradicionais leva tempo e é de suma importância a ajuda do gerente de produto junto aos desenvolvedores para fazer uma transição bem-sucedida.

Dessa forma, observa-se que é necessário moldar os futuros profissionais de Tecnologia da Informação que ainda estão em preparo nas Instituições de Ensino Superior nas técnicas de Engenharia de *Software* exigidas pelo mercado, oferecendo um ambiente de formação onde a cultura ágil funcione de forma madura e consistente, tendo em vista que alcançar maturidade ágil ainda é um desafio.

2.3 Formação acadêmica teórico/prática em desenvolvimento profissional de *software*

Nas atuais IES, em geral, que disponibilizam formação em cursos de TI, a carga-horária teórica é bem contemplada. Inúmeras disciplinas formam o aluno com cenários e ensinamentos para a aplicação futura no mercado de trabalho. Porém, torna-se difícil a obtenção da noção real do ambiente de desenvolvimento de software sem uma formação prática. Infelizmente, as IES ainda não equilibram de maneira satisfatória os ensinamentos práticos e teóricos. Peckham e Batson (2004) afirma que "a universidade encontra dificuldades para fornecer uma

experiencia balanceada entre a teoria aprendida e a prática existente no mercado". Dessa forma, os formandos em cursos de TI saem de suas graduações sem a maturidade mínima exigida pelo mercado (BEGOSSO *et al.*, 2011).

Muitos docentes buscam formas de suprir essa necessidade, na qual destaca-se, por exemplo, a iniciativa de Escritório de Software (BARTH *et al.*, 2012) da Faculdade Bandeirantes. Tal iniciativa aborda uma disciplina obrigatória no currículo dos discentes que tem como objetivo formar profissionais com a qualidade esperada e na velocidade demandada pelo mercado e não só simplesmente ensinar conhecimento técnico.

Outra tentativa que pode-se citar é o Núcleo de Práticas em Informática (GONÇALVES *et al.*, 2013). Tal núcleo surgiu com o objetivo de suprir as necessidades de sistemas para uso interno do *campus* da Universidade Federal do Ceará em Quixadá. Mas com o passar do tempo, atividades como o provimento de estágio para estudantes dos cursos de graduação foram adicionadas aos objetivos do núcleo.

Com o mesmo objetivo, o *Campus* da Universidade Federal do Ceará em Crateús idealizou a criação de um setor chamado Núcleo de Prática em Desenvolvimento de Sistemas - NPDS -, com o objetivo de propiciar experiências práticas aos alunos que estão na graduação dos cursos de Sistemas de Informação e Ciência da Computação, núcleo este que será abordado como ambiente para adoção de métodos ágeis deste trabalho.

3 METODOLOGIA

Este capítulo apresenta a metodologia utilizada neste trabalho. As seções a seguir abordarão sobre o Núcleo de Práticas em Desenvolvimento de Sistemas e o método de pesquisa *Design Science*.

3.1 Núcleo de Práticas em Desenvolvimento de Sistemas (NPDS)

O Núcleo de Práticas em Desenvolvimento de Sistemas é um setor do *campus* da UFC em Crateús criado com o objetivo de fornecer aos estudantes dos cursos de tecnologia da informação e comunicação um ambiente para realização de projetos de *software* e *hardware* que possam prepará-los para o mercado de trabalho. A equipe é formada por professores e funcionários da UFC responsáveis por garantir a qualidade do trabalho aprendido e executado pelos alunos. Oferece-se também, todos os semestres, vagas para estágio, onde os alunos podem estagiar e cumprir o estágio curricular obrigatório dos cursos de Ciência da Computação e Sistemas de Informação.

No início da pesquisa, o NPDS dispunha de quatro projetos em andamento, contando com onze alunos distribuídos nestes projetos. Para gerenciar os projetos e dar suporte técnico aos alunos, o Núcleo conta com 3 professores e um analista de sistemas.

O processo neste núcleo não disponibilizava de práticas definidas, o que dificultava o entendimento por parte dos que estavam inseridos. O que era executado partia das indicações do gerente de projetos.

Muito mais que criar oportunidades para uma experiência prática, é de suma importância que tal experiência seja a mais próxima possível da exigência do mercado. Até o início deste trabalho, o NPDS não dispunha de um processo de desenvolvimento de *software* bem definido, sendo este o cenário inicial da pesquisa.

3.2 *Design Science* e sua aplicação ao trabalho

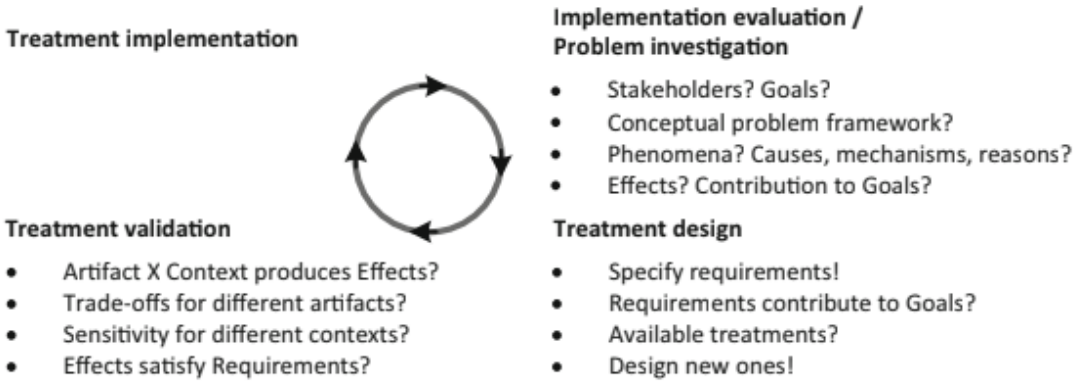
Para desenvolver a pesquisa, escolheu-se a metodologia *Design Science* (WIERINGA, 2014), inicialmente, por trata-se de uma metodologia moderna e amplamente utilizada em estudos na área da Engenharia de *Software*. Essa metodologia utiliza-se da investigação de artefatos em um contexto, estes artefatos estudados "são projetados para interagir com um contexto de problema, a fim de melhorar algo nesse contexto"(WIERINGA, 2014). O *Design*

Science define o ciclo da engenharia que consiste em :

- Investigação do problema;
- Construção da Solução;
- Validação da Solução;
- Implementação da Solução;
- Avaliação da Implementação.

O ciclo pode ser observado na Figura 2.

Figura 2 – Ciclo da engenharia



Fonte: (WIERINGA, 2014)

Como elaboração da pesquisa, faz-se necessária a definição do artefato a ser investigado e do contexto. Identificamos no nosso trabalho o artefato como sendo o processo ágil de desenvolvimento de *software* e o contexto como sendo o NPDS. Assim, elabora-se o que o autor chama de *Design Problems*, que especifica objetivos para se construir uma solução para a resolução do problema. Juntamente, elabora-se as *Knowledge Questions* tal que as respostas a essas perguntas irá ajudar a verificar se o objetivo foi atingido. Dessa forma o Quadro 1 expõe os *Design Problems* e *Knowledge Questions* definidas para este trabalho.

Quadro 1 – *Design Problems* e *Knowledge Questions*

Design Problems	Knowledge Questions
Definir um processo de desenvolvimento ágil baseado nos itens verificados pelo AMM	O NPDS tem um processo de software definido?
Garantir que o processo de software definido tenha o nível II de maturidade ágil do AMM	O processo de desenvolvimento e software do NPDS é ágil? Quanto?
Implantar o processo de desenvolvimento elaborado através da execução das práticas definidas a fim de melhorar a maturidade ágil do NPDS	O NPDS executa o processo de desenvolvimento definido?

Fonte: Próprio autor.

Baseando-se no contexto do NPDS e como ele se comporta atualmente, faz-se necessário a adoção de um processo de desenvolvimento ágil para adequar tal núcleo aos padrões de mercado. Além disso, é indispensável que, após a implantação do processo, o mesmo seja realmente executado e ainda mensurado para se obter o controle de maturidade. Assim, o presente trabalho propõe as seguintes etapas, baseadas no *Design Science*, que serão expostas a seguir:investigação do problema, construção da solução, validação da solução e sua implementação/implantação e avaliação da implementação.

3.2.1 Investigação do problema

Wieringa (2014) cita algumas técnicas para investigação do problema, dentre as quais foi escolhido para este trabalho *Observational Case Studies* pelo fato de que "o pesquisador pode estudar uma amostra de casos, mas o objetivo da pesquisa de estudo de caso não é adquirir conhecimento sobre amostras, mas sobre casos individuais."(WIERINGA, 2014). Assim, será elaborado um questionário baseado no nível II do AMM (*Agile Maturiry Model*) (PATEL; RAMACHANDRAN, 2009), um modelo de maturidade para análise de aplicação de métodos ágeis em organizações, e enviado aos membros do NPDS para que seja feito um diagnóstico para identificar possíveis deficiências no processo atual de software. Com estes resultados, pode-se dar início a construção da solução.

3.2.1.1 Diagnóstico obtido na Investigação

O diagnóstico do NPDS antes da implementação do processo será realizado através da análise das respostas do questionário do AMM nível II aplicado aos membros do NPDS. A análise indicará qual o grau de aderência do NPDS em relação cada um dos pontos de agilidade propostos pelo *framework* AMM, ou seja, em relação a cada uma das afirmativas do questionário AMM. Os níveis de aderência definidos servirão de base para elaboração do processo de desenvolvimento, uma vez que indicarão diretamente quais as fragilidades do NPDS em relação à agilidade do processo executado antes da implementação da solução.

3.2.2 Construção da Solução

A definição do processo ágil de *software* parte do princípio de adoção de métodos ágeis. Tal processo será construído baseado no diagnóstico obtido através dos dados obtidos

do processo descrito na Investigação do Problema. A construção da solução será abordada nas seções que seguem.

3.2.2.1 Utilização do framework para adoção de métodos ágeis

Com a finalidade de prover uma base teórica mais formal para a construção do processo, utilizou-se o *framework* de adoção ASSF proposto por Qumer e Henderson-Sellers (2008). Tal *framework* define um modelo de aspecto conceitual ágil e como este modelo influencia o contexto da organização, atrelando quais práticas ágeis adequadas devem ser levadas em consideração para cada item do modelo. O modelo leva em consideração alguns pontos importantes na organização. Considera-se conhecimento sobre o contexto do desenvolvimento, criando-o e compartilhando-o (tácito em explícito). Também, considera-se a governança em TI, que visa identificar e implementar as iniciativas para determinar sistematicamente os envolvidos em equipes capacitadas (equipes ágeis) que têm o direito de tomar, dar entrada e responsabilizar os envolvidos pelas decisões. O modelo propõe um núcleo do método que tem 6 componentes importantes para o processo:

- Agilidade;
- Pessoas;
- Processo;
- Produto;
- Ferramentas;
- Abstração.

Os seis componentes são usados e combinados para construir um processo de desenvolvimento de software ágil usando uma abordagem de engenharia de método situacional para um contexto organizacional específico. O processo ágil de desenvolvimento de software criado com a ajuda do ASSF conterá aspectos de governança e conhecimento que anteriormente não eram o foco dos métodos ágeis tradicionais.

Também, o *framework* dispõe de seis estágios derivados do *Agile Adoption and Improvement Model* (AAIM) (GILL *et al.*, 2007) e suas sugestões de práticas. Os estágio são:

- AAIML 6: Produção enxuta, manter-se ágil:
  - Produção de qualidade;
  - Recursos mínimos possíveis;
  - Manutenção do processo ágil.

- AAIML 5: Aprendizagem
  - Pesquisa e aprendizado de lições;
  - Gerenciamento de aprendizado;
  - Aplicação das lições aprendidas.
- AAIML 4: Orientado para as Pessoas
  - Valorizar as pessoas;
  - Não ignore processos e ferramentas;
  - Mantenha intactos os valiosos.
- AAIML 3: Artefatos executáveis
  - Versões de software executáveis;
  - Documentação mínima;
  - Incentivar a Documentação Mínima.
- AAIML 2: Comunicação orientada
  - Comunicação focada;
  - Cooperativa e Colaborativa;
  - Comunicação cara a cara.
- AAIML 1: Velocidade, flexibilidade e capacidade de resposta
  - Iterativo e incremental;
  - Incentivar e acomodar mudanças;
  - Refletir mudanças.

O processo definido para o NPDS teve como objetivo alcançar apenas os 3 primeiros estágios do ASSF devido ao nível de maturidade verificado pela etapa de Investigação do problema.

3.2.3 Validação da Solução

Para assegurar que o processo definido possui agilidade adequada, é necessário medir o grau de maturidade ágil do mesmo. Para isso, será utilizado a ferramenta de análise 4-DAT (GILL; HENDERSON-SELLERS, 2006), sugerida por Qumer e Henderson-Sellers (2008) como componente do ASSF. A ferramenta analisa os processos ágeis a partir de quatro perspectivas ou dimensões: caracterização do escopo do método, caracterização da agilidade, caracterização dos valores ágeis (manifesto ágil) e caracterização do processo de software. Para cada uma das dimensões propostas, o 4-DAT analisa uma série de atributos do processo por meio de questões.

A única dimensão atualmente quantificada do 4-DAT é a caracterização da agilidade, que permite uma avaliação numérica do grau de agilidade tanto em larga escala (nível de processo) quanto em pequena escala (práticas). O cálculo desta dimensão funcionará da seguinte forma: uma tabela é construída a partir do processo definido, no qual valores de célula 0 ou 1 são inseridos para cada fase (para uma avaliação à nível de processo) ou, em um nível mais detalhado, para cada prática individual (para uma avaliação à nível de práticas). Considera-se as cinco características de agilidade: flexibilidade (FY), velocidade (SD), magreza (LS), aprendizagem (LG) e capacidade de resposta (RS). Tais características são analisadas para cada fragmento do processo e então o total geral do processo (e consequentemente o grau médio de agilidade) é gerado. A Figura 3 exemplifica com mais detalhes este procedimento.

Figura 3 – Cálculo da dimensão 2 do 4-DAT

	Agility features					
	FY	SD	LS	LG	RS	Total
(i) Phases						
Phase 1	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
Phase 2	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
Phase 3	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
etc.	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
Total	$(0 - x)$	$(0 - x)$	$(0 - x)$	$(0 - x)$	$(0 - x)$	$0 - 5 * x$
Degree of agility (high level)	$(0 - x)/x$	$(0 - x)/x$	$(0 - x)/x$	$(0 - x)/x$	$(0 - x)/x$	Total divided by number of cells in table
(ii) Practices						
Practice 1	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
Practice 2	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
etc.	0 or 1	0 or 1	0 or 1	0 or 1	0 or 1	0 – 5
Total	$(0 - y)$	$(0 - y)$	$(0 - y)$	$(0 - y)$	$(0 - y)$	$0 - 5 * y$
Degree of agility (low level)	$(0 - y)/y$	$(0 - y)/y$	$(0 - y)/y$	$(0 - y)/$ y	$(0 - y)/y$	Total divided by number of cells in table

Fonte: (QUMER; HENDERSON-SELLERS, 2008)

Qumer e Henderson-Sellers (2008) afirmam que apenas é necessário que o valor obtido seja aproximado de 0,5 a 0,6 para que qualquer método ágil construído ter agilidade média suficiente para se qualificar como um método ágil.

Dessa forma, utilizou-se apenas a dimensão dois devido à inviabilidade diante do tempo disponível para a pesquisa e por esta ser a mais objetiva, trabalhando com análise quantitativa e levando em conta atributos de grande relevância para a determinação da agilidade do processo. Dispondo do processo definido e validado, é necessário a implantação do mesmo nas equipes que compõe o núcleo.

3.2.4 Implementação da solução

Após definição do processo, o mesmo deve ser apresentado aos membros do núcleo. Para esta apresentação, deve-se realizar uma reunião apresentando cada prática definida, em que

fase do processo de desenvolvimento ela impacta e quais suas características ágeis para que, dessa forma, exista entendimento claro de todos os membros para a execução do processo. As práticas estarão dispostas em fragmentos com template fornecido pelo ASSF (QUMER; HENDERSON-SELLERS, 2008) para melhor visualização dos membros. Após isso, os membros de cada equipe devem colocar em prática o processo de desenvolvimento estabelecido.

3.2.5 Avaliação da Implementação

Após a execução da solução, faz-se necessário avaliar se o contexto apresentou mudanças em relação ao que foi investigado antes da aplicação da solução. É importante também verificar tais mudanças em relação às *Knowledge Questions* e aos *Design Problems* estabelecidos inicialmente (Quadro 1).

A partir da análise dos resultados da reaplicação do questionário do AMM Nível II será possível verificar se a implementação da solução apresentou maior maturidade ágil do que antes da aplicação e se o nível II do AMM foi atingido.

4 RESULTADOS

De acordo com o *Design Science* (WIERINGA, 2014), é necessário definir os objetos de estudo que são projetados para interagir com um contexto de problema, a fim de melhorar algo nesse contexto. O trabalho tem como artefato um processo de desenvolvimento ágil de software e o como contexto o Núcleo de Práticas de Desenvolvimento de Sistemas (NPDS). Após isso, o *Design Science* (WIERINGA, 2014) apresenta etapas para estudo dos objetos definidos e elaboração de um projeto para solução. As etapas são: Investigação do problema, Projeto de Solução, Validação da Solução, Implementação da Solução e Avaliação da Implementação, tendo este último o objetivo de verificar se alguma melhoria foi observada em relação às *Knowledge Questions*.

4.1 Investigação do Problema

A investigação do problema neste trabalho foi realizada através de um estudo de caso observacional com o objetivo de diagnosticar e identificar o nível de maturidade ágil e o grau de adesão às práticas ágeis do processo de desenvolvimento de *software* do NPDS antes da aplicação da solução. Então, foi elaborado um questionário de vinte e sete questões retiradas do nível II do AMM (*Agile Maturiry Model*) (PATEL; RAMACHANDRAN, 2009), um modelo de maturidade para análise de aplicação de métodos ágeis em organizações. As afirmativas dispostas no questionário são divididas nas áreas de Planejamento de projetos (Planejamento de *Release*), Desenvolvimento dirigido a *Story Cards* (Engenharia de Requisitos), Disponibilidade do cliente no local, Introdução ao TDD (*Test Driven Development*). O nível II foi escolhido pois trata-se do nível mais baixo do modelo. As aplicações de níveis mais altos só são aplicados quando a organização apresenta aderência total aos níveis mais baixos. Na aplicação do questionário, para cada afirmativa, utilizando-se uma Escala de Likert (LIKERT, 1974) como opções de resposta, o membro do NPDS poderia responder: "Concordo totalmente", "Concordo parcialmente", "Indiferente ou prefiro não opinar", "Discordo parcialmente"ou "Discordo totalmente". Foram entrevistados todos os estagiários do NPDS no período 2018.2, num total de 15 alunos, 2 do sexo feminino e 13 do sexo masculino, que estavam divididos em 4 projetos, e todos estavam entre o sétimo e oitavo semestre de suas graduações: 8 de Ciência da Computação e 7 de Sistemas de Informação.

O foco do estudo estava na avaliação da maturidade ágil do processo de desenvolvi-

mento aplicado aos projetos que estavam sendo executados no núcleo. Podemos ver a seguir as afirmativas do questionário do aplicado está disposto a seguir:

- Área: Planejamento de projetos (Planejamento de *Release*)
    1. A técnica de planejamento (Scrum *Planning*) é usada para criar planos de projeto
    2. Estima-se o escopo do projeto
    3. Planejamento de Release é usado para criar cronogramas
    4. A velocidade do projeto é medida
    5. O projeto é dividido em iterações
    6. Estimação é feita pelos desenvolvedores
    7. Estimativas são feitas baseadas em estimativas passadas
    8. O planejamento é baseado no valor de negócio
    9. O cliente ou representante do negócio é convidado para todas as reuniões de planejamento
    10. As estimativas de custo das atividades (pontuação) são baseadas nos fatores do projeto como tamanho, complexidade e fatores organizacionais
    11. O trabalho é reajustado às mudanças na disponibilidade da equipe e dos recursos
- Área: Desenvolvimento dirigido a *Story Cards* (Engenharia de Requisitos)
    1. Os requisitos são documentados através de User Stories
    2. As User Stories utilizam uma estrutura única
    3. Existe um plano para gerenciar story cards (Story Card = post-it da user story com pontuação)
    4. Obtém-se conhecimento dos story cards
    5. Obtém-se comprometimento com os story cards
    6. Inconsistências entre Story Cards e requisitos do cliente são facilmente identificáveis
    7. Elicita-se as necessidades do cliente no local
    8. Existe um plano para mudança de story card
    9. Mudanças de requisitos é gerenciada
    10. Story cards são escritas de múltiplos pontos de vista
- Disponibilidade do cliente no local
    1. Cliente no local é um especialista do domínio do negócio
    2. As decisões de negócio são feitas pelo cliente ou pelo seu representante
    3. Obtém-se comprometimento com os story cards do cliente no local

4. Cliente ou o seu representante está sempre disponível

- Introdução ao TDD (*Test Driven Development*)
    1. Utiliza-se um método simples para representação das entidades e seus relacionamentos
    2. Testes unitários são identificados pelos task cards

Após a aplicação do questionário, analisou-se as respostas e os resultados foram organizados como mostra o Quadro 2. Buscou-se evidências documentais sobre as práticas do processo para assim, baseado nesses dados, determinar o nível de aderência da prática abordada na afirmativa no processo de desenvolvimento, podendo ser: TOTAL, PARCIAL OU NENHUM, seguindo as seguintes regras:

- TOTAL:
    - se existirem evidências da prática (ou evidências não se aplicam) e a maioria das respostas for Concordo Totalmente ou Concordo Parcialmente, pois os registros obrigatórios do processo existem e os desenvolvedores têm a compreensão adequada da prática;
- PARCIAL:
    - se existirem evidências da prática (ou evidências não se aplicam) mas a maioria das respostas variar entre Concordo Parcialmente e Discordo Totalmente, pois os registros obrigatórios do processo existem, porém, os desenvolvedores não têm a compreensão adequada da prática;
    - ou, se não existirem evidências da prática, sendo estas obrigatórias, mas a maioria das respostas for Concordo Totalmente ou Concordo Parcialmente, pois os registros obrigatórios do processo não existem mas os desenvolvedores têm a compreensão adequada da prática;
- NENHUMA:
    - se não existirem evidências da prática, sendo estas obrigatórias, e a maioria das respostas variar entre Concordo Parcialmente e Discordo Totalmente, pois os registros obrigatórios do processo não existem e os desenvolvedores não têm a compreensão adequada da prática;

O Quadro 2 abaixo resume as regras para determinação do nível de aderência:

Dessa forma, os resultados obtidos deste primeiro questionário estão dispostos a seguir. Houveram 10 afirmativas com aderência TOTAL (37,04%), 12 com aderência PARCIAL

Quadro 2 – Análise da definição de aderência

		ANÁLISE DAS EVIDÊNCIAS		NÍVEL DE ADERÊNCIA AO AMM II
		Há evidências OU não se aplica	Não há evidências	
ANÁLISE DAS RESPOSTAS	Maioria concorda total ou parcialmente	X		Total
			X	Parcial
	Maioria concorda parcialmente ou discorda	X		Parcial
			X	Nenhuma

Fonte: Próprio autor.

(44,44%) e 5 com aderência NENHUMA (18,52%).

Figura 4 – Análise do questionário parte 1

	Concordo totalmente	Concordo parcialmente	Indiferente ou prefiro não opinar	Discordo parcialmente	Discordo totalmente	Evidencia	Nível de Aderência
A técnica de planejamento (Scrum Planning) é usada para criar planos de projeto	15	0	0	0	0	Sprints e issues registradas no GitLab, Fotos do Kanban Board na wiki do projeto com as features definidas na sprint;	TOTAL
Estima-se o escopo do projeto	14	1	0	0	0	Fotos do Kanban Board na wiki do projeto com as features definidas na sprint e suas pontuações definidas pela equipe de desenvolvimento no Scrum Planning;	TOTAL
Planejamento de Release é usado para criar cronogramas	8	4	0	2	1	Existe uma data de lançamento para versão (final da sprint) e o escopo para tal versão (MRs aprovadas após revisão de código). Porém, essas informações não são claras para todos os membros da equipe.	PARCIAL
A velocidade do projeto é medida	11	4	0	0	0	Gráfico de entrega de pontos disponibilizado pelo analista do NPDS na wiki do projeto no Gitlab.	TOTAL
O projeto é dividido em iterações	14	1	0	0	0	Sprints documentadas nos milestones do Gitlab e Sprint backlog	TOTAL
Estimação é feita pelos desenvolvedores	15	0	0	0	0	Usa-se o Scrum Poker como técnica para determinar as pontuações das USs no Scrum Planning	TOTAL
Estimativas são feitas baseadas em estimativas passadas	12	2	1	0	0	Os desenvolvedores pontuam as issues da nova sprint usando como experiência as pontuações da sprint passada. Alguns desenvolvedores não têm a compreensão de que, considerar os pontos entregues na(s) sprint(s) anterior(es) é um forma de basear em estimativas passsadas	PARCIAL
O planejamento é baseado no valor de negócio	9	6	0	0	0	No planejamento da Sprint, o cliente ou representante define quais features tem maior prioridade para entrega.	TOTAL
O cliente ou representante do negócio é convidado para todas as reuniões de planejamento	2	8	0	3	2	Não existe evidência de convite formal ao cliente para reuniões de planejamento.	NENHUM
As estimativas de custo das atividades (pontuação) são baseadas nos fatores do projeto como tamanho, complexidade e fatores organizacionais	10	3	2	0	0	O conceito de pontos do Scrum está bem compreendido pela maioria dos desenvolvedores como uma medida de complexidade.	TOTAL
O trabalho é reajustado às mudanças na disponibilidade da equipe e dos recursos	10	4	0	0	1	Os desenvolvedores pontuam as issues considerando feriados ou possíveis provas que terão na sprint e assim definem. Porém, isto é feito apenas no Scrum Planning e não durante a sprint.	PARCIAL
Os requisitos são documentados através de User Stories	7	4	0	1	3	Os requisitos são documentados por meio do registro de issues no GitLab e post-its para o Kanban Board onde é utilizado um outro modelo de User Stories. O nível de detalhamento das USs varia muito, de maneira que, frequentemente, as informações contidas não refletem de maneira clara o requisito, gerando dúvidas e retrabalho.	PARCIAL

Fonte: Resultado da pesquisa.

Figura 5 – Análise do questionário parte 2

	Concordo totalmente	Concordo parcialmente	Indiferente ou prefiro não opinar	Discordo parcialmente	Discordo totalmente	Evidencia	Nível de Aderência
As User Stories utilizam uma estrutura única	7	4	3	0	1	As User Stories tem a seguinte estrutura de acordo com fotos do Kanban Board disponibilizadas na wiki de cada projeto: Número da issue registrada no GitLab, quantidade de pontos definidas para esta feature, descrição da feature de acordo com o registro da issue no GitLab.	PARCIAL
Existe um plano para gerenciar story cards (Story Card = post-it da user story com pontuação)	9	2	1	0	3	No quadro kanban são registradas tasks, que são desdobramentos das USs, equivalentes aos story cards no processo adotado. Falta esse entendimento por parte de alguns desenvolvedores. Não há evidência nas ferramentas pois os post-its são descartados ao final da sprint.	PARCIAL
Obtém-se conhecimento dos story cards	10	1	1	2	1	Não há evidência nas ferramentas, alguns desenvolvedores relatam que determinadas issues não têm informações suficientes para compreensão do requisito	PARCIAL
Obtém-se comprometimento com os story cards	11	1	0	1	2	Não há evidência nas ferramentas, mas a maioria dos desenvolvedores compreendem o comprometimento pelo fato de que é o time quem determina a pontuação a ser entregue.	PARCIAL
Inconsistências entre Story Cards e requisitos do cliente são facilmente identificáveis	3	3	3	4	2	Apenas os desenvolvedores que compreendem muito bem os requisitos são capazes de identificar inconsistências.	NENHUM
Elicita-se as necessidades do cliente no local	5	7	2	0	1	Ocorre apenas em parte dos projetos	PARCIAL
Existe um plano para mudança de story card	8	3	3	0	1	A maioria dos desenvolvedores entende que é possível mudar USs durante a sprint. Verificou-se pelo histórico de alterações de USs do Gitlab que algumas USs foram alteradas durante a sprint.	TOTAL
Mudanças de requisitos são gerenciadas	10	2	0	2	1	Alguns desenvolvedores consideram que, ao ocorrer mudanças, às vezes, não há análise de impacto sobre o planejamento. Quando ocorre uma mudança de requisito, a feature é replanejada com a equipe e refeita a pontuação desta feature.	PARCIAL
Story cards são escritas de múltiplos pontos de vista	4	5	3	2	1	A maioria do desenvolvedores relata que os Story Cards são escritas no planning junto ao cliente/representante do cliente e o time de desenvolvimento e o gerente de projetos	PARCIAL
Cliente no local é um especialista do domínio do negócio	5	4	2	4	0	Verificou-se que em todos os projetos realizados, o cliente que participava era também um especialista no domínio.	TOTAL
As decisões de negócio são feitas pelo cliente ou pelo seu representante	7	3	2	1	2	A maioria dos desenvolvedores relatou que a aprovação das features após o fim da sprints e as decisões de prioridades do que deve ser feito sempre tem o aval no cliente/representante do cliente.	NENHUM
Obtém-se comprometimento do cliente ou do seu representante com os story cards	6	5	3	0	1		TOTAL
Cliente ou o seu representante está sempre disponível	0	11	1	2	1	Ocorre indisponibilidade do cliente mas o representante do cliente está sempre disponível.	PARCIAL
Utiliza-se um método simples para representação das entidades e seus relacionamentos	1	2	8	0	4	Não há evidência de uso de método simples para representação das entidades e seus relacionamentos.	NENHUM

Fonte: Resultado da pesquisa.

Figura 6 – Análise do questionário parte 3

	Concordo totalmente	Concordo parcialmente	Indiferente ou prefiro não opinar	Discordo parcialmente	Discordo totalmente	Evidencia	Nível de Aderência
Testes unitários são identificados pelos task cards	5	2	3	3	2	Não há evidência.	NENHUM

Fonte: Resultado da pesquisa.

Os resultados apresentados acima foram utilizados como base para elaboração de um processo com foco em manter os pontos que já apresentam aderência total e melhorar os pontos que ainda não apresentam aderência total ao AMM II.

4.2 Construção da Solução

Após a coleta de dados e análise do nível de aderência ao AMM II, iniciou-se a elaboração do processo de desenvolvimento ágil com base no diagnóstico encontrado e no framework ASSF (QUMER; HENDERSON-SELLERS, 2008). As práticas foram definidadas baseados na consulta ao especialista. O mesmo é gerente de projetos do núcleo com experiência no mercado de trabalho com projetos que tinham alto nível de maturidade. Dessa forma, as práticas definidas para o processo estão dispostas a seguir:

1. Parametrização do Projeto;
2. Kickoff Meeting;
3. Parametrização do Processo;
4. Documentação de requisitos;
5. Ramp-up da aplicação;
6. Sprint Planning SP1;
7. Sprint Planning SP2;
8. Daily Meeting;
9. Sprint Review;
10. Release Test;
11. Release Deployment;
12. Fechamento de Projeto.

As práticas foram posicionadas, com o respectivo impacto de cada uma delas, nas fases do ciclo de vida do software, descrito por Sommerville (2003), conforme apresentado na Figura 7.

O framework disponibiliza um template para identificar os fragmentos do processo. Estes fragmentos podem ser analisados com base nas fases ou nas práticas do processo. Neste trabalho, optou-se pelas práticas do processo por se tratar de um maior nível de detalhe do processo e melhor entendimento dos membros do NPDS. O template está exemplificado no Quadro 3. Assim, os fragmentos do processo definido estão disponíveis no Apêndice A. Tendo as práticas definidas, será necessário validar a agilidade do processo utilizando o 4-DAT.

Figura 7 – Impacto das práticas definidas nas fases do ciclo de vida do software

ID	PRATICA	FASE			
		Concepção	Elaboração	Construção	Transição
1	Parametrização do Projeto	MUITO	POUCO	POUCO	POUCO
2	Kickoff Meeting	MUITO	MEDIO	POUCO	POUCO
3	Parametrização do Processo	MUITO	MEDIO	POUCO	POUCO
4	Documentação de requisitos	MUITO	MUITO	MEDIO	POUCO
5	Ramp-up da aplicação	MEDIO	MUITO	POUCO	POUCO
6	Sprint Planning SP1	POUCO	MEDIO	MUITO	MUITO
7	Sprint Planning SP2	POUCO	MEDIO	MUITO	MUITO
8	Daily Meeting	POUCO	MEDIO	MUITO	MUITO
9	Sprint Review	POUCO	MEDIO	MUITO	MUITO
10	Release Test	POUCO	POUCO	MUITO	MUITO
11	Release Deployment	POUCO	POUCO	MUITO	MUITO
12	Fechamento de Projeto	POUCO	POUCO	POUCO	MUITO

Fonte: Resultado da Pesquisa

Quadro 3 – Template para os fragmentos do processo

ID e Nome	ID único e nome do fragmento do processo.
Descrição e Propósito	Os detalhes relacionados e o propósito do fragmento do processo.
Abstração	Qual mecanismo de abstração (Objeto, agente, serviço, etc.) este fragmento de processo suporta?
Ferramentas e Pessoas	Que tipo de ferramentas e pessoas são necessárias para usar o fragmento de processo de forma bem sucedida?
Estilo de Desenvolvimento	Que estilo de desenvolvimento (Iterativo, rápido) é necessário para usar o fragmento de processo de forma bem sucedida?
Ambiente Físico	Que ambiente físico (co-alocado ou distribuído) é necessário para o fragmento de processo?
Pre e Pós Condições	Quais pré e pós condições devem ser verdadeiras antes e depois da execução do fragmento do processo?
Restrições e Riscos	Quais são as possíveis restrições e riscos anexadas ao fragmento do processo?
Grau de agilidade	Qual o grau de agilidade (medido em relação aos atributos de agilidade) do fragmento do processo?
Valor de negócio	Qual o valor de negócio atribuido ao fragmento do processo?
Alertas	Quais são as possíveis situações que o fragmento do processo não deve ser aplicado?

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

4.3 Validação da Solução

Após a definição do processo, faz-se necessário determinar, conceitualmenteo, o grau de agilidade para cada prática, utilizando a dimensão dois do framework 4-DAT (GILL; HENDERSON-SELLERS, 2006), cujos atributos de agilidade estão descritos no Quadro 4. Para cada atributo deve-se prencher com valores zero ou um. Depois, somar os valores dos atributos e dividi-los por cinco e assim determinar o grau de agilidade da prática.

Quadro 4 – Atributos de agilidade do 4-DAT

1. Flexibilidade (Flexibility)	O método se adapta a mudanças esperadas ou inesperadas?
2. Velocidade (Speed)	O método produz resultados rapidamente?
3. Magreza (Leanness)	O método segue o menor período de tempo, utiliza instrumentos econômicos, simples e de qualidade para produção?
4. Aprendizado (Learning)	O método aplica conhecimento prévio atualizado e experiência para aprender?
5. Responsividade (Responsiveness)	O método exibe sensibilidade (percepção de mudanças)?

Fonte: Próprio autor. Adaptado de Gill e Henderson-Sellers (2006)

Posteriormente, tendo definido os graus de agilidade de cada prática, faz-se necessário definir o grau de agilidade do processo completo de desenvolvimento de software. Este grau é definido a partir da soma dos graus de agilidade de cada atributo de agilidade, tendo seus valores individuais para representar qual grau o processo tem em relação aos atributos mostrados no Quadro 4 e também a partir da soma dos graus de agilidade de cada prática dividido pela quantidade de práticas definidas. Um exemplo pode ser visto na Figura 3 apresentada no terceiro capítulo deste trabalho. Dessa forma, o grau de agilidade da solução definida pode ser visto no Quadro 5

Quadro 5 – Cálculo do Grau de Agilidade pelo 4-DAT (GILL; HENDERSON-SELLERS, 2006)

ID	Nome da Prática	Flexibilidade	Velocidade	Magreza	Aprendizado	Responsividade	TOTAL
1	Parametrização do Projeto	0	1	0	1	0	0,4
2	Kickoff Meeting	1	1	0	1	1	0,8
3	Parametrização do Processo	0	1	0	1	0	0,4
4	Documentação de requisitos	1	1	0	1	1	0,8
5	Ramp-up da aplicação	0	1	1	1	0	0,6
6	Sprint Planning SP1	1	1	1	1	1	1
7	Sprint Planning SP2	1	1	1	1	1	1
8	Daily Meeting	1	1	1	1	1	1
9	Sprint Review	1	1	1	1	1	1
10	Release Test	1	1	0	1	1	0,8
11	Release Deployment	1	1	0	0	1	0,6
12	Fechamento de Projeto	0	1	0	1	0	0,4
GRAUS DE AGILIDADE		0,67	1	0,42	0,92	0,67	0,73

Fonte: Próprio autor

O grau de agilidade obtido do processo foi de 0.73. Para uma comparação, Gill e Henderson-Sellers (2006) demonstra o cálculo da dimensão dois do 4-DAT no método SCRUM (SCHWABER, 1997) onde foi obtido o grau de agilidade 0.80, tendo uma diferença de 0.07.

Pode-se observar também que o processo foi elaborado com foco em melhorar as afirmativas do diagnóstico na qual apresentaram nível NENHUM de aderência. O Quadro 6 apresenta as afirmativas com nenhuma aderência e as práticas criadas para aumentar o nível de adesão.

Tendo validado a solução obtida, faz-se necessário a implantação da mesma no

Quadro 6 – Associação de afirmativa com nível de aderência NENHUM e prática definida

Afirmativa com NENHUM de Aderência	**Prática definida associada**
O cliente ou representante do negócio é convidado para todas as reuniões de planejamento	6 - Sprint Planning SP1
Inconsistências entre Story Cards e requisitos do cliente são facilmente identificáveis	6 - Sprint Planning SP1 7 - Sprint Planning SP2
As decisões de negócio são feitas pelo cliente ou pelo seu representante	6 - Sprint Planning SP1
Utiliza-se um método simples para representação das entidades e seus relacionamentos	4 - Documentação de requisitos
Testes unitários são identificados pelos task cards	10 - Release Test

Fonte: Próprio autor

NPDS para posterior avaliação dos membros inseridos no processo.

4.4 Implementação (implantação) da solução

Com o processo definido e validado, o mesmo deve ser implantado para os projetos desenvolvidos no NPDS. No momento da implantação do processo, o NPDS dispões de sete membros desenvolvedores dispostos em dois projetos. Todos estavam entre o sétimo e oitavo semestre de suas graduações: cinco de Ciência da Computação e dois de Sistemas de Informação. A solução foi aplicada e após três meses foi realizada a avaliação da solução para verificar melhora em relação ao processo anterior.

4.5 Avaliação da Implementação

Depois do período determinado para implantação do processo de desenvolvimento, foi realizado um segundo estudo de caso observacional com o objetivo de verificar o nível de maturidade ágil e o grau de adesão às práticas ágeis dos projetos *software* do NPDS no período 2019.1. Então, utilizou-se o mesmo questionário de vinte e sete questões referentes ao nível II do AMM (*Agile Maturiry Model*) (PATEL; RAMACHANDRAN, 2009).

O questionário e a análise utilizada estão descritos na Seção 4.1. Os resultados obtidos pela aplicação e avalaiação do formulário após a implementação do process estão dispostos a seguir:

Figura 8 – Análise da segunda aplicação do questionário parte 1

	Concordo totalmente	Concordo parcialmente	Indiferente ou prefiro não opinar	Discordo parcialmente	Discordo totalmente	Evidencia	Nível de Aderência
A técnica de planejamento (Scrum Planning) é usada para criar planos de projeto	7	0	0	0	0	Sprints e issues registradas no GitLab, Fotos do Kanban Board na wiki do projeto com as features definidas na sprint;	TOTAL
Estima-se o escopo do projeto	5	1	1	0	0	Fotos do Kanban Board na wiki do projeto com as features definidas na sprint e suas pontuações definidas pela equipe de desenvolvimento no Scrum Planning;	TOTAL
Planejamento de Release é usado para criar cronogramas	4	2	0	1	0	Existe uma data de lançamento para versão (final da sprint) e o escopo para tal versão (MRs aprovadas após revisão de código). Alguns dos projetos, por estarem em seu início ainda não passaram por um planejamento para Release	PARCIAL
A velocidade do projeto é medida	6	0	1	0	0	Gráfico de entrega de pontos disponibilizado pelo analista do NPDS na wiki do projeto no Gitlab.	TOTAL
O projeto é dividido em iterações	7	0	0	0	0	Sprints documentadas nos milestones do Gitlab e Sprint backlog	TOTAL
Estimação é feita pelos desenvolvedores	5	1	1	0	0	Usa-se o Scrum Poker como técnica para determinar as pontuações das USs no Scrum Planning	TOTAL
Estimativas são feitas baseadas em estimativas passadas	4	2	1	0	0	Os desenvolvedores pontuam as issues da nova sprint usando como experiência as pontuações da sprint passada. Alguns desenvolvedores não têm a compreensão de que, considerar os pontos entregues na(s) sprint(s) anterior(es) é um forma de basear em estimativas passsadas	TOTAL
O planejamento é baseado no valor de negócio	7	0	0	0	0	No planejamento da Sprint, o cliente ou representante define quais features tem maior prioridade para entrega.	TOTAL

Fonte: Próprio autor.

Figura 9 – Análise da segunda aplicação do questionário parte 2

O cliente ou representante do negócio é convidado para todas as reuniões de planejamento	5	1	0	1	0	Os clientes são convidados por e-mail apenas para as reuniões onde sua presença é indispensável por se tratar de sua indisponibilidade de horário. Porém o representante está sempre presente	TOTAL
As estimativas de custo das atividades (pontuação) são baseadas nos fatores do projeto como tamanho, complexidade e fatores organizacionais	3	2	1	1	0	O conceito de pontos do Scrum está bem compreendido pela maioria dos desenvolvedores como uma medida de complexidade.	TOTAL
O trabalho é reajustado às mudanças na disponibilidade da equipe e dos recursos	3	2	0	2	0	Os desenvolvedores pontuam as issues considerando feriados ou possíveis provas que terão na sprint e assim definem. Porém, isto é feito apenas no Scrum Planning e não durante a sprint.	PARCIAL
Os requisitos são documentados através de User Stories	7	0	0	0	0	Os requisitos são documentados por meio do registro de issues no GitLab e post-its para o Kanban Board onde é utilizado um outro modelo de User Stories. O nível de detalhamento das USs varia muito, de maneira que, frequentemente, as informações contidas não refletem de maneira clara o requisito, gerando dúvidas e retrabalho.	TOTAL
As User Stories utilizam uma estrutura única	6	1	0	0	0	As User Stories tem a seguinte estrutura de acordo com fotos do Kanban Board disponibilizadas na wiki de cada projeto: Número da issue registrada no GitLab, quantidade de pontos definidas para esta feature, descrição da feature de acordo com o registro da issue no GitLab.	TOTAL
Existe um plano para gerenciar story cards (Story Card = post-it da user story com pontuação)	5	1	1	0	0	No quadro kanban são registradas tasks, que são desdobramentos das USs, equivalentes aos story cards no processo adotado. Falta esse entendimento por parte de alguns desenvolvedores. Não há evidência nas ferramentas pois os post-its são descartados ao final da sprint.	TOTAL
Obtém-se conhecimento dos story cards	6	1	0	0	0	Na ferramenta Gitlab as issues são detalhadas no título para melhro entendimento	TOTAL
Obtém-se comprometimento com os story cards	7	0	0	0	0	Os desenvolvedores compreendem o comprometimento pelo fato de que é o time quem determina a pontuação a ser entregue.	TOTAL
Inconsistências entre Story Cards e requisitos do cliente são facilmente identificáveis	4	3	0	0	0	Os desenvolvedores compreendem muito bem os requisitos e identificam inconsistências.	TOTAL
Elicita-se as necessidades do cliente no local	4	3	0	0	0	O cliente ou representante estão presentes ao elicitar as necessidades sempre que possível	TOTAL

Fonte: Próprio autor.

Figura 10 – Análise da segunda aplicação do questionário parte 3

Existe um plano para mudança de story card	4	2	0	0	0	A maioria dos desenvolvedores entende que é possível mudar USs durante a sprint. Verificou-se pelo histórico de alterações de USs do Gitlab que algumas USs foram alteradas durante a sprint.	TOTAL
Mudanças de requisitos são gerenciadas	4	3	0	0	0	Alguns desenvolvedores consideram que, ao ocorrer mudanças, às vezes, não há análise de impacto sobre o planejamento. Quando ocorre uma mudança de requisito, a feature é replanejada com a equipe e refeita a pontuação desta feature.	TOTAL
Story cards são escritas de múltiplos pontos de vista	2	3	2	0	0	A maioria do desenvolvedores relata que os Story Cards são escritas no planning junto ao cliente/representante do cliente e o time de desenvolvimento e o gerente de projetos	PARCIAL
Cliente no local é um especialista do domínio do negócio	6	0	0	0	1	Verificou-se que em todos os projetos realizados, o cliente que participava era também um especialista no domínio.	TOTAL
As decisões de negócio são feitas pelo cliente ou pelo seu representante	6	1	0	0	0	A maioria dos desenvolvedores relatou que a aprovação das features após o fim da sprints e as decisões de prioridades do que deve ser feito sempre tem o aval no cliente/representante do cliente.	TOTAL
Obtém-se comprometimento do cliente ou do seu representante com os story cards	5	2	0	0	0	O cliente ou representante tem o entendimento de, ao	TOTAL
Cliente ou o seu representante está sempre disponível	4	2	0	0	1	Ocorre indisponibilidade do cliente mas o representante do cliente está sempre disponível.	TOTAL
Utiliza-se um método simples para representação das entidades e seus relacionamentos	4	0	2	0	1	São elaborados rascunhos em quadros na criação de entidades e seus relacionamentos, feito uma fotografia e disponibilizado na wiki do projeto	PARCIAL
Testes unitários são identificados pelos task cards	4	0	0	2	0	Definiu-se um membro bolsista para a realização dos testes unitários baseados nos task cards	PARCIAL

Fonte: Próprio autor.

Observou-se que houve melhoria em relação a primeira aplicação do questionário. Um dos pontos principais observados nesta segunda análise é a melhora o entendimento do processo por parte dos membros, a aproximação com o cliente e a execução de testes para melhoria da qualide dos softwares desenvolvidos, pontos esses que são bastante importantes na metodologia ágil, mostrando assim um aumento na maturidade ágil dos membros.

5 CONCLUSÕES E TRABALHOS FUTUROS

Como observado neste trabalho, demonstrou-se que é possível garantir o uso de processos ágeis de desenvolvimento de software em ambientes acadêmicos e melhorar formação em processos ágeis dos concludentes em cursos de TI. Os resultados apresentados mostram o contraste entre o processo anterior e o processo ágil definido em relação a execução e entedimento das práticas do processo. Pode-se observar também que o nível de maturidade do processo alcançou um bom grau de agilidade em comparação a metodologias ágeis já utilizadas no mercado.

Para atingir o objetivo de estabelecer e implantar um processo de desenvolvimento ágil no NPDS, fez-se necessário encontrar um diagnóstico para que o processo definido fosse construído para sanar as deficiências encontradas. Este diagnóstico foi obtido através de um questionário extraido do nível II do AMM e partir disso, definido o nível de aderência para cada afirmativa abordada no questionário e mostrando as áreas com deficiência, definir o processo.

O processo definido apresentado neste trabalho é composto por doze práticas estabelecidas através de uma investigação no contexto do NPDS para diagnosticar possíveis fraquezas para que o novo processo fosse definido de maneira objetiva. As práticas definidas também levaram em conta a agilidade do processo e seu impacto nos projetos desenvolvidos dentro do núcleo. As práticas foram definidas a partir do template e sugestões do ASSF para melhor detalhamento e entendimento do processo.

Tendo o processo definido, ele deve ser ágil. Assim, a utitilização do 4-DAT premitir avaliar conceitualmente o grau de agilidade do processo. O processo teve 0.73 de grau de agilidade, que comparado a métodos já contemplados no mercado, como o SCRUM (SCHWABER, 1997) que teve 0.80 de grau de agilidade segundo o 4-DAT (GILL; HENDERSON-SELLERS, 2006), apresenta diferença de 0.07.

Após a construção do processo faz-se necessário que ele seja entendido e executado no contexto que se deseja mellhorar. Assim, apresentou-se às equipes de desenvolvimento o processo e seus fragmentos, detalhando cada prática e seu impacto no fluxo de trabalho dos núcleo.

Dessa forma, após o período de utilização do processo, mostrou-se a evolução a partir dos resultados na segunda aplicação do questionário e respectiva elaboração do diagnóstico, onde foi possível observar que nenhuma das afirmativas tiveram nível de aderência NENHUM.

Portanto, o trabalho mostra que o uso do AMM, 4-DAT e ASSF apresentaram bons

resultados como métodos para alcançar maturidade ágil de equipes de desenvolvimento de software.

5.1 Limitações e Trabalhos Futuros

Neste trabalho, observou-se que, ao aplicar os questionários para obtenção do diagnóstico do processo e dos resultados após a definição do método, a amostra se condensava em alunos de apenas dois cursos e sempre do sétimo e oitavo período. Também, na definição do diagnóstico, utilizou-se apenas das afirmativas extraidas do nível II do AMM. Para a validação conceitual do processo, foi utilizado a dimensão II do 4-DAT. Além disso, também destaca-se a aplicação do método apenas em um ambiente acadêmico.

Dessa forma, tem-se como trabalhos futuros a aplicação deste método com níveis mais altos do AMM para uma maior abrangência do processo e um diagnóstico mais elaborado; aplicação do trabalho para amostras de outros cursos de TI com alunos de períodos diferentes; utilizar das demais dimensões do 4-DAT e até de outros métodos de avaliação de processos para um parecer mais detalhista e, por fim, aplicar o trabalho a ambientes no mercado de trabalho para uma maior cobertura do método.

REFERÊNCIAS

BARTH, F. J.; BURD, L.; PIMENTEL, M. Escritório de projetos: simulando o ambiente de projetos de software em cursos de tecnologia. In: **XX Workshop sobre Educação em Computação-WEI**. [S.l.: s.n.], 2012.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R. *et al.* Manifesto for agile software development. 2001.

BEGOSSO, L. R.; BEGOSSO, L. C.; POLETTO, A. S. R. S.; CUNHA, D. S. da; LIMA, F. C. de. Programa de residência em software. In: **XIX Workshop de Educaç ao em Informática., Natal, Brasil**. [S.l.: s.n.], 2011.

GILL, A.; HENDERSON-SELLERS, B. Measuring agility and adaptibility of agile methods: A 4 dimensional analytical tool. In: IADIS PRESS. **The IADIS international conference on applied computing 2006**. [S.l.], 2006.

GILL, A.; HENDERSON-SELLERS, B.; MCBRIDE, T. Agile adoption and improvement model. In: POLYTECHNIC UNIVERSITY OF VALENCIA. **European, Mediterranean and Middle Eastern Conference on Information Systems**. [S.l.], 2007.

GONÇALVES, E. J.; BEZERRA, C. I.; ALMENDRA, C. C.; SAMPAIO, A. L.; VASCONCELOS, D. R. Núcleo de práticas em informática: Contribuindo para a formaçao em sistemas de informaçao através do desenvolvimento de projetos de software. In: **Anais do WEI-XXI Workshop sobre Educação em Computação, Maceió, Brasil**. [S.l.: s.n.], 2013.

LIKERT, R. A method of constructing an attitude scale. **Scaling: A sourcebook for behavioral scientists**, Aldine Publishing, Chicago, p. 233–243, 1974.

PATEL, C.; RAMACHANDRAN, M. Agile maturity model (amm): A software process improvement framework for agile software development practices. **International Journal of Software Engineering, IJSE**, v. 2, n. 1, p. 3–28, 2009.

PECKHAM, J.; BATSON, T. Web development group: an enterprising campus-based internship program for cs majors. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 19, n. 5, p. 17–24, 2004.

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

QUMER, A.; HENDERSON-SELLERS, B. A framework to support the evaluation, adoption and improvement of agile methods in practice. **Journal of Systems and Software**, Elsevier, v. 81, n. 11, p. 1899–1919, 2008.

QUMER, A.; HENDERSON-SELLERS, B.; MCBRIDE, T. Agile adoption and improvement model. Citeseer, 2007.

RAO, K. N.; NAIDU, G. K.; CHAKKA, P. A study of the agile software development methods, applicability and implications in industry. **International Journal of Software Engineering and its applications**, v. 5, n. 2, p. 35–45, 2011.

ROYCE, W. W. Managing the development of large software systems: concepts and techniques. In: IEEE COMPUTER SOCIETY PRESS. **Proceedings of the 9th international conference on Software Engineering**. [S.l.], 1987. p. 328–338.

SCHWABER, K. Scrum development process. In: **Business object design and implementation**. [S.l.]: Springer, 1997. p. 117–134.

SOMMERVILLE, I. **Engenharia de Software, Tradução de André Maurício de Andrade Ribeiro; Revisão técnica de Kechi Hirama**. [S.l.]: São Paulo, Addison Wesley, 2003.

WIERINGA, R. J. **Design science methodology for information systems and software engineering**. [S.l.]: Springer, 2014.

APÊNDICE A – FRAGMENTOS DO PROCESSO DEFINIDO

Quadro 7 – Fragmento do Processo 1

<table>
<tr><td>ID e Nome
(ID & Name)</td><td colspan="2">1</td><td>Parametrização
do Projeto</td></tr>
<tr><td>Descrição de Propósito
(Description & Purpose)</td><td colspan="3">Essa prática pode ser executa apenas pelo gerente de projetos
a partir de dados conhecidos do projeto ou através de reunião.
É necessário:
- Estabelecer o tempo do projeto
- Estabelecer a equipe do projeto
- Se já houver clareza suficiente, estabelecer o escopo do projeto
É importante formalizar de maneira simples (ex.: email) e
incluir na documentação do projeto</td></tr>
<tr><td>Abstração
(Abstraction)</td><td colspan="3">Objeto</td></tr>
<tr><td>Ferramentas e Pessoas
(Tools & People)</td><td colspan="3">- Cliente
- Gerente de Projetos
- Editor de texto para registrar</td></tr>
<tr><td>Estilo de
Desenvolvimento
(Development Style)</td><td colspan="3">Iterativo</td></tr>
<tr><td>Ambiente Físico
(Physical Environment)</td><td colspan="3">Qualquer ambiente</td></tr>
<tr><td>Pré e Pós Condições
(Pre and Post Conditions)</td><td colspan="3">Pre:
- Conhecer os recursos disponíveis para a execução do projeto
Post:
- Email enviado ao cliente deixando ciente dos recursos do projeto
- Dados registrados da wiki do projeto</td></tr>
<tr><td>Restrições e Riscos
(Constraints and Risks)</td><td colspan="3">Restrição: é necessário que o cliente
já tenha estabelecido claramente os recursos do projeto
Risco: a falta de clareza nos recursos
disponíveis pode gerar alto risco para o planejamento</td></tr>
<tr><td rowspan="6">Grau de Agilidade
(Degree of agility)</td><td>Atributo
(Attribute)</td><td>Valor
(Value)</td><td>Grau
(Degree)</td></tr>
<tr><td>1. Flexibilidade (Flexibility)</td><td>0</td><td rowspan="5">0.4</td></tr>
<tr><td>2. Velocidade (Speed)</td><td>1</td></tr>
<tr><td>3. Magreza (Leanness)</td><td>0</td></tr>
<tr><td>4. Aprendizado (Learning)</td><td>1</td></tr>
<tr><td>5. Responsividade (Responsiveness)</td><td>0</td></tr>
<tr><td>Valor de Negócio
(Business Value)</td><td colspan="3">- Controle da expectativa do cliente
- Transparência
- Formalização ágil dos recursos estabelecidos
- Compartilhamento rápido de informações do projeto</td></tr>
<tr><td>Alertas
(Alerts)</td><td colspan="3"></td></tr>
</table>

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 8 – Fragmento do Processo 2

ID e Nome (ID & Name)	2	Kickoff Meeting	
Descrição de Propósito (Description & Purpose)	Reunião realizada entre equipe e cliente onde o foco principal é permitir que o cliente se expresse e exponha as suas necessidades, cabendo à equipe ajudá-lo a entender essas necessidades e propor, em alto nível, soluções e estabelecer restrições quanto às soluções propostas		
Abstração (Abstraction)	Serviço		
Ferramentas e Pessoas (Tools & People)	- Cliente - Analista experiente - Pode ser utilizado desde ferramentas mais simples como papel, quadro e caneta/pincel até software para prototipacao rápida - É importante tb ter acesso à internet para possivelmente apresentar softwares os solucoes para uso de comparacao - Identificar e definir os stakeholders que devem validar o projeto - Determinar uma agenda para participação no mínimo quinzenal do cliente ou do responsável pela validação do produto		
Estilo de Desenvolvimento (Development Style)	Iterativo		
Ambiente Físico (Physical Environment)	Encontro deve ser preferencialmente presencial		
Pré e Pós Condições (Pre and Post Conditions)	PRE: O cliente deve estar ciente de que é importante já ter refletido sobre as suas necessidades e possíveis soluções já existente que ele acredita que podem resolver o problema POS: Gerar os seguintes artefatos: 1 - Esboço de solução (diagramas, mockups, etc) 2 - Texto descritivo da visão preliminar de escopo 3 - E-mail para o cliente contendo os artefatos 1 e 2 4 - Armazenar os artefatos 1 e 2 na wiki do projeto		
Restrições e Riscos (Constraints and Risks)	Restrições: - O analista deve ser experiente para ter habilidade para, já nesse momento, definir decisões arquiteturais relevantes do projeto - O cliente deve ter consigo (ou ser ele próprio) o usuário final do sistemas e pessoas diretamente envolvidas no processo		
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	1	0.8
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	0	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	1	
Valor de Negócio (Business Value)	- Passa ao cliente uma boa impressão devido à agilidade na visão da solução - Desburocratiza a elaboração de documentos tanto para o cliente como para a equipe - Compartilhamento rápido de informações do projeto		
Alertas (Alerts)	- Se o cliente exigir mais rigor na especificação, documentação e aceitação da solução proposta - Se a complexidade da aplicação for muito alta será necessária a elaboração de documentos mais robustos e aceitação mais clara do cliente		

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 9 – Fragmento do Processo 3

ID e Nome (ID & Name)	3	Parametrização do Processo	
Descrição de Propósito (Description & Purpose)	Com a equipe definida, o gerente de projeto deve, em consenso com os membros da equipe, definir parâmetros que irão definir o processo: - Tamanho das Sprints - Datas de Planning e Review - Ferramentas manuais e tecnologias de apoio ao processo (quadro, gitlab, etc) - Definir escopo de referência para o Planning Poker - Definir pessoas e respectivos papéis no processo		
Abstração (Abstraction)	Objeto		
Ferramentas e Pessoas (Tools & People)	- Gerente de Projetos - Equipe - Wiki do projeto		
Estilo de Desenvolvimento (Development Style)	Iterativo		
Ambiente Físico (Physical Environment)	Sala do projeto		
Pré e Pós Condições (Pre and Post Conditions)	Pre: - o gerente e pelo menos 1 membro da equipe terem maturidade e experiência com processos ágeis - tecnologias e definições básica de arquitetura definidas - alinhamento com a agenda do cliente Pos: Atributos do processo definidos e registrados na wiki do projeto		
Restrições e Riscos (Constraints and Risks)	Riscos: ausência de membros experientes em processos e nas tecnologias poderá acerretar na necessidade de reparametrizar o processo com frequência		
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	0	0.4
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	0	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	0	
Valor de Negócio (Business Value)	- Transparência do trabalho da equipe - Planejamento claro - Compartilhamento rápido de informações do projeto		
Alertas (Alerts)			

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 10 – Fragmento do Processo 4

<table>
<tr><td>ID e Nome
(ID & Name)</td><td colspan="3">4</td><td>Documentação de Requisitos</td></tr>
<tr><td>Descrição de Propósito
(Description & Purpose)</td><td colspan="4">Durante todo o projeto é importante o registro de todos os artefatos que possam ajudar a entender as regras de negócio, os casos de uso e os requistos não funcionais do sistema:
Esta pratica consiste em registrar na wiki do projeto todos os artefatos úteis ao entedimento do projeto, por exemplo:
- Diagramas
- Fotos de quadro ou papeis com esboços de soluções
- Documentos utilizados manualmente pelo cliente que servam como base para formulário (ou fotos dos mesmo)
- Gravações de reuniões, conversas ou vídeos
- Esboço da arquitetura do software;
- Descrição de regras de negócio
- Riscos envolvidos no projeto
- Descrião de requisitos não funcionais como: tecnologias, plataformas de execução e requisitos mínimos do sistema</td></tr>
<tr><td>Abstração
(Abstraction)</td><td colspan="4">Objeto</td></tr>
<tr><td>Ferramentas e Pessoas
(Tools & People)</td><td colspan="4">- Gerente de Projetos
- Analista
- Equipe
- Wiki do projeto</td></tr>
<tr><td>Estilo de Desenvolvimento
(Development Style)</td><td colspan="4">Iterativo</td></tr>
<tr><td>Ambiente Físico
(Physical Environment)</td><td colspan="4">Qualquer ambiente</td></tr>
<tr><td>Pré e Pós Condições
(Pre and Post Conditions)</td><td colspan="4">Pre: possuir informações confiáveis do projeto
Pós: artefatos registrado na wiki de forma legível e organizada</td></tr>
<tr><td>Restrições e Riscos
(Constraints and Risks)</td><td colspan="4">Risco: indisponibilidade ou perda das informações</td></tr>
<tr><td rowspan="6">Grau de Agilidade
(Degree of agility)</td><td>Atributo
(Attribute)</td><td colspan="2">Valor
(Value)</td><td>Grau
(Degree)</td></tr>
<tr><td>1. Flexibilidade (Flexibility)</td><td colspan="2">1</td><td rowspan="5">0.8</td></tr>
<tr><td>2. Velocidade (Speed)</td><td colspan="2">1</td></tr>
<tr><td>3. Magreza (Leanness)</td><td colspan="2">0</td></tr>
<tr><td>4. Aprendizado (Learning)</td><td colspan="2">1</td></tr>
<tr><td>5. Responsividade (Responsiveness)</td><td colspan="2">1</td></tr>
<tr><td>Valor de Negócio
(Business Value)</td><td colspan="4">- Transparência
- Conhecmiento compartilhado e acessível</td></tr>
<tr><td>Alertas
(Alerts)</td><td colspan="4">É preciso estar atento para não negligenciar artefatos importantes e esquecer de realizar os registros</td></tr>
</table>

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 11 – Fragmento do Processo 5

ID e Nome (ID & Name)	5	Ramp-up da Aplicação	
Descrição de Propósito (Description & Purpose)	Escolhe-se um ou mais membros da equipe de maior experiência técnica para que, de acordo com as tecnologias e arquitetura definida, ele possa fazer a configuração do ambiente, a estrutura inicial da aplicação, deixando o projeto adequado para o desenvolvimento paralelo e cooperativo		
Abstração (Abstraction)	Serviço		
Ferramentas e Pessoas (Tools & People)	- Membros mais experientes da equipe - Gitlab - Ambiente de desenvolvimento (IDEs, editores de texto, máquinas virtuais, etc)		
Estilo de Desenvolvimento (Development Style)	Rápido		
Ambiente Físico (Physical Environment)	Sala do projeto		
Pré e Pós Condições (Pre and Post Conditions)	Pre: - Arquitetura e tecnologias bem definidas Pos: - Código fonte da estrutura base da aplicação e demais configurações		
Restrições e Riscos (Constraints and Risks)	A eficiência esperada pela prática pode ser comprometida caso nenhum dos membros da equipe tenha experiência adequada para elaborar uma boa estrutura base para a aplicação		
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	0	0.6
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	1	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	0	
Valor de Negócio (Business Value)	- Entrega rápida - Redução de riscos		
Alertas (Alerts)			

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 12 – Fragmento do Processo 6

ID e Nome (ID & Name)	6	Sprint Planning SP1	
Descrição de Propósito (Description & Purpose)	Trata-se da reunião de planejamento da Sprint, o seu objetivo é decidir o que será feito na Sprint. Para isso, o Product Owner apresenta para a equipe o que é mais prioritário no Backlog do Produto, e explica-os detalhadamente até que a equipe tenha um claro entendimento do requisito, e as user stories são definidas		
Abstração (Abstraction)	Serviço		
Ferramentas e Pessoas (Tools & People)	- Cliente - Equipe - Gerente de Projetos - Ferramentas de apoio a exposição de ideias: quadro, papel, software de mockup, etc		
Estilo de Desenvolvimento (Development Style)	Iterativo		
Ambiente Físico (Physical Environment)	Sala do projeto, videoconferencia		
Pré e Pós Condições (Pre and Post Conditions)	Pre: - O cliente dever ter uma ideia clara dos requisitos desejados Pos: - Requisitos compreendidos e user stories elaboradas e registradas no gitlab		
Restrições e Riscos (Constraints and Risks)	A efeciencia na elaboração e no entendimento dos requisitos pode ser compromentida caso o PO não tenha boa "didática"e clareza do que deseja, ou, caso não haja pessoas no time com boa capacidade de análise		
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	1	1
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	1	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	1	
Valor de Negócio (Business Value)	- Transparência - Rapidez - Compatilhamento de conhecimento		
Alertas (Alerts)			

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 13 – Fragmento do Processo 7

<table>
<tr><td>ID e Nome
(ID & Name)</td><td colspan="3">7</td><td>Sprint Planning SP2</td></tr>
<tr><td>Descrição de Propósito
(Description & Purpose)</td><td colspan="4">É a parte da Reunião de Planejamento da Sprint onde o time deve, entre seus membros:
- executar o Planning Poker para pontuar as user stories definidas
- quebrar as user stories em tarefas diárias
- limitar quais o time se compromete a entregar na sprint
- registrar os tickets de user stories e tarefas no quadro kanban
- Incluir as user stories selecionadas na sprint referente no gitlab</td></tr>
<tr><td>Abstração
(Abstraction)</td><td colspan="4">Serviço</td></tr>
<tr><td>Ferramentas e Pessoas
(Tools & People)</td><td colspan="4">- Equipe
- Aplicativo par planing poker
- Caneta e Post-its
- Quabro Kanban</td></tr>
<tr><td>Estilo de
Desenvolvimento
(Development Style)</td><td colspan="4">Iterativo</td></tr>
<tr><td>Ambiente Físico
(Physical Environment)</td><td colspan="4">Sala do projeto</td></tr>
<tr><td>Pré e Pós Condições
(Pre and Post Conditions)</td><td colspan="4">Pre:
- Backlog com quantidade de user stories no mínimo suficiente para toda a sprint
Pos:
- Quadro kanban contendo as USs e seuas respectivas pontuaçóes, e as tarefas
- Gitlab com a sprint referente preenchida</td></tr>
<tr><td>Restrições e Riscos
(Constraints and Risks)</td><td colspan="4">Risco:
- Planejamento inadequado do escopo: comprometer-se com muito mais ou
com muito menos do que o time é capaz de entregar</td></tr>
<tr><td rowspan="6">Grau de Agilidade
(Degree of agility)</td><td>Atributo
(Attribute)</td><td colspan="2">Valor
(Value)</td><td>Grau
(Degree)</td></tr>
<tr><td>1. Flexibilidade (Flexibility)</td><td colspan="2">1</td><td rowspan="5">1</td></tr>
<tr><td>2. Velocidade (Speed)</td><td colspan="2">1</td></tr>
<tr><td>3. Magreza (Leanness)</td><td colspan="2">1</td></tr>
<tr><td>4. Aprendizado (Learning)</td><td colspan="2">1</td></tr>
<tr><td>5. Responsividade (Responsiveness)</td><td colspan="2">1</td></tr>
<tr><td>Valor de Negócio
(Business Value)</td><td colspan="4">- Transparência
- Planejamento consistente</td></tr>
<tr><td>Alertas
(Alerts)</td><td colspan="4"></td></tr>
</table>

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 14 – Fragmento do Processo 8

ID e Nome (ID & Name)	8	Daily Meeting	
Descrição de Propósito (Description & Purpose)	Reunião realizada pela equipe no início do expediente que geralmente tem duração de 15 minutos. Tem como objetivo principal identificar impedimentos. Nela são abordadas perguntas para cada membro como: O que foi feito ontem? O que irá fazer hoje? Quais as dificuldades e/ou impedimentos encontrados para a realização das tarefas?		
Abstração (Abstraction)	Serviço		
Ferramentas e Pessoas (Tools & People)	- Equipe - Caneta e Post-its - Quadro Kanban		
Estilo de Desenvolvimento (Development Style)	Iterativo		
Ambiente Físico (Physical Environment)	Sala do Projeto		
Pré e Pós Condições (Pre and Post Conditions)	Pre: Equipe presente Pos: Impedimentos identificados e registrados no quadro kanban		
Restrições e Riscos (Constraints and Risks)			
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	1	1
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	1	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	1	
Valor de Negócio (Business Value)	- Planejamento - Controle de riscos - Compartilhamento de informações		
Alertas (Alerts)			

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 15 – Fragmento do Processo 9

ID e Nome (ID & Name)	9	Sprint Review	
Descrição de Propósito (Description & Purpose)	Ao chegar no fim do período determinado para o término da Sprint, a equipe se reune com o gerente de projetos e o cliente (PO) para validar as funcionalidades listadas no Backlog da Sprint. Nesta reunião, o que foi desenvolvido é mostrado ao cliente juntamente com a explicação da funcionalidade e o cliente expressa se aquilo é válido de acordo com o que foi pedido. Ainda, faz-se a análise de risco em relação ao backlog restante e analisa-se também o desempenho demonstrado na Sprint. Um membro responsável alimenta os gráficos que mostram o desempenho da equipa e as registra na wiki juntamente com a imagem final do quadro kanban		
Abstração (Abstraction)	Serviço		
Ferramentas e Pessoas (Tools & People)	- Cliente - Equipe - Gerente de Projetos - Versão do Projeto em Funcionamento - Quadro Kanban - Wiki do projeto - Planilha de indicadores		
Estilo de Desenvolvimento (Development Style)	Iterativo		
Ambiente Físico (Physical Environment)	- Encontro deve ser preferencialmente presencial, na sala do projeto ou sala de reuniões		
Pré e Pós Condições (Pre and Post Conditions)	Pre: - versão executável do sistema - status atualizado das user stories Pos: - user stories aceitas e rejeitadas - registro do quadro kanban atualizado na wiki - planilha de indicadores atualizada		
Restrições e Riscos (Constraints and Risks)	Para uma validação justa das USs é importante que as mesmas tenham critérios de aceitação claros, a sua ausência pode gerar insatisfação tanto para o PO como para a equipe		
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	1	1
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	1	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	1	
Valor de Negócio (Business Value)	- Feedback do cliente - Antecipação de falhas - Gerência da expectativa do cliente - Percepção real da evolução do projeto		
Alertas (Alerts)			

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 16 – Fragmento do Processo 10

ID e Nome (ID & Name)	10	Release Test	
Descrição de Propósito (Description & Purpose)	A qualquer momento da Sprint a equipe pode gerar uma nova release do software. Para isso, deve-se registrar no GitLab a tag referente e o Release Notes da versão explicando as uses stories implementadas, melhorias e os bugs corrigidos. A nova release é implantada em um ambiente de testes e envia-se um email para a equipe de testes solicitando os testes. A equipe de testes deve executar os testes e ao final da execução liberar um relatório informando se a versão está aprovada ou não, a justificativa e quais casos de teste falharam, e a situação dos antigos e dos novos bugs. Para controle interno, a equipe de testes deve atualizar a planilha de bugs		
Abstração (Abstraction)	Serviço		
Ferramentas e Pessoas (Tools & People)	- Gerente de Configuração - Versão do software disponível para teste - Gitlab: tag e release notes - Email - Equipe de Teste - TestLink para execução dos teste - Relatório de resultados de testes de versão - Planilha de Bugs		
Estilo de Desenvolvimento (Development Style)	Iterativo		
Ambiente Físico (Physical Environment)	As equipes podem estar separadas, mas de preferência nas salas do projeto		
Pré e Pós Condições (Pre and Post Conditions)	Pre: - Versão do software com novas features, melhorias e correçoes Pos: - Relatório de execução de testes da versão elaborado - Planilha de bugs atualizada		
Restrições e Riscos (Constraints and Risks)	A equipe de testes precisa entender bem o que precisa ser testado e ter casos de teste com qualidade adequada para não comprometer a qualidade do produto		
Grau de Agilidade (Degree of agility)	Atributo (Attribute)	Valor (Value)	Grau (Degree)
	1. Flexibilidade (Flexibility)	1	0.8
	2. Velocidade (Speed)	1	
	3. Magreza (Leanness)	0	
	4. Aprendizado (Learning)	1	
	5. Responsividade (Responsiveness)	1	
Valor de Negócio (Business Value)	- Controle rigoroso das versões do software - Garantia da qualidade do produto - Antecipação de falhas		
Alertas (Alerts)			

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 17 – Fragmento do Processo 11

<table>
<tr><td>ID e Nome
(ID & Name)</td><td colspan="2">11</td><td>Release Deployment</td></tr>
<tr><td>Descrição de Propósito
(Description & Purpose)</td><td colspan="3">Após uma versão ser aprovada na etapa de Release Test, a mesma pode ser considerada pronta para ser implantada em produção. Para isso, a equipe deve notificar o cliente por email informando data e hora da implantação, o nome ou número da versão, e o release notes da versão.
Utilizando uma janela de tempo que não comprometa o uso da aplicação, o gerente de configuração deve fazer backup de tudo o que for necessário caso precise retroceder para a versão antiga e só então trocar a versão do software no ambiente de produção ou homologação.
É sempre importante monitorar atentamente o funcionamento nos primeiros dias após a implantação para identificar falhas.</td></tr>
<tr><td>Abstração
(Abstraction)</td><td colspan="3">Serviço</td></tr>
<tr><td>Ferramentas e Pessoas
(Tools & People)</td><td colspan="3">- Gerente de configuração
- Versão do software
- Email
- Ambiente de produção</td></tr>
<tr><td>Estilo de Desenvolvimento
(Development Style)</td><td colspan="3">Iterativo</td></tr>
<tr><td>Ambiente Físico
(Physical Environment)</td><td colspan="3">Qualquer ambiente que der o acesso necessário ao ambiente de produção ou homologação</td></tr>
<tr><td>Pré e Pós Condições
(Pre and Post Conditions)</td><td colspan="3">Pre:
- Versão aprovada do sofware
Pos:
- Nova versão do software implantada no ambiente de produção ou homologação</td></tr>
<tr><td>Restrições e Riscos
(Constraints and Risks)</td><td colspan="3">É comum haver um grande risco associado às trocas de versão. Para reduzir os dados é extremamente recomendado o backup de todos os arquivos necessário para uma possível ação de rollback</td></tr>
<tr><td rowspan="6">Grau de Agilidade
(Degree of agility)</td><td>Atributo
(Attribute)</td><td>Valor
(Value)</td><td>Grau
(Degree)</td></tr>
<tr><td>1. Flexibilidade (Flexibility)</td><td>1</td><td rowspan="5">0.6</td></tr>
<tr><td>2. Velocidade (Speed)</td><td>1</td></tr>
<tr><td>3. Magreza (Leanness)</td><td>0</td></tr>
<tr><td>4. Aprendizado (Learning)</td><td>0</td></tr>
<tr><td>5. Responsividade (Responsiveness)</td><td>1</td></tr>
<tr><td>Valor de Negócio
(Business Value)</td><td colspan="3">- Funcionalidades prontas para uso
- Percepção real de evolução do projeto
- Satisfação do cliente</td></tr>
<tr><td>Alertas
(Alerts)</td><td colspan="3"></td></tr>
</table>

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)

Quadro 18 – Fragmento do Processo 12

<table>
<tr><td>ID e Nome
(ID & Name)</td><td colspan="2">12</td><td>Fechamento do Projeto</td></tr>
<tr><td>Descrição de Propósito
(Description & Purpose)</td><td colspan="3">Consiste na entrega final do projeto onde devem ser disponibilizados os artefatos gerados pelo projeto e confirmado os escopo realizado até ali</td></tr>
<tr><td>Abstração
(Abstraction)</td><td colspan="3">Serviço</td></tr>
<tr><td>Ferramentas e Pessoas
(Tools & People)</td><td colspan="3">- Gerente de Projetos
- Cliente
- E-mail (notificação de fim de projeto com descrição do escopo realizado)
- Código fonte
- Manual de uso do sistema</td></tr>
<tr><td>Estilo de Desenvolvimento
(Development Style)</td><td colspan="3">Iterativo</td></tr>
<tr><td>Ambiente Físico
(Physical Environment)</td><td colspan="3">Qualquer ambiente</td></tr>
<tr><td>Pré e Pós Condições
(Pre and Post Conditions)</td><td colspan="3">PRE: Estar próximo à data de finalização do projeto
POS: Todos os artefatos de fechamento entregues ao cliente</td></tr>
<tr><td>Restrições e Riscos
(Constraints and Risks)</td><td colspan="3">Risco: o planejamento inadequado pode gerar problemas de prazo para geração dos manuais ou problemas de qualidade do produto, caso testes e validação não sejam realizados antecipadamente</td></tr>
<tr><td rowspan="6">Grau de Agilidade
(Degree of agility)</td><td>Atributo
(Attribute)</td><td>Valor
(Value)</td><td>Grau
(Degree)</td></tr>
<tr><td>1. Flexibilidade (Flexibility)</td><td>0</td><td rowspan="5">0.4</td></tr>
<tr><td>2. Velocidade (Speed)</td><td>1</td></tr>
<tr><td>3. Magreza (Leanness)</td><td>0</td></tr>
<tr><td>4. Aprendizado (Learning)</td><td>1</td></tr>
<tr><td>5. Responsividade (Responsiveness)</td><td>0</td></tr>
<tr><td>Valor de Negócio
(Business Value)</td><td colspan="3">- Entrega final do produto
- Documentação final do produto
- Satisfação do cliente</td></tr>
<tr><td>Alertas
(Alerts)</td><td colspan="3">É preciso estar atento para que todos os artefatos de software, incluindo o manual, estejam prontos, testados e validados com certa antecedência</td></tr>
</table>

Fonte: Próprio autor. Adaptado de Qumer e Henderson-Sellers (2008)