AGILECRITPATH: IDENTIFICANDO TAREFAS CRÍTICAS EM AMBIENTES ÁGEIS

Rachel Vital Simões

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Toacy Cavalcante de Oliveira

Rio de Janeiro
Junho de 2019

AGILECRITPATH: IDENTIFICANDO TAREFAS CRÍTICAS EM AMBIENTES ÁGEIS

Rachel Vital Simões

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof.Toacy Cavalcante de Oliveira, D.Sc.

_____

Prof. Guilherme Horta Travassos, D.Sc.

_____

Prof. Eber Assis Schmitz, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2019

Simões, Rachel Vital

AGILECRITPATH: Identificando Tarefas Críticas em ambientes Ágeis / Rachel Vital Simões – Rio de Janeiro: UFRJ/COPPE, 2019.

XIII, 114 p.: il.; 29,7 cm.

Orientador: Toacy Cavalcante de Oliveira

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2019.

Referências Bibliográficas: p. 82-91.

1. Engenharia de Software 2. Métodos Ágeis 3. Método do Caminho Crítico. I. Oliveira, Toacy Cavalcante de II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

AGRADECIMENTOS

Primeiramente a Deus por te me dado saúde e força para superar as dificuldades.

Aos meus filhos Arthur e Theo por tudo que eles representam.

Aos meus familiares pelo apoio. Aos meus avós, Alberto e Margarida, que mesmo não estejando mais aqui presentes, eu sei o quanto estariam felizes com o meu feito.

Ao meu orientador, Toacy, por toda a atenção, paciência, empenho e pelos constantes direcionamentos. Pela confiança depositada no trabalho e pelas inúmeras reuniões, aulas, discusões, ensinamentos e ainda pelos conselhos sempre valiosos.

Aos professores da COPPE, que eu tive o prazer de conhecer e assistir às suas aulas: Guilherme, Ana Regina, Jano e Henrique. Ao Breno, Valéria e ao Professor Guilherme, novamente, pela parceria e incentivo na escrita do meu primeiro artigo publicado.

Aos membros da banca examinadora, professores Guilherme Travassos e Eber Schmitz pelo interesse e disponibilidade.

Aos meus colegas do grupo de pesquisa Prisma pelas constantes contribuições, pela companhia e pelos ensinamentos. À Glaucia, Ulisses, Renata, Alciléia, André e Edson pelas valiosas dicas e sempre dispostos a ajudar. Gratidão especial à Rebeca, sempre disposta a colaborar e propiciar o sucesso da pesquisa.

Agradeço, por fim, a OWSE empresa que eu trabalho, a qual sempre acreditou no meu empenho e assim me apoiou nesse desafio. Agradeço também ao Walter Magiolo que foi um grande incentivador.

A todos, minha eterna gratidão.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

AGILECRITPATH – IDENTIFICANDO TAREFAS CRÍTICAS EM AMBIENTES ÁGEIS

Rachel Vital Simões

Junho/2019

Orientador: Toacy Cavalcante de Oliveira

Programa: Engenharia de Sistemas e Computação

Planejar e monitorar a execução de atividades de desenvolvimento de software em ambientes ágeis não é um procedimento trivial. Uma das principais falhas do planejamento ágil é não considerar as dependências existentes entre as tarefas do projeto. Dependências entre tarefas encontradas em planos de projeto de desenvolvimento de software podem levar ao surgimento de caminhos críticos, onde as tarefas precisam ser tratadas em uma sequência rigorosa, porque a conclusão de algumas tarefas depende da conclusão de outras. Não gerenciar essas dependências pode reduzir o desempenho da equipe e atrasar a entrega do produto. Está dissertação está dividida em quatro partes principais: (1) mapeamento sistemático sobre dependências de tarefas em ambientes ágeis, (2) um estudo exploratório realizado na indústria, (3) o desenvolvimento de uma ferramenta*, AgileCritPath*, como uma forma de apoiar as equipes de desenvolvimento na identificação de tarefas críticas do projeto; e (4) uma avaliação da ferramenta *AgileCritPath* baseada no Modelo de Aceitação de Tecnologia (TAM). Através do estudo *in vivo* consolidamos os resultados observados no uso da ferramenta em uma empresa de desenvolvimento de software onde avaliamos os benefícios da identificação das dependências entre tarefas em um ambiente ágil.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AGILECRITPATH – IDENTIFYING CRITICAL TASKS IN AGILE ENVIRONMENTS

Rachel Vital Simões

June/2019

Advisor: Toacy Cavalcante de Oliveira

Department: Systems and Computer Engineering

Planning and monitoring the execution of software development activities in agile environments are not trivial procedures. One of the main flaws of agile planning is not considering the dependencies that exist between project tasks. Dependencies between tasks found in software development project plans may lead to the emergence of critical paths, where tasks need to be handled in a strict sequence because the completion of some tasks depends on the completion of others. Not managing such critical paths may reduce team performance and delay product delivery. The dissertation is divided into four main parts: (1) study of Systematic Mapping about task dependencies in agile environments, (2) an exploratory study conducted in industry, (3) development of a tool, AgileCritPath, as a way to support teams to identify critical project tasks; and (4) an evaluation of the AgileCritPath tool based on the Technology Acceptance Model (TAM). In the study of Systematic Mapping (1) we tried to identify in the literature works related to the management and identification of dependencies in agile environments. Through the exploratory study (2) we have identified empirical evidences that there is a need to identify and control dependencies between tasks in the development of software in agile environments. (3) The use of the AgileCritPath tool allowed us to introduce the concepts of the Critical Path Method into a software development organization. Finally, (4) through the in vivo study we consolidate the observed results of the use of the tool in a company where we evaluate the benefits of identifying the dependencies between tasks in an agile environment.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE QUADROS

LISTA DE SIGLAS

ASD	*ADAPTIVE SOFTWARE DEVELOPMENT*
BPM	*BUSINESS PROCESS MANAGEMENT*
BPMN	BUSINESS PROCESS MODEL AND NOTATION
CPM	*CRITICAL PATH METHOD*
CMMI	*CAPABILITY MATURITY MODEL INTEGRATION*
CMMN	CASE MANAGEMENT MODEL AND NOTATION
DAD	*DISCIPLINED AGILE DELIVERY*
DSDM	*DYNAMIC SYSTEM DEVELOPMENT METHOD*
EAP	*ESTRUTURA ANALÍTICA DE PROJETO*
EMS	*ESTUDO DE* MAPEAMENTO *SISTEMÁTICO*
EVM	*EARNED VALUE MANAGEMENT*
FDD	*FEATURE DRIVEN DEVELOPMENT*
GP	*GESTÃO DE PROJETOS*
GQM	*GOAL QUESTION METRIC*
KIP	*KNOWLEDGE-INTENSIVE* PROCESSES
LESS	*LARGE SCALE SCRUM*
MPS-BR	*MELHORIA DO PROCESSO DE SOFTWARE BRASILEIRO*
MSF	*MICROSOFT SOLUTIONS FRAMEWORK*
PIC	*PROCESSOS INTENSIVOS EM CONHECIMENTO*
PERT	*PROGRAM EVALUATION AND REVIEW TECHNIQUE*

SAFe	SCALED AGILE *FRAMEWORK*
SMS	*SYSTEMATIC MAPPING STUDY*
TAM	*TECHNOLOGY ACCEPTANCE MODEL*
TDD	*TEST DRIVEN DEVELOPMENT*
XP	*EXTREME PROGRAMMING*

1. Introdução

Este capítulo apresenta as principais questões motivadoras para a realização desta dissertação, assim como o problema, seus objetivos, a metodologia utilizada e a estrutura como esse texto encontra-se organizado.

1.1. Contexto

Desde meados da década de 90, os métodos ágeis têm se destacado na área de engenharia de software. Através de metodologias ágeis, as empresas têm conseguido atingir seus objetivos e responder às mudanças de forma rápida (Suomalainen et al., 2015) (Torrecilla-Salinas et al., 2015).

Grandes organizações de desenvolvimento de software estão iniciando e adaptando seus processos internos para adoção de práticas ágeis. Porém, ainda existem muitos desafios na área para que essas práticas possam atender às necessidades das empresas, principalmente, quando se trata de gerenciar e acompanhar as demandas. (Heikkilä et al., 2013)

Como resposta à crescente pressão para melhorar a gestão, acompanhar as constantes mudanças e reduzir o tempo de entrega no mercado, organizações de software adotaram diferentes práticas de desenvolvimento ágil para promover entregas de software mais rápidas e confiáveis.

1.2. Motivação

O aumento da adoção das práticas ágeis faz com que os meios tradicionais de gestão de projetos sejam redefinidos. De acordo com a maior pesquisa recorrente sobre adoção ágil, o *State of Agile Survey* (VersionOne, 2016), 43% dos entrevistados trabalhavam em organizações de desenvolvimento com mais de 50% de equipes usando métodos ágieis, e 62% de quase 4.000 entrevistados vieram de uma organização com mais de cem pessoas. Embora essa pesquisa não seja científica, indica um número significativo de organizações usando métodos ágieis.

Muitos métodos ágeis preveem certo nível de gestão para as tarefas, porém, das principais técnicas aplicadas, (VersionOne, 2018) nenhuma delas garante a solidez do planejamento que a equipe está seguindo durante a iteração. Na maioria das vezes, um erro de planejamento só fica evidente no final do prazo estabelecido para a entrega.

Cohn em seu livro "*Agile Estimating and Planning*", aborda as principais falhas do planejamento ágil. Dentre elas, o autor cita que tarefas de desenvolvimento de software são dependentes. (Cohn, 2005) É um erro dos times ágeis planejar as tarefas da iteração como se elas fossem independentes.

Segundo pesquisa *Agile Report* 2018 (VersionOne, 2018) 69% das organizações que adotam métodos ágeis buscam iterações curtas, o que torna esse problema ainda mais evidente. As organizações devem estar preparadas para conseguir planejar as entregas constantes e que agreguem valor ao négocio.

Bick et al. (2018) identificaram em seu estudo que a falta de conscientização sobre as dependências que existem nas tarefas do processo de desenvolvimento de software é uma explicação chave para a coordenação ineficaz da equipe. A falta de conhecimento das dependências entre as tarefas gera um plano de atividades desalinhado e que pode não ser possível ser executado pela equipe.

Tendo em vista o argumentado, é possível definir como motivação para a realização desta dissertação os pontos a seguir:

- Mesmo existindo processos e técnicas para planejamento da iteração em métodos ágeis, estes não garantem a solidez do plano;
- Desconsiderar as dependências entre as tarefas faz com que percamos a noção de priorização e ainda inviabiliza a avaliação dos impactos das tarefas com dependências;
- Fazer o detalhamento das tarefas na fase de planejamento e não levar em consideração aspectos como a dependência entre as tarefas pode mascarar erros no planejamento;
- O método do Caminho Crítico funciona em modelos tradicionais de gestão de projetos como um método eficaz para acompanhar o andamento

das tarefas e apoia na identificação dos atrasos que podem comprometer a data de entrega do produto.

1.3. Problema

Metodologias ágeis como *Scrum*, *XP*, *FDD*, *TDD*, *ASD*, *MSF*, *DSDM* possuem atividades específicas voltadas para o planejamento das iterações. Metodologias propostas para escalar o uso das metodologias ágeis como *SAFe*, *Nexus*, *Scrum@Scaler*, *LeSS* e *DAD* reforçam ainda mais a importância do planejamento das iterações.

Através de estudos realizados na literatura e um estudo exploratório realizado em uma empresa de desenvolvimento de software brasileira, conseguimos identificar que:

- Um motivo para a falha no planejamento de projetos em ambientes ágeis é não considerar as dependências que existem entre as tarefas (Cohn, 2005). Durante a observação de dados reais de tarefas executadas em uma iteração foi possível perceber que existem dependências entre as tarefas e durante o planejamento essas dependências não são consideradas pela equipe;
- A gestão das dependências é uma questão fundamental no desenvolvimento do software, pois através dela é possível definir o fluxo de execução entre as atividades e direcionar melhor o esforço da equipe (Bick et al. 2018) (Aslam and Ijaz, 2018). Somando-se a isso, as dependências entre as tarefas diminuem o nível de agilidade da equipe (Lomas et al., 2006). Através das análises dos dados, nós vimos que a existência de algumas dependências não gerenciadas prejudicava o desempenho da equipe e aumentava o risco de atraso da entrega.
- Gerenciar as dependências entre as tarefas não é algo trivial (Strode, 2016), mesmo quando a organização tem um processo de desenvolvimento de software extremamente simples. Quando analisamos o log de execução das tarefas, vimos que existia uma complexidade grande nos planos das iterações que consideravam as dependências entre as tarefas. Além disso, um ambiente com muitas incertezas e mudanças pode demandar um esforço maior para manter a gestão das dependências durante a fase de execução.
- Como as dependências entre as tarefas não eram consideradas pela equipe, o atraso das tarefas críticas não ficava perceptível, logo era difícil avaliar o

impacto de alguns atrasos. Estudos de Engenharia de Requisitos ágeis (Milicic et al., 2014), (Patton and Economy, 2014) reforçam que maiores ganhos são obtidos quando as dependências são descobertas ou detectadas o mais cedo possível em um projeto. Quanto mais cedo as dependências forem identificadas, mais fácil torna-se avaliar possíveis riscos de atrasos;

Com os problemas identificados, supõe-se que:

Identificar as dependências que existem entre as tarefas pode trazer benefícios ao planejamento e execução do projeto. Através da aplicação de práticas baseadas no Método do Caminho Crítico em ambientes ágeis podemos auxiliar a equipe no planejamento das entregas e durante a execução da iteração. A identificação das tarefas críticas permite que a equipe monitore as tarefas que devem ser finalizadas e os prazos necessários para execução das tarefas considerando as dependências existentes.

1.4. Objetivos

Alinhado aos problemas apresentados anteriormente, o objetivo geral desta dissertação é **identificar as dependências entre as tarefas, propor o uso de conceitos baseados no Método do Caminho Crítico para identificar tarefas críticas em ambientes ágeis**.

Para definir o objetivo para este estudo foi utilizada a abordagem GQM (*Goal, Question, Metric*) (Solingen et al., 2002). De acordo com a abordagem GQM, o principal objetivo deste trabalho é:

Analisar os projetos de desenvolvimento de software em ambientes ágies
Com o propósito de compreender
Com respeito às tarefas e suas dependências
Do ponto de vista dos profissionais da prática
No contexto do uso das dependências para encontrar as tarefas críticas

Este objetivo geral pode ser decomposto nos seguintes objetivos específicos:

I. **Obter o estado da arte das abordagens existentes.** Este objetivo busca identificar trabalhos que visam a melhorar a forma de priorização das tarefas de desenvolvimento de software em ambientes ágeis e obter uma visão do estado da arte sobre as estratégias já definidas na área. Para alcançar esse objeto foi realizada uma investigação inicial das estratégias através de uma Revisão da Literatura *Ad hoc*.

II. **Identificar as dependências entre tarefas que impactam o desenvolvimento do software em ambientes ágeis.** A dependência entre as tarefas é um ponto importante que pode bloquear o fluxo de execução de tarefas. Para alcançar esse objetivo, incluímos um Mapeamento Sistemático para identificar no estado da arte, quais dependências podem existir em projetos ágeis.

III. **Aplicar conceitos do Método do Caminho Crítico como estratégia para identificação das tarefas críticas em ambientes ágeis.** Para alcançar esse objetivo realizamos um estudo exploratório em uma empresa de desenvolvimento de software para avaliar o uso do Método do Caminho Crítico em tarefas de desenvolvimento de software de um projeto real;

IV. **Propor uma estratégia (e uma ferramenta) que permita à equipe de desenvolvimento obter informações sobre as tarefas críticas.** Nesse objetivo desenvolvemos uma ferramenta para apoiar a equipe de desenvolvimento de software na identificação das tarefas que estão no caminho crítico;

V. **Avaliar a estratégia proposta.** Para conseguirmos avaliar a estratégia, realizamos um estudo *in vivo* com objetivo de analisar a eficiência e a adequação da técnica através do uso da ferramenta em um ambiente industrial. A avaliação da ferramenta ocorreu através do Modelo de Aceitação de Tecnologia (TAM).

1.5. Metodologia de Pesquisa

Esta seção apresenta as atividades planejadas para alcançar o objetivo desta pesquisa. A metodologia adotada está apresentada na Figura 1.1, juntamente com o cronograma e a evolução dos estudos. Cada etapa da metodologia será discutida em maiores detalhes nos próximos capítulos dessa dissertação.

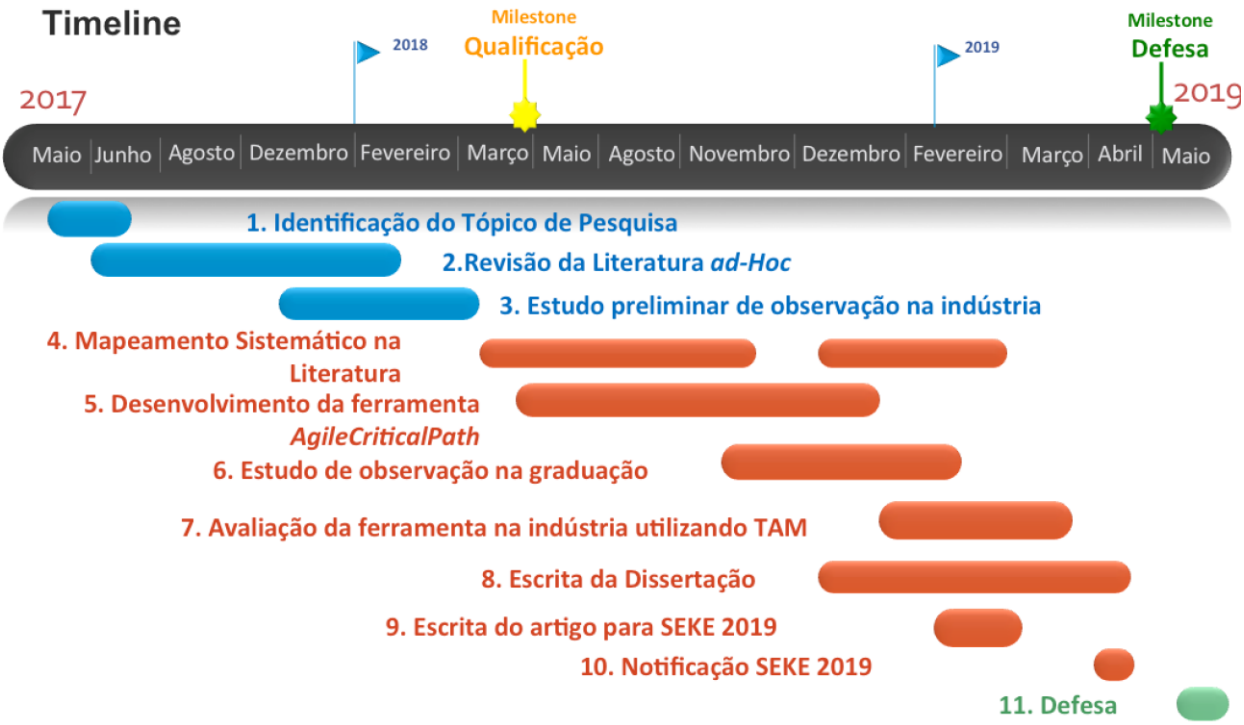


Figura 1.1: Metodologia de Pesquisa

1. **Identificação do Tópico de Pesquisa:** O tema de pesquisa foi identificado através do relato de experiência de uma organização que utiliza métodos ágeis há mais de 10 anos. O problema foi discutido com o orientador em diferentes situações e contextos trazendo o tema para um viés acadêmico. As discusões com o grupo de pesquisa Prisma também ajudaram na fundamentação das ideias de que o tópico era apropriado para ser mais desenvolvido como parte de uma dissertação de mestrado.
2. **Revisão da Literatura *Ad Hoc*:** após a seleção do tópico, foi realizada uma revisão da literatura *ad hoc* para obter mais informações sobre o problema de gestão de tarefas em ambientes ágeis, constatar a existência do problema em outros contextos e obter o estado da arte das soluções já existentes. Durante a revisão da literatura, observamos

que existe um número considerável de estudos realizados relacionados ao tema principalmente nos últimos anos. Observamos também indícios que a comunidade científica considera o assunto apropriado para estudos mais detalhados.

3. **Estudo exploratório na indústria:** Após constatarmos os resultados obtidos na revisão da literatura, buscamos aprofundar nossas ideias analisando dados de projetos de uma organização de desenvolvimento de software. Durante a análise desse estudo, observamos que existem dependências entre as tarefas e essas dependências influenciam no processo de execução das atividades do projeto. Os métodos ágeis não fornecem meios que facilitem a gestão das dependências que existem entre as tarefas. Esse estudo exploratório foi realizado seguindo as diretrizes discutidas em detalhe no Capítulo 4 dessa dissertação.
4. **Mapeamento Sistemático da Literatura:** Somando-se o resultado da revisão da Literatura e o estudo preliminar realizado através da observação na indústria, identificamos que poderíamos aplicar conceitos do Método do Caminho Crítico em ambientes ágeis, porém seria pertinente conhecer quais as dependências que podem influenciar a execução das tarefas considerando o contexto de ambientes ágeis. O Mapeamento Sistemático da Literatura foi capaz de nos dar evidências dos tipos de dependências que podem existir em projetos de desenvolvimento de software que utilizam métodos ágeis. Com esses resultados, conseguimos avaliar quais tipos de dependências poderíamos tratar nessa dissertação.
5. **Desenvolvimento da ferramenta *AgileCritPath*:** Com a realização dos estudos anteriores, observamos que o uso do Método do Caminho Crítico é viável mesmo em ambientes ágeis. Durante o estudo de observação na indústria (item 3) conseguimos identificar com a ajuda de um especialista uma quantidade relativamente grande de dependências. Vimos que gerenciar essas dependências para obter os caminhos críticos sem nenhuma automação iria demandar um esforço maior da equipe. O desenvolvimento da ferramenta também seria um ganho para o desenvolvimento dessa pesquisa, pois poderíamos validar os conceitos em diferentes ambientes e projetos. No Capítulo 5 foi detalhado o processo de desenvolvimento da ferramenta.

6. **Estudo de Observação com os alunos da graduação:** Com o desenvolvimento das versões iniciais da aplicação *AgileCritPath*, identificamos a oportunidade de avaliação e uso da ferramenta com os alunos da disciplina de Qualidade de Software. Durante a disciplina, os alunos desenvolveram um projeto no *GitHub*. Eles utilizaram um processo de desenvolvimento de software baseado no Scrum, e as tarefas eram registradas no *GitHub* através da ferramenta ZenHub. Para avaliar o uso da proposta nesse cenário, desenvolvemos uma versão da aplicação *AgileCritPath* que consulta as tarefas do *Github*. No Capítulo 6, foi detalhado o estudo Piloto onde disponibilizamos o uso da ferramenta durante o desenvolvimento do projeto dos alunos da graduação do curso de Ciência da Computação.
7. **Avaliação da ferramenta na indústria utilizando TAM:** Com o objetivo de analisar a eficiência e adequação do uso do Método do Caminho Crítico em um ambiente ágil utilizamos o Modelo de Aceitação de Tecnologia (TAM) para avaliar a ferramenta na indústria. O estudo foi conduzido utilizando o protocolo formal, de acordo com as diretrizes obtidas na literatura e estão discutidas em detalhe no Capítulo 6 deste trabalho.
8. **Escrita da Dissertação:** Os estudos realizados nos passos anteriores foram documentados e organizados em formato de dissertação para facilitar futuras referencias.
9. **Escrita do artigo para SEKE 2019:** Com o resultado da pesquisa, escrevemos um artigo submetido para o SEKE 2019 - *The 31st International Conference on Software Engineering and Knowledge Engineering*.
10. **Notificação SEKE 2019:** Artigo aceito como *Full Paper*.

1.6. Organização do Texto

Este trabalho está organizado em 7 capítulos e 5 apêndices. O presente **Capítulo 1 - Introdução** apresenta motivação para desenvolvimento deste trabalho, o problema, os objetivos da pesquisa e a metodologia.

Capítulo 2 – Fundamentação Teórica apresenta a Fundamentação Teórica, com a descrição dos conceitos sobre Métodos Ágeis, Gestão de Projetos, Método do Caminho Crítico, dependências e seus impactos na execução das tarefas de desenvolvimento de software.

O Capítulo 3 – Trabalhos Relacionados apresenta a revisão da literatura realizada para identificar dependências existentes em projetos de desenvolvimento de software que utilizam métodos ágeis. Neste capítulo serão apresentados os trabalhos relacionados identificados na literatura.

O **Capítulo 4 – Estudo Exploratório** apresenta um estudo preliminar realizado na indústria para fundamentar as ideias e conceitos da dissertação. Através de dados reais de uma empresa de densenvolvimento de software aplicamos conceitos do Método do Caminho Crítico para avaliar a viabilidade do uso da técnica em um ambiente ágil.

No **Capítulo 5 – A ferramenta *AgileCritPath*** apresenta a aplicação *AgileCritPath* desenvolvida durante a pesquisa. Neste capítulo detalhamos as etapas do desenvolvimento, a arquitetura e como a ferramenta funciona. Hoje a aplicação se integra com o *GitHub* e com o sistema de gestão de tarefas *Redmine*.

O **Capítulo 6 – Validação da Ferramenta** apresenta o processo realizado para validação da ferramenta utilizando o modelo de aceitação de tecnologia TAM.

Por fim, no **Capítulo 7 - Conclusão** apresenta as contribuições, limitações, trabalhos futuros e considerações finais sobre a dissertação.

No **Apêndice A – Formulário de Exclusão** apresenta os trabalhos excluídos na revisão da literatura realizada através de um Estudo de Mapeamento Sistemático.

No **Apêndice B – Formulários de identificação das dependências entre tarefas por iteração** apresenta as dependências identificadas durante o estudo preliminar de observação na indústria. O estudo de observação foi realizado através da análise dos dados de três iterações onde identificamos as dependências que existiam entre as tarefas. Nesta seção tem três formulários, sendo um formulário para cada iteração analisada.

Na **Apêndice C – Rede de tarefas executadas na iteração 2.14** estamos mostrando através de fluxos, as dependências identificadas na iteração 2.14. Essa análise faz parte dos resultados do estudo de exploratório.

Na **Apêndice D – TAM - Formulário de Consentimento e Desimpedimento de Participação** apresenta o formulário de consentimento de participação no estudo de validação da ferramenta.

No **Apêndice E – TAM - Caracterização do Participante** apresenta o formulário utilizado para caracterizar os participantes do estudo de validação da ferramenta.

2. Fundamentação Teórica

Este capítulo apresenta os fundamentos e conceitos envolvidos nessa dissertação. O capítulo inicia-se com a introdução e posteriormente com a explicação dos conceitos os quais são detalhados em tópicos.

2.1. Introdução

A proposta central desta dissertaç é utilizar conceitos do Método do Caminho Crítico em ambientes organizacionais que utilizam métodos ágeis. Para que a proposta seja viável é fundamental um esforço para identificar os tipos de dependências que existem entre as tarefas. Identificar as dependências entre as tarefas não é um procedimento trivial e nem usual em organizações que utilizam métodos ágeis, porém, em nossos estudos, vimos que através da identificação das dependências entre as tarefas podemos gerenciar as tarefas críticas e diminuir os riscos de atrasos nas entregas dos projetos.

Nas próximas seções, iremos apresentar conceitos relacionados à proposta deste trabalho, dando ao leitor uma visão geral sobre Métodos Ágeis, Gestão de Projetos, Método do Caminho Crítico, as Classificações das Dependências e sua importância na definição de um plano de trabalho.

2.2. Métodos Ágeis

Os métodos ágeis oferecem atrativos para as empresas de software, pois prometem um menor tempo de entrega do produto no mercado, além de maior flexibilidade para acomodar as mudanças nos requisitos e assim, aumentar a capacidade de reagir às necessidades dos clientes (Williams and Cockburn, 2003).

Desde meados da década de 90, os métodos ágeis têm se popularizado na área de engenharia de software, com isso, as empresas veem conseguido atingir seus objetivos e responder às mudanças de forma rápida.

Os métodos ágeis não significam simplesmente desenvolvimento rápido de software, mas sim a capacidade de adaptação e flexibilidade às mudanças nos processos, nos produtos e no ambiente. Para Boehm and Turner (2004), os métodos ágeis de desenvolvimento de software promovem uma melhora no nível de satisfação dos clientes, uma diminuição nas taxas de defeitos, períodos de desenvolvimento mais curtos e facilita a adequação às mudanças.

O desenvolvimento de software ágil derrubou muitas diretrizes aceitas para gerenciamento de projetos de TI e engenharia de software (Boehm and Turner, 2004), porém problemas reconhecidos ao adotar o desenvolvimento ágil de software em certas culturas organizacionais ainda não foram resolvidos. (Livrari and Livrari, 2011)

Uma notável diferença entre os métodos ágeis e o modelo tradicional de desenvolvimento de software é a maneira como as atividades do projeto são organizadas. A adoção de iterações curtas, reuniões diárias, programação em pares, entregas frequentes de produtos, testes automatizados e integração contínua, reuniões de planejamento, lições atendidas, o uso do quadro *Kanban* (Strode, 2016) são muitas mudanças que podem impactar a forma de gestão das tarefas em ambiente ágeis. Os principais objetivos dessas mudanças foram, por um lado, permitir que as organizações se adaptem rapidamente às necessidades mutáveis dos clientes (Pikkarainen et al., 2008) e, por outro lado, entregar resultados valiosos ao cliente o mais rápido possível.

Metodologias ágeis estão cada vez mais populares devido ao ambiente instável onde as organizações trabalham. De acordo com (Lomas et al., 2006), a abordagem ágil permite mais mudanças, maior comunicação e colaboração entre os participantes, entregas frequentes e maior interação com as partes interessadas. Em cenários onde companhias de desenvolvimento de software enfrentam situações como mudanças constantes nas funcionalidades, time-to-market reduzido e necessidade de entrega contínua de versões ao cliente, a adoção das práticas ágeis têm se tornado uma tendência.

2.4. Gestão Ágil de Projetos

Atualmente existe uma inevitável necessidade de novas abordagens de gerenciamento de projetos em ambientes com constantes mudanças, inovação contínua e que requerem redução de custos (Conforto and Amaral, 2010) (Špundak, 2014). O gerenciamento de projetos ágeis tem se mostrado cada vez mais evidente e tem causado uma revolução silenciosa na área que há algum tempo não sofria grandes mudanças.

Os processos de gestão de projetos tradicionais utilizados nas organizações são encarados como modelos maduros onde os processos estão bem definidos e atendem a quase todas as áreas. Apesar da maturidade e robustez dos métodos tradicionais de gestão de projetos, um número crescente de autores acredita que aplicar essas técnicas de maneira uniforme aos projetos pode ser uma desvantagem (Kaliprasad, 2005) (Rose, 2010) (Špundak, 2014). Os métodos tradicionais empregados no planejamento e na gestão de projetos não se adequam plenamente em projetos complexos e quando são executados em um ambiente dinâmico.

Abordagens tradicionais de gestão de projetos nos remetem à ideia de que projetos são relativamente simples, previsíveis e lineares, com limites claramente definidos. Esse cenário faz com que seja fácil planejar e conseguir seguir um plano sem muitas mudanças (Boehm, 2002) (Kaliprasad, 2005) (Saynisch, 2010) (Špundak, 2014). Abordagens tradicionais deixam de lado aspectos cruciais quanto às entregas constantes, à redução d*o time-to-market* e quando existem muitas incertezas no processo.

Contudo, com a popularização do uso de metodologias ágeis, os modelos de gestão tradicional de projetos já não atendem tão plenamente, por isso iniciou-se o movimento para estudos do gerenciamento ágil de projetos. Em 2011, o termo “Gerenciamento Ágil de Projetos” ultrapassou o termo “Desenvolvimento Ágil de Software” no Google Trends (Stettina and Hörz, 2015).

A agilidade está na capacidade de criar e de responder às mudanças, a fim de agregar valor ao ambiente de negócios turbulento (Highsmith, 2003). As principais características de ambientes ágeis estão na possibilidade de oferecer entregas constantes, resultados mais confiáveis e estar aderente a um ambiente que favorece inovação contínua. Em ambientes ágeis, ter abordagens que ofereçam a capacidade

de adaptação durante o ciclo de vida dos projetos é mais importantes do que ter a previsibilidade oferecida pela gestão de projetos tradicionais.

No ano de 2015 o CHAOS Report (Figura 2.1) estudou 50.000 projetos em todo o mundo. Esses projetos vão desde pequenas melhorias até o desenvolvimento de grandes sistemas. Como pode ser observado, o percentual de projetos de sucesso em ambientes ágeis é significativamente maior que os projetos de abordagens tradicionais.

Tabela 2.1: CHAOS Re Report 2015 . Retirado de: https://www.infoq.com/articles/standish-chaos-2015

Tamanho	Método	Bem Sucedido	Desafiado	Falhou
Todos os tamanhos	Ágil	39%	52%	9%
	Cascata	11%	60%	29%

Até mesmo grandes organizações de desenvolvimento de software já estão iniciando e adaptando seus processos internos para adoção de metodologias ágeis. Porém ainda existem muitos desafios na área para que a adoção dessas práticas possa atender às necessidades das empresas, principalmente, quando se trata de gerenciar e acompanhar as demandas. (Heikkilä et al., 2013)

2.5. Visualizando das tarefas no quadro *Kanban*

Em métodos ágeis, o trabalho técnico da equipe de desenvolvimento é definido por meio de tarefas. As tarefas são estimadas pela equipe e geralmente representam uma pequena parte do trabalho esperado.

As tarefas podem ser visualizadas usando o quadro *Kanban* apresentado na Figura 2.1. O quadro *Kanban* representa visualmente o trabalho em vários estágios de um processo usando cartões para representar tarefas e colunas para representar cada estágio do processo. Os cartões são movidos da esquerda para a direita a fim de mostrar o progresso e ajudar a coordenar as equipes que executam o trabalho.

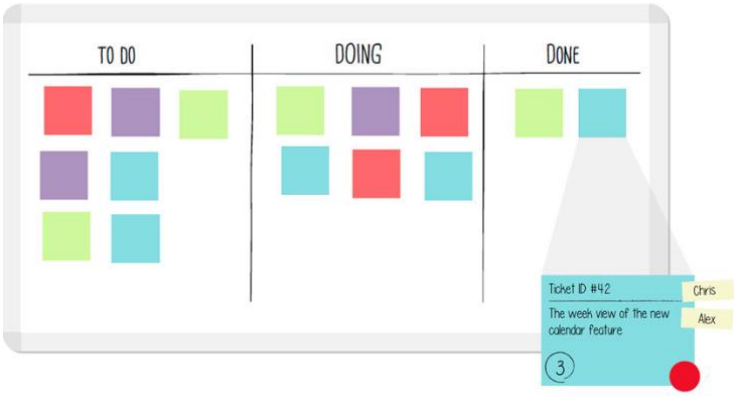


Figura 2.1: Quadro *Kanban*

O principal objetivo do *Kanban* não é sobrecarregar os membros da equipe, uma vez que é recomendado que uma atividade seja iniciada somente quando algum membro da equipe esteja disponível para executá-la com dedicação (Hazır, 2015).

O fato de ser possível usar o *Kanban*, sem necessariamente implementar conceitos como *timebox*, torna o método mais simples (Corona and Pani, 2013), porém, o quadro *Kanban* também pode ser usado em conjunto com metodologias como o *Scrum*, o qual torna a prática bastante utilizada em ambientes ágeis.

Embora o *Kanban* seja amplamente utilizado nas organizações, ele não mostra a visão das dependências. No quadro, não é possível ver quais tarefas podem bloquear a execução de outras. O fluxo de execução de tarefas não fica claro para a equipe.

Alaidaros (2018) identifica quais desafios e critérios que afetam o monitoramento do progresso do método Agile *Kanban* usando um método de revisão narrativa. Este estudo revelou que no método Agile *Kanban* falta mecanismo para rastreamento do progresso. Assim, ele precisa ser integrado a outros métodos, pois não possui uma definição padrão para o desenvolvimento de software e suas práticas específicas ainda não estão rigorosamente definidas. (Alaidaros et al., 2018)

O monitoramento do progresso é uma atividade importante no gerenciamento, pois, garante que a execução do projeto seja realizada de acordo com as expectativas de orçamento, cronograma e qualidade (Despa, 2014), (Hazır, 2015). A implementação de mecanismos de monitoramento de progresso em um ambiente ágil é fundamental para contribuir para o sucesso do projeto.

2.6. Método do Caminho Crítico

O Método do Caminho Crítico ou a Análise do Caminho Crítico foi originalmente desenvolvido pelas empresas *DuPont* e *Remington* em 1950 e desde então, tem sido empregado em várias organizações de diferentes tamanhos e ramos de atividades. O Método do Caminho Crítico surgiu, inicialmente, para gerenciar projetos mais extensos e complexos. Porém, a técnica pode ser utilizada para gerenciar qualquer tipo de projeto e inclusive, pode ser utilizada em tarefas de linha de produção.

Na área de gestão de projetos, o método é considerado como uma opção relevante para melhorar a visibilidade e para validar a viabilidade de entrega dos projetos em um determinado prazo.

O método é utilizado em conjunto com o diagrama de redes PERT (*Program Evaluation and Review Technique*), organizado em tarefas conforme as suas dependências. O método identifica a sequência de atividades na qual, caso uma atrase, todo o projeto estará atrasado, em outras palavras, a sequência das atividades que não tem folga. Segundo o Project Management Institute — PMBOK® 5ª edição, página 176 — o caminho crítico consiste na sequência de atividades que representa o caminho mais longo de um projeto. Em resumo, é a menor duração possível para que o projeto seja finalizado completando todas as suas atividades. O caminho crítico do projeto é o maior caminho do diagrama de rede e determina o prazo mais curto para a conclusão das atividades. Desta forma, o caminho crítico aponta quais atividades os responsáveis devem ter maior atenção.

Um projeto pode ter mais de um caminho crítico. O método do caminho crítico não leva em consideração a disponibilidade de recursos.

O Método do Caminho Crítico ajuda a direcionar ações de gerenciamento de tempo em um projeto. Quando se trata de reduzir o cronograma é bem menos custoso agir nas atividades que geraram efetivamente atraso no projeto do que reduzir o tempo de todas as atividades do cronograma. Um atraso em uma tarefa nem sempre vai atrasar o término do projeto, mas um atraso em uma tarefa do caminho crítico vai representar um atraso no término do projeto.

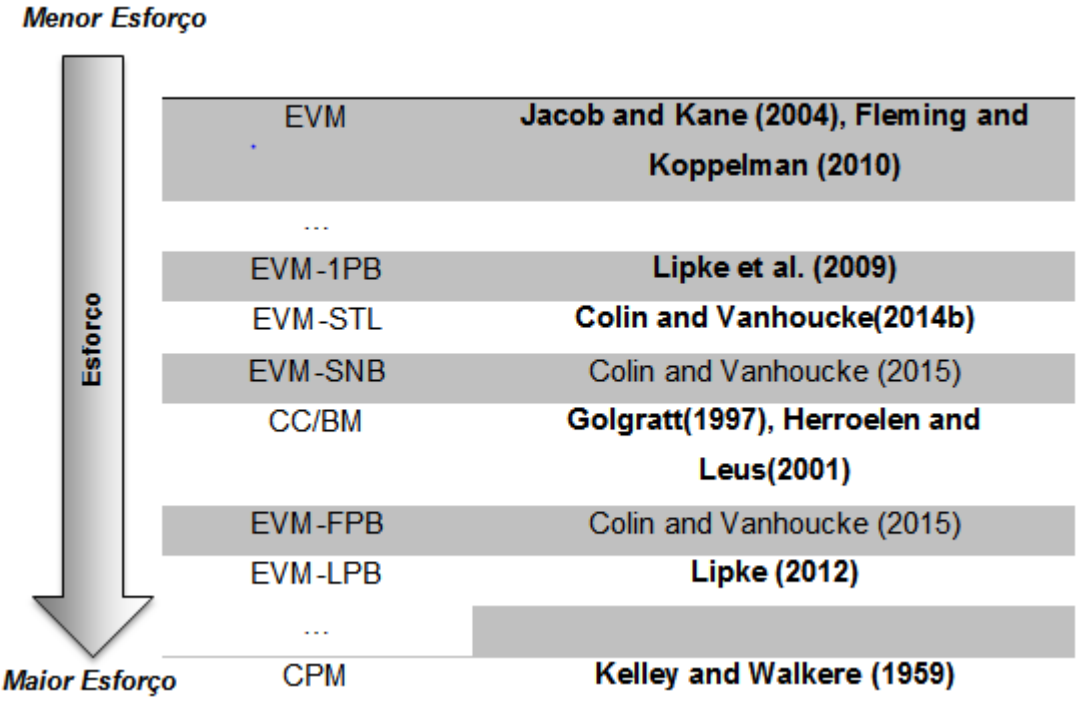


Figura 2.2: Classificação dos procedimentos de controle de projeto (Colin and Vanhoucke, 2015)

A pesquisa realizada por Colin and Vanhoucke (2015) mostra a classificação dos procedimentos de controle de projeto de acordo com o esforço que deve ser investido para mantê-los.

No topo da Figura. 2.2, o gerenciamento do valor agregado (EVM) é classificado como processo de controle que exige menos esforço do gerente dos projetos, enquanto CPM (*Critical Path Method*) foi classificado como uma técnica de baixo nível que requer um controle detalhado das atividades exigindo um maior nível de esforço.

Técnicas como EVM fornece uma visão sobre o desempenho do projeto,

calculando métricas nos níveis superiores da EAP (Colin and Vanhoucke, 2015), enquanto CPM pode ser usado na prática para formar uma base para a previsão da duração total do projeto e possibilita verificar o progresso durante a execução do projeto (Kelley Jr and Walker, 1959).

O Método do Caminho Crítico quando bem aplicado oferece benefícios como: visão da rede de tarefas, identificação de folgas, gerenciamento de *buffers*, auxilio no controle das execuções das tarefas para iniciar o mais cedo ou o mais tarde possível sem impactar a data final da entrega.

Uma descrição detalhada do Método do Caminho Crítico pode ser encontrada em (Ahuja et al., 1994) e (Stevenson et al., 2007).

2.7. Dependências

Dependências são definidas por Crowston e Osborn (1998) sendo uma ação que ocorre quando o progresso de uma atividade depende da produção em tempo hábil de uma ação anterior ou presença de um artefato, uma pessoa ou uma informação específica. (Crowston and Osborn, 1998)

O tratamento adequado das dependências que existem em um processo de desenvolvimento de software quando estamos em um ambiente ágil é um tópico já explorado na literatura, porém, atualmente, estudos empíricos sobre o tema ainda são escassos.

Durante a pesquisa, identificamos trabalhos que tratam as dependências em projetos de desenvolvimento de software em diversos aspectos conforme Tabela 2.3.

Tabela 2.3: Estudos sobre Dependências no processo de desenvolvimento de software.

Estudos sobre Dependências em Desenvolvimento de Software	**Publicações**
Dependências no contexto de desenvolvimento software (Definição e/ou Classificação)	(Malone et al., 1999); (Crowston and Osborn, 1998); (Wagstrom and Herbsleb, 2006); (Grinter, 1996); (Strode, 2016)
Dependências em ambientes distribuídos de desenvolvimento de software	(Espinosa et al., 2007), (Aslam and Ijaz, 2018); (Kerzner and Kerzner, 2017)
Dependências que interferem na	(Kraut and Streeter, 1995); (Staudenmayer,

coordenação/agilidade em desenvolvimento de Software	1997); (Grinter, 1996); (Wageman, 1995) (Espinosa et al., 2007); (Marks et al., 2001); (Hoegl et al., 2004); (Amrit and Van Hillegersberg, 2008); (Faraj and Sproull, 2000); (McChesney and Gallagher, 2004); (Kraut and Streeter, 1995); (Nidumolu, 1995); (Malone et al., 1999); (Lomas et al., 2006)

Dependências que existem entre as tarefas durante o desenvolvimento de software são consideradas relevantes para vários autores (Cohn, 2005) (Lomas et al., 2006) (Strode, 2016) (Bick et al. 2018), porém nem sempre são devidamente tratadas. Assim como ocorre em projetos de software “tradicionais”, gerenciar devidamente as dependências é um fator determinante para que as entregas possam ser cumpridas.

Dependências entre as tarefas diminuem o nível de agilidade da equipe (Lomas et al., 2006) e podem impactar o prazo de entrega dos produtos de software. Para Bick et al. (2018), a gestão das dependências é uma questão fundamental no desenvolvimento do software, pois através dela é possível definir a coordenação entre as atividades. Além disso, a identificação adequada de dependências é importante para maximizar a eficiência do projeto e reduzir os riscos (Shen et al., 2003), (Duggan et al., 2004), (Jiang and Jiang, 2008), (Sutherland and Schwaber, 2013), (Korkala and Maurer, 2014), (Strode, 2016). Estudos de Engenharia de Requisitos ágeis (Milicic et al., 2014), (Patton and Economy, 2014) reforçam que maiores ganhos são obtidos quando as dependências são descobertas ou detectadas o mais cedo possível em um projeto.

Dependências não gerenciadas podem restringir ou bloquear o progresso das atividades, levando a atrasos à medida que as pessoas aguardam recursos, aguardam que outras atividades sejam concluídas ou que as informações necessárias sejam disponibilizadas. (Strode, 2016)

Os tipos de dependências utilizadas nesse trabalho seguem a taxomomia proposta por Strode, D.E, Figura 2.3.

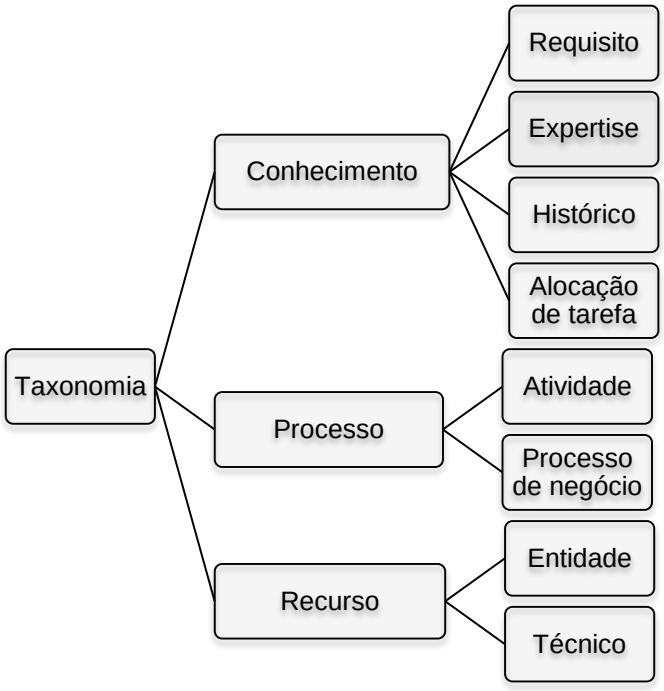


Figure 2.3 Taxomomia de dependências para projetos de desenvolvimento de software ágil (Strode, 2016)

Com as dependências identificadas é possível estabelecer a sequência em que as tarefas podem ser executadas. Essas informações são a base para o desenvolvimento de um cronograma ou plano do projeto.

A partir da identificação das dependências podemos definir a relação lógica entre as tarefas. Em modelos de gestão de projetos tradicionais, as relações lógicas entre as tarefas são definidas como:

- Término para início (TI)
- Início para Término (IT)
- Início para Início (II)
- Término para Término (TT)

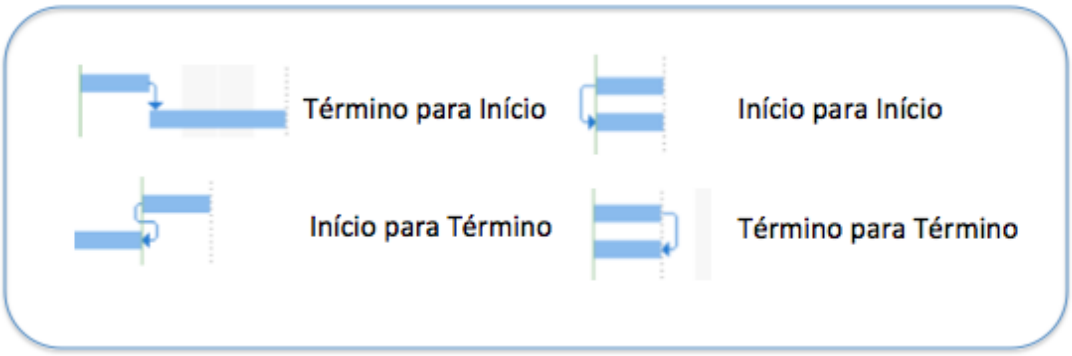


Figura 2.4: Relações Lógicas entre as tarefas

Duas tarefas podem ser dependentes e o tipo da relação lógica entre elas irá determinar quando ambas podem ser iniciadas e/ou concluídas.

Nas relações lógicas entre as tarefas existe o tempo de espera ou antecipação que pode influenciar no prazo de execução das tarefas.

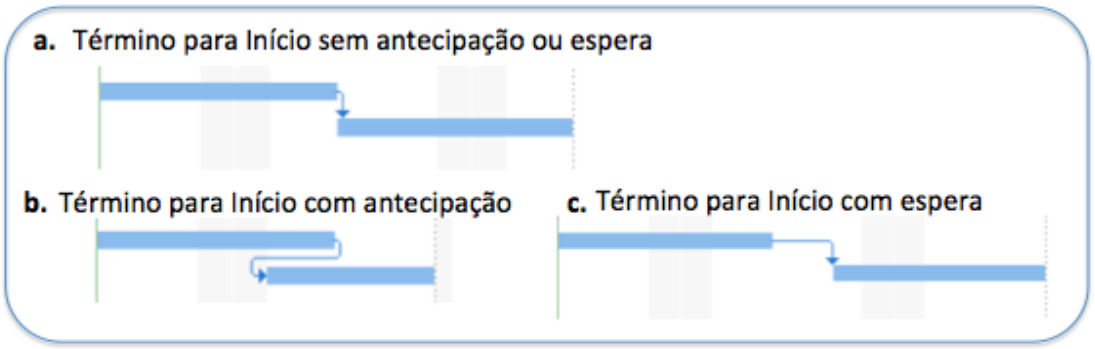


Figura 2.5: Relações Lógicas entre tarefas com antecipação e espera

Na Figura 2.5 mostra como pode ocorrer diferenças durante a execução da tarefa quando consideramos o tempo de antecipação ou espera. No exemplo da Figura 2.5.a tarefa A tem uma relação lógica de término para início. Na Figura 2.5.b mostra a mesma relação lógica com o tempo de antecipação de dois dias e a mesma relação lógica de Término para Início considerando uma espera que ficaria conforme Figura 2.5.c.

O entendimento sobre o relacionamento entre tarefas é importante para o desenvolvimento do planejamento do projeto mesmo em ambientes ágeis. Os tipos de relação lógica das tarefas determinam quando as tarefas poderão ser realizadas. Os

quatro tipos de relacionamento entre tarefas – término a início (TI), início a término (IT), início a início (II) e término a término (TT) – são úteis para esquematizar a sequência lógica para o desenvolvimento das tarefas do projeto.

2.8. Considerações sobre o Capítulo

Este capítulo apresentou conceitos sobre métodos ágeis, gestão de projetos, dependências entre tarefas e o Método do Caminho Crítico.

Os métodos ágeis tem sido uma alternativa à gestão tradicional de projetos de desenvolvimento de software, porém gerenciar esse ambiente ainda tem sido um desafio. Contudo, através de técnicas já consolidadas em modelos tradicionais de gerenciamento de projetos podemos resolver alguns desafios que ainda existem em ambientes ágeis.

O Método do Caminho Crítico é uma abordagem que permite encontrar as tarefas do projeto que não tem flexibilidade de datas, sendo tarefas que devem ser concluídas dentro do prazo determinado. O uso do método em ambientes ágéis pode facilitar o controle das entregas, pois a equipe poderá saber exatamente quais são as tarefas cruciais que podem interferir no prazo da entrega do produto.

O próximo capítulo apresenta um Estudo de Mapeamento Sistemático (EMS) realizado para obter o estado da arte de como a academia tem pesquisado e apresentado soluções para melhorar a gestão de tarefas em ambientes ágeis.

3. Trabalhos Relacionados

Este capítulo apresenta as informações coletadas por meio de um Estudo de Mapeamento Sistemático (EMS) para obter informações mais detalhadas sobre o problema de gestão de dependências de tarefas em ambientes ágeis, constatar a existência do problema em outros contextos, obter o estado da arte das soluções já existentes, e por fim, fundamentar as seguintes etapas da pesquisa.

3.1 Introdução

Este capítulo está organizado da seguinte forma: a seção atual apresenta uma introdução sobre a importância de pesquisar a existência do problema em outros ambientes; a Seção 3.2 apresenta os objetivos do estudo de mapeamento sistemático; a Seção 3.3 apresenta as etapas realizadas durante o planejamento e a Seção 3.4 a execução do Estudo de Mapeamento Sistemático. As etapas de análise dos documentos recuperados são apresentadas na Seção 3.5. Na Seção 3.6 apresentamos os trabalhos relacionados identificados no estudo. A Seção 3.7 apresenta as ameaças à validade do estudo e a Seção 3.8 conclusões.

Algumas metodologias ágeis preveem um nível de gestão para as tarefas, porém, das principais técnicas aplicadas (VersionOne, 2018), nenhuma delas fornece técnicas para gerenciar as dependências que existem no processo de desenvolvimento de software para evitar atrasos na entrega. Os resultados dos estudos mostram que a falta de gestão das dependências entre tarefas produz atividades de planejamento desalinhadas com o que será realmente realizado.

Para apoiar os desenvolvedores de software na escolha da execução das tarefas de forma a não gerar atrasos nas entregas, nós fizemos uma comparação entre os trabalhos relacionados e a proposta dessa dissertação.

Nesse estudo, utilizamos como base, o protocolo de (Petersen et al., 2015), como estratégia para aumentar a confiabilidade do estudo, adicionamos a pesquisa estudos de controle.

3.2 Objetivo da Pesquisa

O estudo de Mapeamento Sistemático (EMS) é um estudo para obter evidências sobre um determinado assunto e fornecer resultados categorizados que foram publicados na área de pesquisa (Petersen et al., 2015) (Barros-Justo et al., 2018).

O estudo de mapeamento sistemático relatado neste capítulo foi conduzido com o objetivo de buscar o estado da arte da literatura sobre o objetivo apresentado no Capítulo 1. A seção atual apresenta o protocolo utilizado para selecionar estudos para esta pesquisa. O protocolo utilizado é o processo de criação de sequências de pesquisa e a definição de um escopo de pesquisa. Esses procedimentos foram conduzidos para ter um resultado abrangente sobre a importância e características das dependências existentes em um processo de desenvolvimento de software em ambientes ágeis. Reconhecendo quais são as propostas atuais, conseguimos extrair as informações como objetivo, metodologia e resultados encontrados nos trabalhos pesquisados. Para este estudo, realizamos uma busca automatizada no Scopus, um banco de dados *on-line,* que indexa vários outros bancos de dados científicos.

O protocolo sugerido por Petersen et al., (2015) também utiliza a abordagem GQM (Solingen et al., 2002) para definir as metas para o EMS. A pesquisa desenvolvida neste trabalho tem como objetivo a análise de artigos que visam identificar como as organizações acompanham a execução de tarefas em ambiente ágeis.

O estudo está organizado nas etapas de planejamento, execução e análise. Facilitando assim a replicação da pesquisa.

Os objetivos deste estudo são:

> **Analisar** projetos de desenvolvimento de software em ambientes ágeis
> **Com o propósito de** caracterizar
> **Com respeito à** estratégia, método, métricas e resultados
> **Do ponto de vista do** pesquisador
> **No contexto** do uso das dependências entre as tarefas.

Uma vez definido o objetivo deste Estudo de Mapeamento Sistemático, as próximas seções apresentam as etapas de planejamento e execução do EMS de acordo com a meta estabelecida.

3.3 Planejamento

O planejamento do estudo foi dividido em três atividades: (1) definir as perguntas de pesquisa para o EMS de acordo com os objetivos e (2) utilizar o PICO - *Population or Problem, Intervention, Comparison e Outcome* (PAI et al., 2004) para auxiliar as sequências de pesquisa e, a partir disso, (3) criar uma *string* de busca.

De acordo com o objetivo definido para este EMS, a seguinte questão de pesquisa é apresentada:

Quais são as estratégias de gestão das dependências entre tarefas existentes em ambientes ágeis de projetos de desenvolvimento de software?

O objetivo dessa questão de pesquisa nos permite identificar os estudos existentes na área que nos permitiu criar um resumo e uma categorização dos trabalhos relacionados.

Para organizar e estruturar a *string* de busca com base no objetivo e na questão de pesquisa, nós utilizamos a abordagem PICO - *Population or Problem, Intervention, Comparison e Outcome*. A Tabela 3.1 apresenta a descrição e objetivos do PICO.

Tabela 3.1: PICO (Pai et al., 2004) do EMS.

(P) População (*Population*)**:**	Desenvolvimento de Software em ambientes ágeis
(I) Intervenção (*Intervention*)	Pesquisas relacionadas com identificação de dependências entre tarefas
(C) Comparação (*Comparison*)	Não há, o objetivo principal do estudo é caracterizar a abordagem e não compará-las.
(O) Resultados (*Output*)**:**	Soluções associada à gestão de tarefas

Baseados na estrutura PICO, os seguintes termos e seus sinônimos foram identificados:

Tabela 3.2**:** Lista de sinônimos do EMS.

(P) *Population*	Agile Software Development, Agile
(I) *Intervention*	Dependency, Dependency task, Agile Dependency,
(C) *Comparison*	Não há.
(O) *Output*	technique, technical, approach, tool, mechanism, Methods research, study, analysis;

A partir desta lista de sinônimos a *String* de busca foi definida e é apresentada no Quadro 3.1.

Quadro 3.1: *String* de busca

("Agile Software Development" OR "Agile") AND ("Dependency" OR "Dependency task" OR "Agile Dependency") AND ("techni*" OR "approach" OR "method*" OR "tool" OR "research" OR "analysis")

String de busca utilizada no *Scopus*:

TITLE-ABS-KEY (("Agile Software Development" OR "Agile") AND ("Dependency" OR "Dependency task" OR "Agile Dependency") AND ("techni*" OR "approach" OR "method*" OR "tool" OR "research" OR "analysis"))

Para a seleção dos artigos, foram estabelecidos critérios de inclusão e exclusão para subsidiar a decisão de quais trabalhos deveriam ser lidos ou não. As etapas de seleção executadas são descritas abaixo:

Critérios de inclusão:

(I1) Trabalhos que abordam tema sobre gestão ou identificação de dependências em projetos de desenvolvimento de software voltados para os ambientes ágeis, apresentando técnicas, métodos, ferramentas, abordagens e análises;

Critérios de Exclusão de Artigos:

(E1) Trabalhos fora da área de computação; OU

(E2) Trabalhos cujo foco de pesquisa encontra-se em outras áreas da computação que não seja a de Engenharia de Software, Gestão de Negócios, Engenharia, Ciência da Computação; OU

(E3) Propostas que não sejam aplicadas a desenvolvimento de software; OU

(E4) Artigos escritos em idiomas diferentes do inglês;

Um grupo de artigos de controle foi definido; sendo esta a estratégia mais comum para avaliar a busca (Petersen et al., 2015) . Os artigos de controle que devem ser obrigatoriamente recuperados pela *String* de busca são:

Aslam, W., Ijaz, F. (2018) **A Quantitative Framework for Task Allocation** In Distributed Agile Software Development In *IEEE Access* 6, pp. 15380-15390 Institute of Electrical and Electronics Engineers Inc.

Strode, D.E. (2016) **A dependency taxonomy for agile software development projects** In *Information Systems Frontiers* 18(1), pp. 23-46

Trkman, M. Mendling, J., Krisper, M. (2016) **Using business process models to better understand the dependencies among user stories** In *Information and Software Technology* 71, pp. 58-76

A seleção dos estudos ocorreu em três etapas:

- ***Etapa 1 - Seleção preliminar das publicações:*** A seleção preliminar das publicações foi realizada por meio da execução da *string* de busca no banco de dados do *Scopus*.
- ***Etapa 2 – Seleção das publicações relevantes – 1º filtro:*** Após a seleção das publicações realizadas na Etapa 1. Foi aplicado o 1º filtro para uma seleção inicial das publicações relevantes. Para isso, os resumos de cada artigo serão lidos e avaliados de acordo com os critérios de inclusão e exclusão definidos no planejamento do estudo. Em alguns artigos, apenas a leitura dos resumos não foi o suficiente, gerando dúvidas se deveriam ou não ser incluídos. Neste caso, os artigos foram selecionados para leitura completa.
- ***Etapa 3 – Seleção das publicações relevantes – 2º filtro:*** A seleção final das publicações relevantes dar-se-a por meio da leitura completa dos textos das publicações selecionadas no 1º filtro. Estas publicações também serão avaliadas de acordo com os critérios de inclusão e exclusão. A avaliação para a inclusão dos artigos retornados na busca foi realizada pela autora da dissertação.

3.4 Execução

De acordo com os procedimentos de seleção dos artigos definidos no planejamento, a primeira etapa consistiu na execução da *string* de busca na fonte de dados selecionada.

Nesta etapa foram retornados 203 estudos distintos durante a primeira versão do estudo realizado em Fevereiro/2018. Em Dezembro/2018 atualizamos o estudo e obtivemos 220 artigos realizando a consulta com a mesma *String* de Busca utilizada no primeiro estudo. A Tabela 3.2 e a Tabela 3.3 exibem os resultados desta 1ª etapa.

Tabela 3.3: Retorno da *String* de busca

Base de Busca	**Número de Publicações retornadas**
Scopus	203 (Fevereiro/2018) 220 (Dezembro/2018)

Tabela 3.4: Número de artigos recuperados na busca

Ano da Publicação	**Número de Publicações**
2019	2
2018	21
2017	18
2016	18
2015	22
2014	20
2013	22
2012	13
2011	18
2010	16
< 2010	50
Total	***220***

Após a etapa inicial, os critérios de Inclusão/Exclusão foram aplicados onde restaram 83 artigos. Na etapa seguinte, foram lidos os títulos e abstracts, resultando na seleção de 17 artigos.

Na quarta etapa, foi realizada a leitura do texto completo dos 17 artigos, resultando na seleção dos 6 estudos relevantes. Na quinta etapa, por fim, incluímos 2 artigos que julgamos relevantes para o estudo, conforme Figura 3.1.

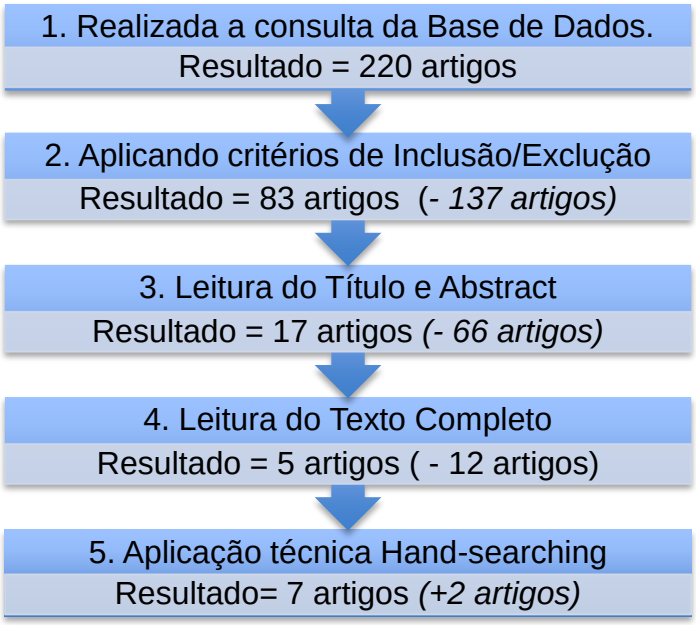


Figura 3.1: Números de artigos em cada etapa do processo de seleção.

A lista de 17 artigos resultantes após aplicação do Passo 3 (Leitura do Título e Abstract), é apresentada na Tabela 3.5.

Tabela 3.5: Artigos selecionados após a leitura do Título e do *Abstract*

Ano	Referência
2018	Alhazmi, A., Huang, S., (2018). **A Decision Support System for Sprint Planning in Scrum Practice**. In *IEEE Southeastcon)* volume 87 Pages 316-334
2018	Bick, S., Spohrer, K., Hoda, R., Heinzl, A. (2018) **Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings** In *IEEE Transactions on Software Engineering* Volume 44, Issue 10, 1 October 2018, Article number 7990187, Pages 932-950
2018	Aslam, W., Ijaz, F. (2018) **A Quantitative Framework for Task Allocation** In Distributed Agile Software Development In *IEEE Acess* 6, pp. 15380-15390 Institute of Electrical and Electronics Engineers Inc.
2017	Elamin, R., Osman, R. (2017) **Towards Requirements Reuse by Implementing Traceability in Agile Development** In *Proceedings - International Computer Software and Applications Conference* 2,8029969, pp. 431-436
2017	Silvax, A., Silva, A., Araújo, T. Perkusich, M,, Dilorenzo, E. (2017) **Ordering the product backlog in agile software development projects: A systematic literature review** In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, SEKE pp. 74-80
2016	Trkman, M. Mendling, J., Krisper, M. (2016) **Using business process models to better understand the dependencies among user stories** In *Information and Software Technology* 71, pp. 58-76
2016	Strode, D.E. (2016) **A dependency taxonomy for agile software development projects** In *Information Systems Frontiers* 18(1), pp. 23-46
2016	Krishnan, B.S., Kovvuri, H., Balasubramani, U.M. (2016) **Effective management of work in a geographically dispersed team using Tasks in Agile methodology** In *2015 IEEE International Symposium on Software Reliability Engineering Workshops*, ISSREW 2015 7392038, pp. 44-5
2016	Grimaldi, P., Perrotta, L., Corvello, V., Verteramo, S. (2016) **An agile, measurable and scalable approach to deliver software applications in a large enterprise** In *International Journal of Agile Systems and Management* 9(4), pp. 326-339
2015	Scheerer, A., Bick, S., Hildenbrand, T., Heinzl, A. (2015) **The effects of team backlog dependencies on agile multiteam systems: A graph theoretical approach** In *Proceedings of the Annual Hawaii International Conference on System Sciences* 2015-March,7070428, pp. 5124-5132
2015	Heikkilä, V.T., Paasivaara, M., Rautiainen, K., (...), Toivola, T., Järvinen, J. (2015**) Operational release planning in large-scale scrum with multiple stakeholders - A longitudinal case study at F-secure Corporation** In *Information and Software Technology* 57(1), pp. 116-140
2013	Staron, M., Meding, W., Hoglund, C., (...), Nilsson, J., Hansson, J. (2013) **Identifying implicit architectural dependencies using measures of source code change waves** In *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications*, SEAA 2013 6619529, pp. 325-332
2013	Strode, D.E. (2013) **Extending the dependency taxonomy of agile software development** In Lecture Notes in Computer Science (including

	subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8224 LNCS, pp. 274-289
2013	Martakis, A., Daneva, M. (2013) **Handling requirements dependencies in agile projects: A focus group with agile software development practitioners** *In* Proceedings - International Conference on Research Challenges in Information Science 6577679
2013	Daneva, M., Van Der Veen, E., Amrit, C., (...), Ramteerthkar, U., Wieringa, R. (2013) **Agile requirements prioritization in large-scale outsourced system projects: An empirical study** In Journal of Systems and Software 86(5), pp. 1333-1353
2010	Koru, A.G., El Emam, K. (2010) **The theory of relative dependency: Higher coupling concentration in smaller modules** In *IEEE Software* 27(2),5420801, pp. 81-89
2010	Gomez, A., Rueda, G., Alarcón, P.P. (2010) **A systematic and lightweight method to identify dependencies between user stories** In *Lecture Notes in Business Information Processing* 48 LNBIP, pp. 190-195

Os artigos Ujigawa and Updegrove, (2016) e Lomas et al., (2006) foram incluídos como parte do estudo por serem considerados relevantes para a pesquisa. Esses artigos foram recuperados durante as revisões da literatura *Ad hoc*.

Tabela 3.6: Artigos selecionados durante pesquisas *Ad hoc*

Ano	**Referência**
2016	Ujigawa, K., Updegrove, D. (2016) **"Agile" CCPM: Critical Chain for Software Development** In *TOCICO Theory of Constraints International Certification Organization.*
2008	C. D. W. Lomas*, J. Wilkinson*, P.G. Maropoulos†, P. C. Matthews* (2008) **Measuring design process agility for the single company product development process** *In* International Journal of Agile Manufacturing - 2008

A partir dos artigos apresentados na Tabela 3.5 foi realizada a leitura completa dos textos onde foram excluídos 12 artigos. Os motivos para a exclusão dos artigos estão apresentados no Apêndice A - Formulário de Exclusão do Estudo de Mapeamento Sistemático.

A Tabela 3.7 apresenta os sete artigos selecionados. Esta tabela apresenta os artigos selecionados com as seguintes informações: identificador, ano de publicação,

autores e título do artigo. O identificador criado para cada artigo servirá de referência no texto posteriormente.

Tabela 3.7: Artigos Selecionados EMS

ID Artigo	Ano	Autores	Título
A1	2018	Bick, S., Spohrer, K., Hoda, R., Heinzl, A. (Bick et al., 2018)	**Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings**
A2	2018	Aslam, W., Ijaz, F. (Aslam and Ijaz, 2018)	**A Quantitative Framework for Task Allocation**
A3	2016	Trkman, M. Mendling, J., Krisper, M. (Trkman et al., 2016)	**Using business process models to better understand the dependencies among user stories**
A4	2016	Strode, D.E. (Strode, 2016)	**A dependency taxonomy for agile software development projects**
A5	2016	Ujigawa, K., Updegrove, D. (Ujigawa and Updegrove, 2016)	**Agile" CCPM: Critical Chain for Software Development**
A6	2013	Martakis, A., Daneva, M. (Martakis and Daneva, 2013)	**Handling requirements dependencies in agile projects: A focus group with agile software development practitioners**
A7	2008	C. D. W. Lomas, J. Wilkinson, P.G. Maropoulos†, P. C. Matthews (Lomas et al., 2006)	**Measuring design process agility for the single company product development process**

Todos os artigos apresentados na tabela 3.7 foram lidos e identificamos os dados extraídos que estão apresentados na tabela 3.8. Cada campo possui uma informação e uma descrição da informação.

As informações extraídas dos artigos selecionados estão apresentadas na Tabela 3.8. A análise dos dados extraídos será apresentada na próxima seção.

Tabela 3.8: Informações extraídas dos artigos selecionados

Informação	Descrição
Id Artigo	Identificador do artigo (encontrado na tabela 3.3)
Objetivo do Estudo	Estratégia utilizada em cada estudo
Metodologia proposta	Informação sobre a metodologia utilizada no estudo
Resultado	Encontrado no estudo

3.5 Análises

A análise das informações extraídas dos artigos reunidos durante a execução é apresentada nesta seção. Para este trabalho, propõe-se uma categorização dos trabalhos, reunindo artigos de acordo com o assunto discutido e apontando questões e diferenças específicas em relação ao que é proposto nesta dissertação. A extração das informações dos artigos foi realizada pelo autor desta dissertação.

Esse processo de categorização segue o protocolo de mapeamento sistemático proposto por (Petersen et al., 2015).

Com base nas informações extraídas dos estudos, foi possível responder à pergunta de pesquisa deste EMS, formulando a Tabela 3.5 e as próximas seções. As próximas subseções descrevem a análise.

3.5.1 Avaliação dos Resultados

Baseado nas informações extraídas dos artigos selecionados foi possível elaborar as respostas iniciais para as questões de pesquisa formuladas no planejamento deste estudo.

Para responder à questão principal “**Quais são as estratégias para tratar dependências entre tarefas em ambientes ágeis de projetos de desenvolvimento**

de software?” selecionamos os artigos com os respectivos resultados, conforme mostrado na Tabela 3.9.

Tabela 3.9: Resultados extraídos dos artigos

ID Artigo	Objetivo do Estudo	Metodologia Proposta	Resultado
A1	Investigar a combinação do planejamento tradicional com o desenvolvimento ágil. A falta de conscientização sobre dependência entre tarefas é uma das principais causas de um desempenho ineficaz.	É apresentado um estudo de caso em uma unidade de desenvolvimento de software com 13 equipes. No estudo é explorado como e porque a combinação do planejamento tradicional e o desenvolvimento ágil em nível de equipe, ainda assim, pode resultar em uma coordenação ineficaz. É apresentada uma variedade de dados, incluindo entrevistas com *Scrum Masters,* proprietários de produtos, arquitetos e gerentes seniores. Usando os procedimentos de análise de dados da *Grounded Theory*, foi identificada uma falta de conscientização sobre dependências entre as equipes de desenvolvimento como uma explicação chave da coordenação ineficaz.	Através dos dados empíricos coletados ficou evidenciado a importância da gestão de dependência para o desenvolvimento das tarefas na equipe e em tarefas inter-equipes.
A2	Atender às necessidades de mercados emergentes de desenvolvimento de software em um paradigma ágil e distribuído.	Foi proposta uma estrutura de alocação de tarefas composta de duas fases: uma, identificando fatores e dependências que influenciam fortemente a decisão de alocação de tarefas; dois, propondo um método quantitativo que aloca tarefas aos membros da equipe que melhor correspondem aos requisitos da tarefa.	É apresentada uma estrutura de alocação de tarefas que considera dependências e fatores influentes para auxiliar na tomada de decisão sobre a alocação de tarefas.
A3	É proposto um método	É utilizada uma abordagem de engenharia de	O artigo aborda um problema que surge do

	que facilita uma melhor compreensão da ordem de execução e das dependências de integração de histórias de usuários, fazendo uso de modelos de processos de negócios. O método associa as histórias do usuário ao elemento de atividade do modelo de processo de negócios correspondente.	método situacional para definir nosso método proposto. Para fornecer compreensão dos métodos propostos. Foram utilizados conceitos ontológicos.	gerenciamento de histórias de usuários em projetos de desenvolvimento de software e foca no contexto ausente de uma história de usuário. O método contribui para a disciplina de modelagem conceitual no desenvolvimento ágil. O experimento fornece uma visão empírica das dependências dos requisitos entre histórias.
A4	Entender situações significativas que os profissionais podem projetar ou selecionar mecanismos apropriados a partir dos métodos ágeis para lidar com as dependências antes que elas bloqueiem o progresso do projeto.	Artigo analisa as dependências em três casos típicos de desenvolvimento ágil de software co-localizado e apresenta as dependências como uma taxonomia com regras de decisão para alocação de dependências em categorias.	O artigo apresenta uma taxonomia de dependências que estão presentes em projetos de software em ambientes ágies. O resultado das análises dos casos mostra que as dependências de conhecimento, processo e recursos estão mais presentes com predominância da dependência do conhecimento.
A5	Melhorar o gerenciamento de projetos em contextos ágeis para prover mais adaptabilidade aos desenvolvedores, capacidade de resposta e	O artigo constrói uma rede de tarefas para ser utilizada na gestão de buffers para uso em um ambiente de desenvolvimento de software ágil. Através da expansão do conhecimento sobre a Teoria das Restrições e Corrente Crítica, ele sugere a adoção do gerenciamento de *buffer*,	Foi projetada uma estratégia de ganho mútuo entre desenvolvedores e suas organizações para planejar e escolher os melhores recursos para as tarefas do projeto, assim como realizar a gestão de *buffer* entre as tarefas e definir o período necessário para atender à entrega.

	autonomia em seus projetos.	adotando metodologias ágeis.	
A6	Esta pesquisa se propõe a descobrir conceitos que os profissionais em empresas de vários tamanhos em todo o mundo e em vários setores usam para lidar com dependências de requisitos em seus projetos ágeis de software.	Foi utilizada pesquisa online em grupo, usando um fórum para discussão.	O estudo resultou nas seguintes descobertas: (1) dependências de requisitos ocorrem em projetos ágeis e são importantes para o sucesso desses projetos, assim como isso é conhecido por projetos de software "tradicionais"; (2) dependências dos requisitos (i) foram consideradas e tratadas como parte do gerenciamento de riscos, (ii) foram consideradas como responsabilidade dos membros individuais da equipe e (iii) afetaram principalmente o planejamento do projeto; (3) comunicação e colaboração contínuas – são duas características essenciais de qualquer método ágil. Foram consideradas críticas para mitigar os riscos devido a dependências; (4) uma abordagem híbrida para a arquitetura entre métodos ágeis e planejados foi percebida para produzir escalabilidade máxima e ajudar a lidar com dependências; (5) "preocupações transversais", uma categoria de dependências, não foram uniformemente entendidas em um contexto ágil e exigem mais pesquisas.
A7	Este artigo procura explorar a maneira pela qual a agilidade em todo o processo de	São apresentados dois estudos de caso com os dados qualitativos empíricos sobre os cronogramas do projeto e eventos inesperados. Alguns dados também foram	Através de dados coletados de dois estudos de casos foi possível calcular a medida e forneceu um feedback de cada projeto.

	desenvolvimento do produto pode ser medida usando uma medida de agilidade previamente definida: (KAY) *Key Agility Index*.	coletados por meio de entrevistas com especialistas e podem ser usados com o (KAY) *Key Agility Index* para fornecer uma medida realista do tempo de projeto.	No trabalho, o autor identifica que as dependências das tarefas interferem no índice KAI do projeto.

Durante a fase de análise da pesquisa identificamos que muitos trabalhos que têm como tema “Dependência entre tarefas” também trata questão como coordenação de tarefas (A4) e alocação de tarefas (A1, A2 e A7).

Nos trabalhos relacionados, os autores apresentam estudos complementares à proposta de pesquisa desta dissertação. Esses estudos serviram como base teórica para aplicação das técnicas, métodos e ferramentas.

O Trabalho A1 (Bick et al., 2018) apresenta uma proposta que enfatiza a importância da gestão de dependências entre tarefas em ambiente ágeis e propõe um modelo híbrido para a realização dos projetos, porém a sua proposta diferencia-se da proposta original dessa dissertação, que é sugerir uma abordagem que preserve as características das metodologias ágeis. No trabalho A2 (Aslam and Ijaz, 2018), o autor propõe um método para alocação de tarefas em ambientes ágeis. Contudo, nosso trabalho buscou preservar a auto-organização da equipe e não influenciamos a distribuição das tarefas. No nosso trabalho apenas damos direcionamentos sobre quais tarefas podem impactar a execução de outras. No trabalho A3 (Trkman et al., 2016), o autor trata as dependências entre histórias, porém durante esses EMS vimos que existem outras dependências que estão presentes no processo de desenvolvimento de software, o que torna o trabalho pouco abrangente. O trabalho A4 (Strode, 2016) mostra um estudo aprofundado sobre as dependências presentes em ambientes ágeis. Apesar de o estudo caracterizar bem os diversos tipos de dependências, o trabalho não apresenta uma abordagem prática para uso dessa informação no dia a dia de uma equipe. O trabalho A5 (Ujigawa and Updegrove, 2016) utiliza a análise do Corrente Crítica em ambientes ágeis, porém para o uso efetivo da técnica proposta é necessário que as tarefas sejam estimadas em tempos absolutos, ao invés, de tempos relativos como geralmente é visto em ambientes ágeis. Outra questão sobre o trabalho A5 é que a proposta é voltada para gerenciar buffer e o tempo de execução da entrega. Aparentemente no tabalho A5, as dependências não ficam explícitas para o time. No trabalho A6 (Martakis and Daneva, 2013), o autor identifica a importância da gestão das dependências entre tarefas para o sucesso dos projetos em ambiente ágeis, porém não é apresentado nenhum modelo ou método para tratar a questão. No trabalho A7 (Lomas et al., 2006), o autor utiliza dados coletados de projetos para induzir a métrica (KAY) *Key Agility Index* para medir o índice de agilidade do projeto. Apesar de o trabalho identificar que as dependências influenciam no índice de agilidade do projeto, não faz parte do escopo da nossa

pesquisa dar algum suporte ao uso da informação das dependências durante o desenvolvimento.

3.6 Ameaça à Validade

No planejamento: protocolo e processo do EMS foram planejados para permitir a replicação de pesquisa, bem como minimizar o viés. Porém, todo o processo foi definido pela autora sem participação de outro pesquisador. Os critérios de inclusão e exclusão definidos também podem ter colaborado para que perdêssemos alguns estudos.

Na análise dos resultados: A interpretação dos dados foi realizada pela autora e isso pode ter influenciado nos resultados encontrados.

3.7 Considerações sobre o Capítulo

Este Capítulo apresentou o Estudo de Mapeamento Sistemático realizado para encontrar estudos sobre a identificação das dependências entre as tarefas em ambientes ágeis. Durante o estudo, identificamos que existem vários trabalhos que reforçam a existência do problema da falta de uma gestão das dependências entre as tarefas em ambiente ágeis, sendo esse também o problema que buscamos resolver.

As fases de planejamento, execução e análise do estudo foram executadas duas vezes, primeiro em fevereiro de 2018 e depois em dezembro de 2018. A reexecução do estudo teve por objetivo atualizar os resultados com estudos mais recentes e perceber se os procedimentos planejados poderiam ser reproduzidos.

Vimos nos trabalhos relacionados que todos apresentam abordagens diferentes da proposta dessa dissertação. Na avaliação dos resultados, apresentamos as diferenças entre os trabalhos relacionados e o trabalho apresentado. Devido à diversidade dos trabalhos, não conseguimos encontrar características comuns para comparação, mas conseguimos identificar em cada trabalho pontos que bucamos alcançar com a solução implementada na ferramenta *AgileCritPath*.

Essas características apontadas na Avaliação dos Resutaldos indicam a relevância que esse trabalho pode ter para a academia e para a indústria. O próximo capítulo apresenta o estudo exploratório realizado para validar a proposta desta dissertação.

4. Estudo Exploratório

Nesse Capítulo será apresentado o estudo explorarório realizado em uma empresa brasileira de desenvolvimento de software. Os principais objetivos deste estudo são: (1) caracterizar a existência de dependências entre tarefas em projetos de desenvolvimento de software que utilizam métodos ágeis; (2) validar a proposta do trabalho que é utilizar práticas do Método do Caminho Crítico em projetos que utilizam métodos ágeis.

Para alcançar os objetivos desse estudo, desenvolvemos uma investigação empírica para validarmos as propostas da dissertação utilizando dados reais de uma empresa com atuação na área de desenvolvimento de software.

4.1 Introdução

A fim de compreender adequadamente a dinâmica da organização no que diz respeito ao planejamento de projetos em um ambiente ágil, optou-se por realizar uma pesquisa de ação exploratória no contexto de uma organização de desenvolvimento de software.

Segundo Runeson e Host (Runeson and Höst, 2009), os estudos exploratórios têm se mostrado adequados na engenharia de software quando se procura estudar novas ideias. Por isso, com o intuito de buscar mais indícios sobre a motivação dessa dissertação, realizou-se, no presente trabalho, este estudo em uma empresa de desenvolvimento de software brasileira.

Através do estudo exploratório, identificamos uma maneira apropriada de investigar fenômenos como: (1) caracterizar a existência de dependências entre tarefas em projetos reais de desenvolvimento de software que utilizam métodos ágeis; (2) aplicar conceitos baseados no Método do Caminho Crítico em projetos que utilizam métodos ágeis.

O estudo exploratório aqui apresentado possui as seguintes etapas:

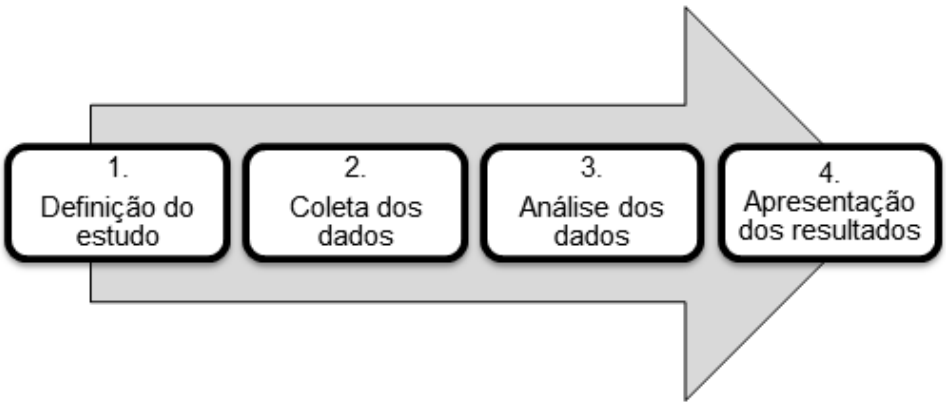


Figura 4.1: Etapas da descrição do Estudo de Observação

A figura 4.1 mostra as etapas definidas para o estudo.

Este estudo foi realizado em uma empresa de desenvolvimento de software que atualmente possui um quadro de 30 funcionários, sendo a maioria desenvolvedores de software. A organização já utiliza métodos ágeis há pelo menos dez anos. As equipes são pequenas, geralmente formadas por grupos de três a oito pessoas sendo: analistas de sistemas, desenvolvedores e testadores.

Durante anos a empresa utilizou Scrum como um guia nos processos de desenvolvimento dos seus sistemas, porém, com o tempo, alguns aspectos da metodologia foram adaptados à necessidade da organização (De França et al., 2017).

4.2 Definição do Estudo

O estudo exploratório foi planejado com o propósito de caracterizar a viabilidade da ideia do trabalho sob dois aspectos: 1) observar a existência de dependências entre tarefas em projetos reais de desenvolvimento de software que utilizam métodos ágeis; 2) aplicar conceitos baseados no Método do Caminho Crítico em projetos que utilizam métodos ágeis.

Para o estudo, estamos considerando dados do sistema de gestão de tarefas da empresa. Através dos dados, conseguimos capturar informações das tarefas definidas durante o planejamento da iteração e informações de execução das tarefas.

A identificação das dependências foi realizada com a ajuda de um desenvolvedor mais experiente que possuía o conhecimento das funcionalidades a

serem desenvolvidas e o conhecimento do processo de desenvolvimento utilizado na organização. O uso do Método do Caminho Crítico em ambientes ágeis foi utilizado dentro das tarefas da iteração já que não tínhamos o planejamento total do projeto. Contudo, mesmo utilizando apenas as tarefas de uma iteração, conseguimos avaliar o uso do método nesse ambiente.

4.2.1. Coleta dos dados

Na fase de coleta dos dados extraímos as tarefas realizadas durante a iteração do sistema de gestão de tarefas Redmine. No sistema de gestão de tarefas da organização conseguimos identificar quais tarefas estavam planejadas na iteração e quais tarefas foram efetivamente realizadas.

Tabela 4.1: Resumo das tarefas da iteração analisada - Iteração 2.14

		Quantidade de Tarefas
DSV	Planejadas	26
	Não Planejadas	18
TST	Planejadas	14
	Não Planejadas	12
Total		70

Na tabela 4.1 podemos observar que as tarefas realizadas na iteração 2.14 são categorizadas como tarefas de desenvolvimento (DSV) e tarefas de testes (TST). Ao final da reunião de planejamento a equipe definiu 26 tarefas planejadas para a equipe de desenvolvimento e 14 tarefas de testes totalizando assim 40 tarefas planejadas na iteração. No final da iteração foi possível observar que foram realizadas todas as tarefas planejadas, mais 18 tarefas não planejadas de desenvolvimento e mais 12 tarefas não planejadas de testes.

Na fase de coleta de dados analisamos dados de três iterações no Redmine da organização.

Tabela 4.2: Resumo das tarefas das iterações 2.12, 2.13 e 2.14

Tarefas	**Sprint 2.12**	**Sprint 2.13**	**Sprint 2.14**

DSV	Planejadas	30	18	26
	Não Planejadas	33	14	18
TST	Planejadas	25	25	14
	Não Planejadas	12	12	12
Total		100	69	70

4.2.2. Análise dos dados

Após a etapa 2 – Coleta dos dados, realizamos a análise dos dados. Nessa etapa fizemos um trabalho para identificação das dependências entre as tarefas. A identificação das dependências foi realizada com a ajuda de um desenvolvedor experiente que conhecia o escopo das tarefas, o processo da organização e conhecia também a arquitetura tecnológica utilizada no projeto.

Para todas as tarefas da iteração analisada no estudo, identificamos as suas respectivas dependências. Nessa fase procuramos também caracterizar o tipo de dependência encontrada. Os tipos das dependências identificadas foram dependências do processo, atividade e requisito. Os tipos das dependências seguiram a classificação sugerida na taxonomia dependências (Strode, 2016).

Na tabela 4.3 mostra um exemplo de formulário utilizado para a caracterização das dependências.

Tabela 4.3: Formulário de caracterização das dependências

#	Tipo	Situação	Título	Esforço	Dependência	Tipo da Dependência
21903	DSV	Done	[Criação Pedido Posicionamento Importacao] - Validação exclusiva para exportação está sendo aplicada para pedidos de importação.	0.5	21849 TI,	Processo
21926	DSV	Done	Alterar a lógica de recuperação de GH em função do tipo do Pedido	4	21849 TI	Processo
21856	TST	Done	Validar a lógica de recuperação de GH em função do tipo do Pedido	4	21926 TI	Processo
21823	DSV	Done	Adaptar o serviço do NavisMiddleware que envia ICU para o ArgoService do N4	5	21839 TI	Processo
21859	TST	Done	Ajustar fluxo, liberar Pagamento para especificar qual DocCobranca está sendo pago	6	21839 TI	Processo
21839	DSV	Done	Ajustar fluxo, liberar Pagamento para especificar qual DocCobranca está sendo pago	6	21846 TI, 21849 TI	Atividade, Processo

Com a análise das tarefas na Tabela 4.3, conseguimos identificar as dependências e classificá-las. A classificação ocorreu de 2 formas: [1] classificamos a dependência quanto a sua relação lógica (TI, II, TT, IT), coluna em verde e [2] classificamos o tipo da dependência coluna em roxo.

Os dados completos da tabela 4.3 encontram-se disponíveis no apêndice B dessa dissertação.

A análise aconteceu em duas etapas onde: (1) etapa 1: analisamos dados das tarefas planejadas da Sprint; e depois (2) etapa 2: analisamos os dados das tarefas após a execução da iteração.

Durante a 1ª etapa, constatamos que a falta de gestão das dependências entre as tarefas resulta em atividades mal planejadas. O gráfico de *Burndown* definido após a reunião de planejamento serve para acompanhamento da Iteração (ver Figura 4.2). Porém, após as análises dos dados podemos constatar que:

1. O gráfico de *Burndown* não considera as dependências que existem entre as tarefas. No caso de uma iteração com muitas dependências

entre as tarefas, a velocidade representada no gráfico não estará correta e isso poderá comprometer o prazo da entrega do produto;

2. Durante a iteração a equipe não conhece as dependências das tarefas. O conhecimento sobre dependência é individual e nem todos da equipe compartilham o mesmo conhecimento;
3. Como as tarefas de testes dependiam muitas vezes de um conjunto de tarefas de desenvolvimento, isso não ficava explícito no gráfico *Burndown* e nem no quadro *Kanban*;
4. As tarefas não planejadas são incluídas na iteração e invalida o planejamento que foi feito na reunião de planejamento. Assim como a inclusão das tarefas não planejadas, novas restrições de dependências entre as tarefas surgem na iteração;
5. Os erros quando descobertos pelo testador, são incluídos como tarefas de erro na iteração, logo o erro está sempre associado a alguma tarefa de teste e em alguns casos pode estar associado à tarefa de desenvolvimento também. Essas dependências não estavam sendo gerenciadas;
6. Mesmo tendo equipe fixa nas iterações, a velocidade da equipe variava e um dos possíveis fatores era consequência da quantidade de dependências que podem existir entre as tarefas;
7. O acompanhamento através do gráfico de *Burndown* não condiz com o que a equipe seria capaz de realizar quando consideramos as dependências das tarefas;

Na 2ª etapa, fizemos uma nova análise com as tarefas concluídas. Fizemos as seguintes constatações:

1. As dependências entre as tarefas dificultam a realização das atividades na velocidade esperada para que o objetivo da iteração seja alcançado no prazo;
2. Durante a execução a equipe priorizava as tarefas seguindo alguns critérios, mas nem sempre priorizava as tarefas que poderiam bloquear a execução de outras;
3. Como muitas vezes tinha mais de uma tarefa de desenvolvimento para uma tarefa de teste, a equipe tinha que ficar atenta para executar essas tarefas de desenvolvimento juntas para não impactar o andamento da tarefa de testes;

4. Acompanhar o andamento das iterações através de indicadores não representa a situação real da iteração. Um atraso em uma tarefa fora do caminho crítico não representa um atraso na conclusão da iteração.
5. O atraso em uma tarefa que bloqueia a execução da outra pode impactar a data de entrega e isso não está explícito para a equipe.

4.3 Apresentação dos Resultados

A partir da análise dos dados foi possível chegar aos seguintes resultados: Fizemos um trabalho de projeção da taxa de esforço máximo por período, nesse caso, em dias, considerando o esforço entre as tarefas e suas respectivas dependências.

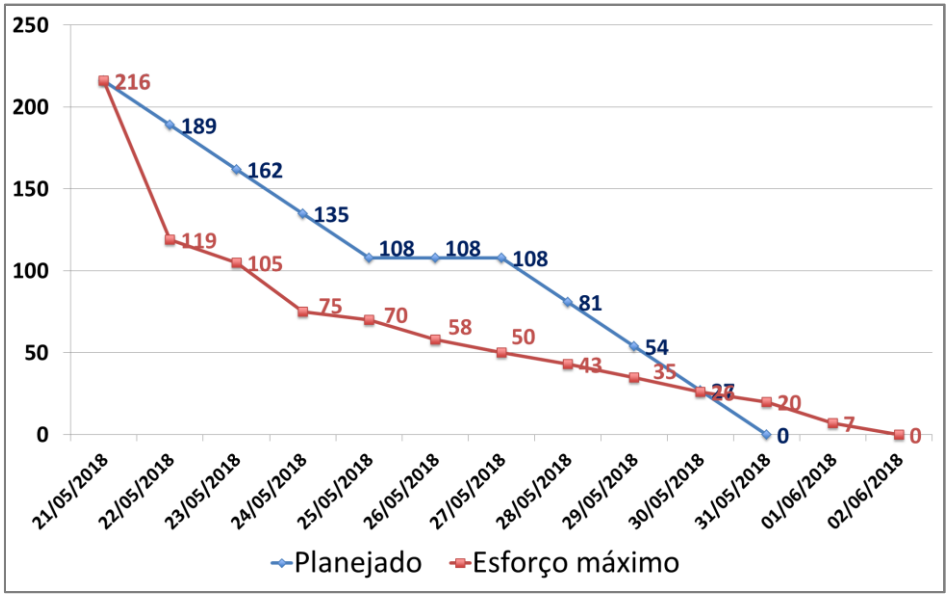


Figura 4.2: Gráfico de *Burndown* com Planejado x Taxa de Esforço Máximo

Na Figura 4.2, é possível observar o Planejamento realizado pela equipe (linha azul) e o tamanho da iteração considerando a taxa máxima de esforço (linha em vermelho). A linha em vermelho representa o tempo mais curto para conclusão das atividades da iteração e indica a taxa máxima de esforço (velocidade) em que a equipe poderá trabalhar. Independente do número de pessoas na iteração, as tarefas não serão concluídas antes desse prazo definido (linha vermelha).

No gráfico apresentado na Figura 4.2, podemos observar que na primeira semana da iteração a equipe conseguiria realizar o planejado já que a taxa de Esforço Máximo é maior que a taxa de Esforço por dia planeajdo para a equipe (linha azul). Porém, na segunda semana, a equipe não conseguiria desenvolver as tarefas dentro

do prazo esperado por conta da dependência que existe entre as tarefas. Essas dependências bloqueiam o desenvolvimento de outras tarefas.

A taxa máxima de esforço é calculada levando em consideração as dependências que impedem a execução de outras tarefas. Para encontrar a taxa máxima de esforço em um determinado período, consideramos o esforço de todas as tarefas que podem ser executadas nesse período.

Analisando o gráfico gerado na Figura 4.2, poderíamos concluir que:

- Levando em consideração as dependências que existem entre as tarefas, o planejamento realizado pela equipe não é possível de ser executado;
- Com uma visão atualizada do gráfico durante a execução da iteração poderíamos observar que um atraso em uma tarefa do caminho crítico irá aumentar o prazo de desenvolvimento e atrasar a conclusão da iteração, independente do número de pessoas na equipe.

Analisando o log de execuções das tarefas de três iterações, extraímos o processo empírico que ocorre na execução das iterações. Com o log de execução das tarefas da Sprint, conseguimos observar os seguintes comportamentos, conforme figura 4.3. Processo mapeado utilizando a notação CMMN (*Case Management Model and Notation*).

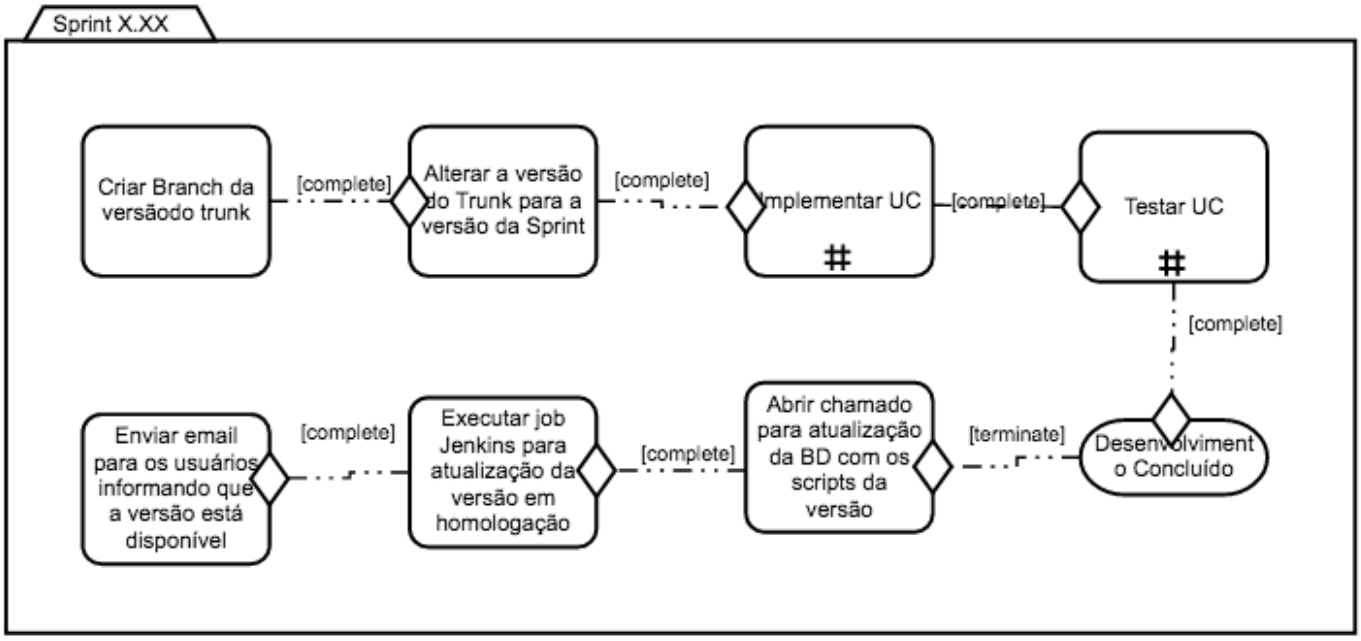


Figura 4.3: Modelo de Processo extraído do Log de Execuções

Sob o aspecto de Gestão de Processos, a execução de uma iteração nada mais é que a instância desse processo representado na Figura 4.3. Na organização analisada, o processo de desenvolvimento de software de uma iteração acontece de uma forma bem simples, porém, ainda sim, existem dependências entre as tarefas.

A partir do modelo de processo extraído, podemos identificar que alguns tipos de dependências conforme a taxinomia proposta por Strode, (2016). As dependências identificadas como vindas do Processo são dependências que existem no modelo de processo da organização (ver figura 4.3). As dependências de Atividade são dependências que fazem parte do domínio do negócio, da aplicação ou da organização (Strode, 2016). Essa informação não existe no modelo de processo, porém essas dependências impactam a execução das tarefas, logo essas informações devem ser consideradas no planejamento.

Tabela 4.4: Tarefas e Dependências identificadas nas Iterações

Iteração 2.14					
Tarefas		**Quantidade de Tarefas**	**Quantidade de Dependências**	**Dependências do Processo**	**Dependências da Atividade**
DSV	Planejadas	26	39	22	17
	Não Planejadas	18	18	18	0
TST	Planejadas	14	15	15	0
	Não Planejadas	12	12	12	0
Total		**70**	**84**	**67**	**17**
Total %			**120%**	**79.76%**	**20.24%**

Iteração 2.13					
Tarefas		**Quantidade de Tarefas**	**Quantidade de Dependências**	**Dependências do Processo**	**Dependências da Atividade**
DSV	Planejadas	18	28	9	19
	Não Planejadas	14	25	14	11
TST	Planejadas	25	13	13	0
	Não Planejadas	12	11	11	0
Total		**69**	**77**	**53**	**24**

Total %		**111.59%**	**68.83%**	**31.17%**

Sprint 2.12					
Tarefas		**Quantidade de Tarefas**	**Quantidade de Dependências**	**Dependências do Processo**	**Dependências da Atividade**
DSV	Planejadas	30	28	9	19
	Não Planejadas	33	24	3	21
TST	Planejadas	25	19	15	4
	Não Planejadas	12	25	25	0
Total		**100**	**96**	**52**	**44**
Total %			**96%**	**54.17%**	**45.83%**

Com o *log* de execução de três iterações, Tabela 4.4 foi possível verificar que os números de dependências identificadas superou a quantidade de tarefas definidas pela equipe nas iterações 2.13 e 2.14. Esse resultado mostra que tratar as dependências das tarefas pode não ser simples quando temos muitas tarefas. Outro fator que nos chamou a atenção foi a quantidade de tarefas, e consequentemente, de dependências que não são planejadas. Essas tarefas e dependências não planejadas mudam o fluxo de execução das tarefas.

No estudo realizado, observamos que a informação de dependências entre as tarefas não são todas inseridas no Redmine da organização, mas a equipe reconhece que é importante ter esse tipo de informação. Mesmo informando as dependências no *Redmine*, a equipe não tem uma visão onde possa identificar a ordem em que as tarefas devem ser executadas.

Durante a fase de análise dos dados na etapa 1, foram identificadas as dependências das tarefas (ver Tabela 4.3) e conseguimos classificar as dependências como dependências de Processo e de Atividade, seguindo a classificação sugerida na taxonomia dependências (Strode, 2016). Strode (2016) apresenta no seu trabalho uma taxonomia de Dependências de projetos de desenvolvimento de software em ambientes ágeis e através desse trabalho, concluímos que outros tipos de dependências também podem ser tratados, em trabalhos futuros.

Com as dependências identificadas, mapeamos a rede de tarefas executadas em uma iteração (ver apêndice C). A figura – apêndice C representa o processo da forma como foi executado na iteração. Quando aumentou o número de execuções de uma atividade do processo, aumentou também as dependências que existem no modelo.

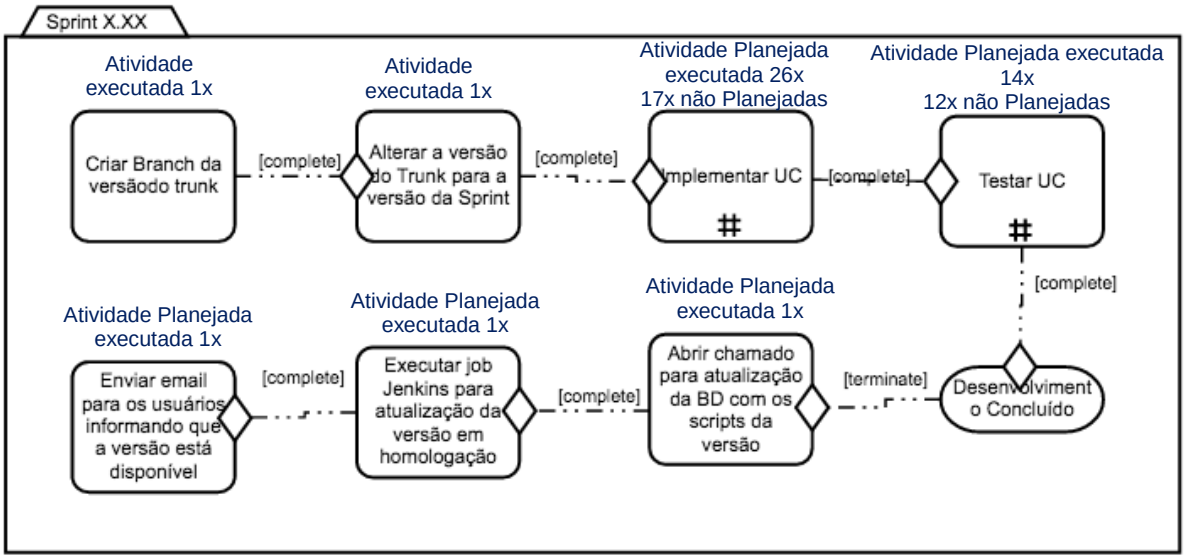


Figura 4.4: Execuções das tarefas do Modelo de Processo

Na análise dos dados conseguimos observar que a atividade “Implementar Caso de Uso” foram planejadas para serem executadas na Sprint 26 vezes e tivemos essa atividade do processo executada mais 17 vezes como tarefa não planejada, totalizando 43 execuções dessa atividade do processo. A atividade definida no processo para Testar Caso de Uso foi planejada no início da Sprint para ser executada 14 vezes, porém durante a execução da Sprint foi executada mais 12 vezes em tarefas não planejadas, totalizando assim 26 execuções dessa atividade do processo Figura 4.4. pocesso mapeado utilizando a notação CMMN (*Case Management Model and Notation*).

No cenário analisado, nós observamos que o número de dependências entre as tarefas quando são planejadas é grande, porém, ainda podem existir as tarefas não planejdas, o que dificulta ainda mais a gestão das tarefas. As tarefas não planejadas são tarefas que são criadas durante o desenvolvimento da versão, podem ser tarefas criadas por causa de um erro no planejamento, podem ser uma funcionalidade que a equipe não tinha total conhecimento no momento do planejamento, alguma mudança

necessária no requisito que precisa ser ajustado nessa versão do produto ou podem ser registro de erro encontrado durante os testes.

Mesmo em um processo simples, como o processo apresentado na Figura 4.3, quando a atividade do processo é instanciada mais de uma vez, o número de dependências aumenta significativamente. Na Figura 4.4 (instância do processo) podemos observar que mesmo em modelos de processos simples, podemos ter uma quantidade grande de dependências que influenciam na execução das tarefas e consequentemente essas dependências devem ser consideradas no planejamento da iteração.

Quando as atividades são dependentes, a duração de uma atividade pode influenciar o tempo de execução da outra tarefa. Para Bick et al., (2018) respostas inadequadas às interdependências geram uma coordenação ineficaz e constituem as principais fontes de falha do projeto (Cataldo et al., 2008), (Cataldo and Herbsleb, 2013). Quando consideramos as dependências entre as tarefas podemos observar que o índice de agilidade diminui (Lomas et al., 2006), logo a velocidade planejada para a iteração pode não ser possível realizar.

Outro fator observado, é que quando estamos decompondo as histórias em tarefas, quanto menor as tarefas, maior a tendência de ter um aumento no número de dependências entre elas. Deixar as tarefas com uma granularidade maior diminui as dependências, mas também influencia na velocidade, porque a tarefa acaba sendo executada por uma pessoa, o que também acaba diminuindo a agilidade da equipe.

4.4 Ameaça à Validade

No estudo exploratório conseguimos identificar possíveis pontos de ameaça à validade do trabalho:

1. O estudo foi aplicado em uma única empresa. A proposta do estudo era validar a existência de dependências em um projeto real e identificar quais os tipos de dependência poderíamos encontrar. Por se tratar de um estudo preliminar, o escopo do estudo ficou limitado na avaliação de tarefas de apenas um projeto.
2. No estudo consultamos apenas um especialista para identificação das dependências. Uma abordagem mais precisa consistiria em envolver toda a equipe. Como esse estudo foi para uma abordagem preliminar,

consideramos isso como uma ameaça à validade e não como uma circunstância que torna o estudo impraticável.

4.5 Considerações sobre o Capítulo

Este capítulo apresentou o estudo exploratório realizado para fundamentar a motivação dessa dissertação.

A partir dos resultados deste estudo foi possível identificar na prática respostas para a fundamentação do trabalho. As informações foram obtidas analisando tarefas de 3 iterações de um projeto real em uma organização de desenvolvimento de software. Por se tratar de um estudo preliminar analisamos dados de um único projeto.

Das tarefas analisadas, conseguimos observar alguns pontos, que reforçam as argumentações dos trabalhos relacionados (capítulo 3), como.

- As dependências entre as tarefas existem mesmo que a organização tenha um processo de desenvolvimento de software simples (Cohn, 2005) (Strode, 2016) (Bick et al., 2018);
- A forma de gestão dos métodos ágeis não prioriza o tratamento das dependências que existem entre as tarefas, mesmo reconhecendo que isso pode ser um motivo de falha nos projetos (Cohn, 2005) (Strode, 2016) (Bick et al., 2018);
- No estudo, a quantidade de dependências identificadas entre as tarefas superaram em 104% a 120% a quantidade de tarefas nas iterações analisadas. Isso mostra que a gestão das dependências pode não ser tão trivial sem um apoio computacional.
- Realizar a análise do caminho crítico durante o planejamento da iteração e ao longo da execução da Sprint poderia ajudar a equipe a priorizar as tarefas que podem impactar a entrega final da iteração. Para validar essa constatação, desenvolvemos estudos mais detalhados que serão apresentados nos próximos capítulos.

No próximo capítulo, será apresentada a ferramenta *AgileCritPath* desenvolvida para apoiar na validação da proposta dessa pesquisa.

5. A Ferramenta *AgileCriticalPath*

O objetivo deste Capítulo é apresentar a ferramenta desenvolvida para oferecer mecanismos de suporte aos problemas apresentados no Capítulo 1.

5.1. Introdução

Métodos ágeis foram projetados para apoiar o desenvolvimento em ambientes instáveis, e de certa forma, eles fornecem soluções para esses problemas (Pries-Heje and Pries-Heje, 2011) (Strode, 2016), porém ainda existem processos organizacionais que precisam ser redefinidos (Bick et al., 2018) para que seja possível uma gestão mais efetiva das demandas.

Abordagens ágeis baseiam-se na ideia de que os desenvolvedores podem se auto-organizar e trabalhar de forma colaborativa (Bick et al., 2018), porém alguns estudos sugerem que projetos grandes e complexos de desenvolvimento de software podem realmente se beneficiar da combinação da flexibilidade inerente ao trabalho em equipe ágil e modelos que apoiem uma estrutura orientada a planos (Barlow et al., 2011), (Cao et al., 2004), (Ramasubbu et al., 2015).

Bick et al., (2018) e Badampudi et al., (2013) defendem abordagem híbrida onde a organização mantém o uso de métodos ágeis, incluindo práticas já consolidadas de metodologias tradicionais que garantem mais previsibilidade, confiabilidade, estabilidade e uso mais efetivo de recursos.

Com a nossa experiência no mercado de engenharia de software conseguimos observar que o desenvolvimento de sistemas de software requer mais que as metodologias podem oferecer, principalmente quando o tamanho e a quantidade das equipes aumentam, quando o trabalho é planejado em um ambiente com constantes mudanças e que exige respostas gerenciais rápidas.

5.2. O Suporte ao Método do Caminho Crítico em um ambiente ágil

Para apoiar a equipe na análise do Caminho Crítico foi desenvolvida a ferramenta *AgileCriticalPath*. O objetivo da ferramenta é ser utilizada em ambientes ágeis onde o trabalho é dividido em tarefas.

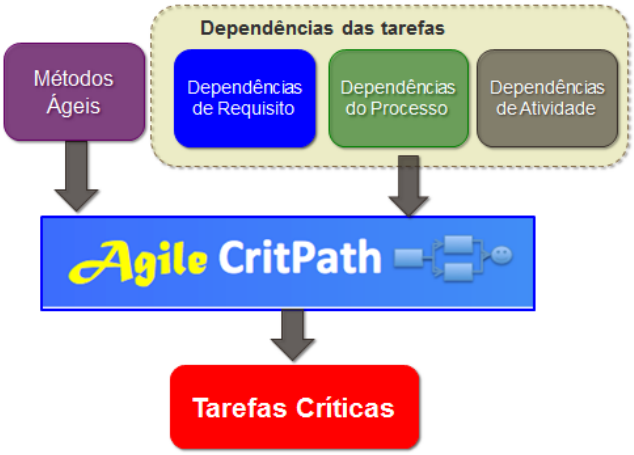


Figura 5.1: Modelo *AgileCritPath*

O Caminho Crítico é calculado com as informações das tarefas planejadas para a equipe, ou tarefas do *Backlog* da iteração. Para obter as informações das tarefas, a ferramenta *AgileCritPath* se conecta ao sistema de Gestão de Tarefas (*Task Manager)*, conforme Figura 5.2.

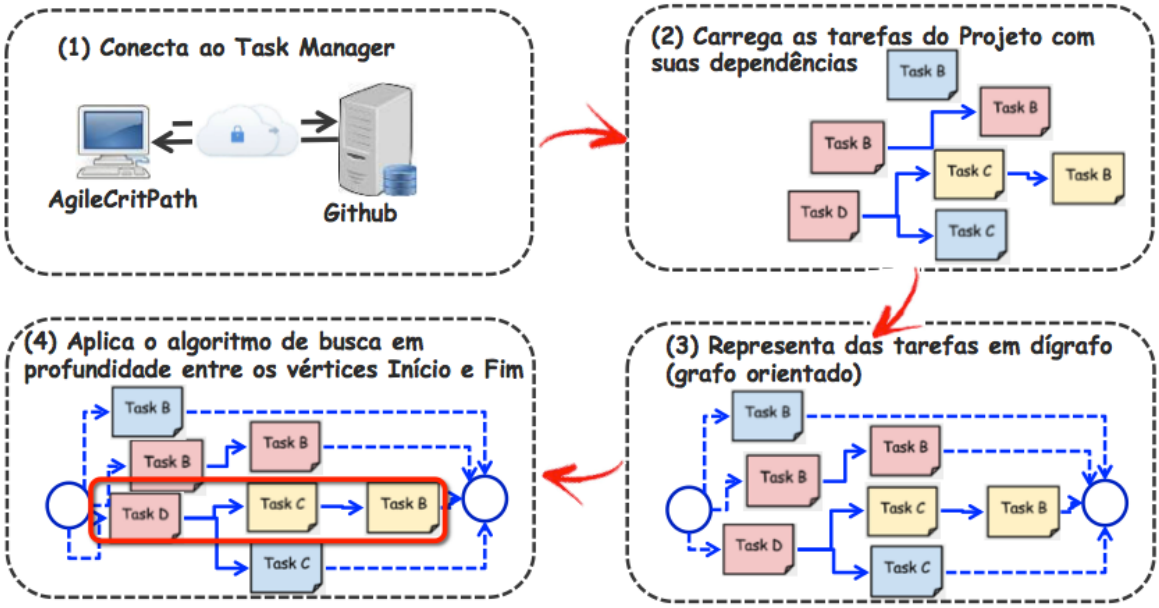


Figura 5.2: Identificando Tarefas Críticas na ferramenta AgileCritPath

Através da conexão com o sistema de Gestão de Tarefas são extraídas as seguintes informações: (1) ID da tarefa, (2) Nome da Tarefa, (3) Tempo estimado, (4) Tempo gasto na tarefa, (5) Pessoa associada à tarefa e (6) ID das tarefas dependentes. Para identificar as tarefas críticas montamos um grafo orientado. Ao grafo resultante é aplicado o algoritmo de busca em profundidade (Rhee et al., 1994) para encontrar todos os caminhos existentes e identificar o maior caminho ou o caminho crítico. As tarefas que compõem o maior caminho serão as tarefas críticas.

A Figura 5.3 mostra um exemplo de grafo criado a partir das informações das tarefas. As linhas sólidas representam as dependências entre as tarefas, as linhas pontilhadas são criadas para associar as tarefas ao nó de início ou fim do grafo.

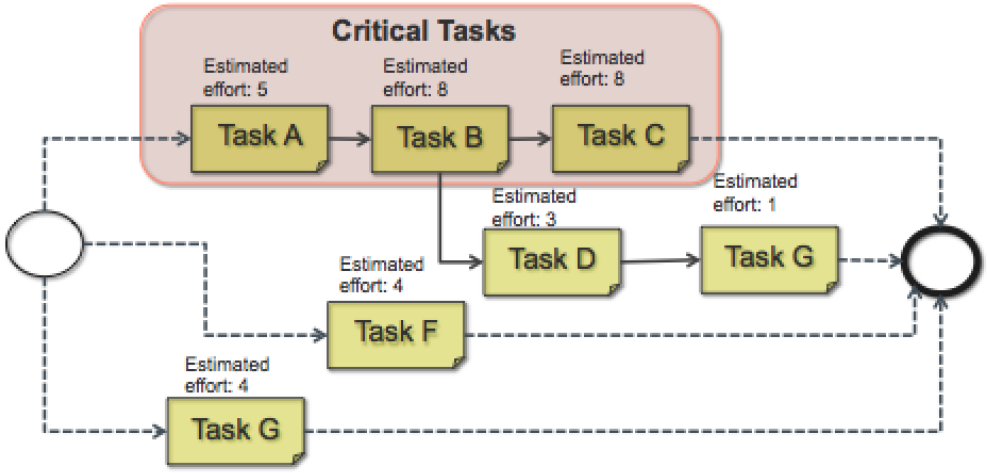


Figura 5.3: Rede de Tarefas

A aplicação foi desenvolvida seguindo a seguinte arquitetura, conforme figura 5.4:

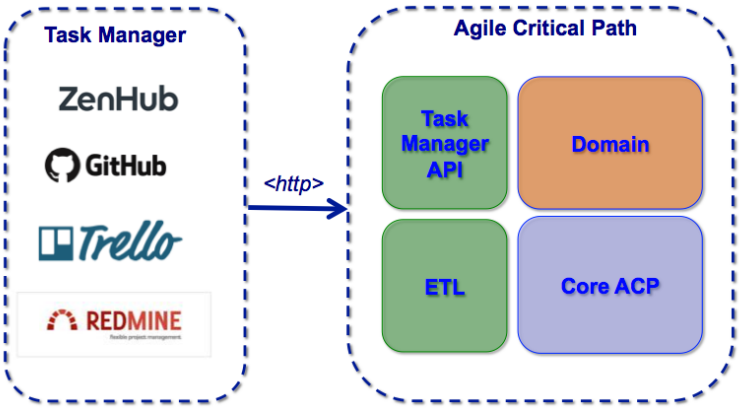


Figura 5.4: Arquitetura da ferramenta

O módulo ETL (*Extract, Transform and Load*) é responsável por extrair, transformar os dados e carregar os objetos conforme o modelo de dados, Figura 5.5. O módulo Core ACP gera o grafo de tarefas e encontra os caminhos existentes. Todos os caminhos encontrados são exibidos em ordem decrescente, mostrando do maior caminho para o menor, conforme o esforço calculado de cada caminho.

O tamanho do caminho é calculado pela soma dos esforços planejados reportados nas tarefas que fazem parte do caminho. Caso a tarefa tenha sido concluída, consideramos o esforço realizado.

A aplicação Web funciona de forma autômoma e se conecta ao Task Manager através das integrações (Task Manager API). A versão atual da ferramenta possui integração com o *GitHub*, ZenHub e Redmine.

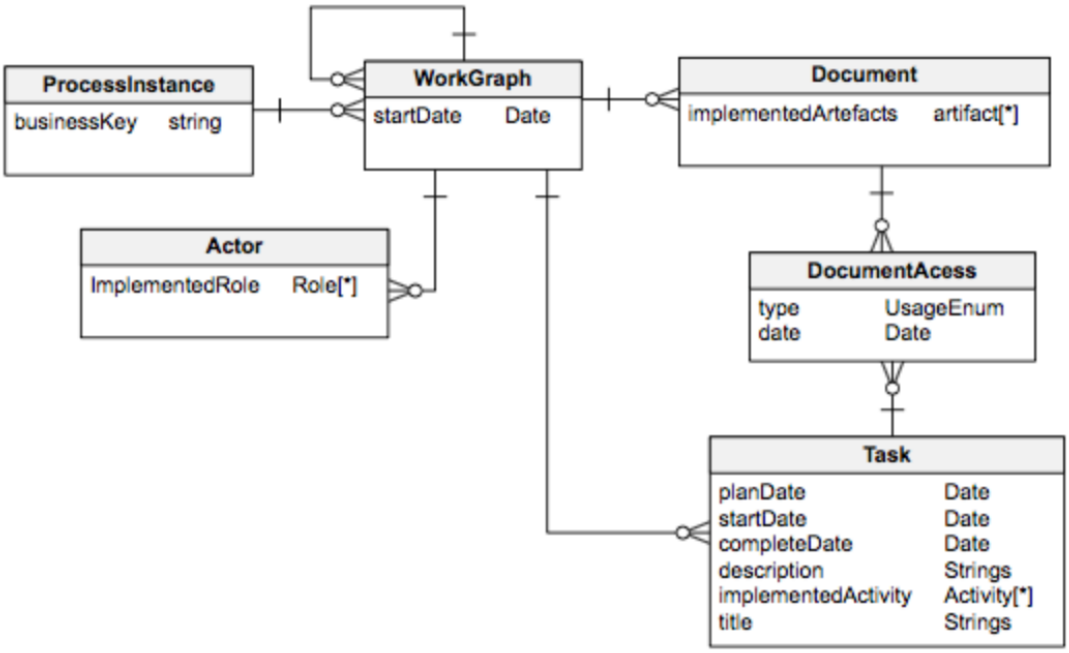


Figura 5.5: Modelo de Dados

O modelo de domínio da aplicação detalhado na Figura 5.5 é o modelo de domínio utilizado no grupo de pesquisa Prisma, o que facilita a reutilização do código em outras soluções. O modelo de domínio é público e encontra-se disponível do *GitHub*: https://github.com/utelemaco/prisma-kip-domain.

A ferramenta foi desenvolvida para ser utilizada por qualquer pessoa da equipe. O usuário poderá logar na aplicação. Na tela inicial, o usuário terá a opção de se conectar ao *GitHub* ou *Redmine* (Figura 5.6).

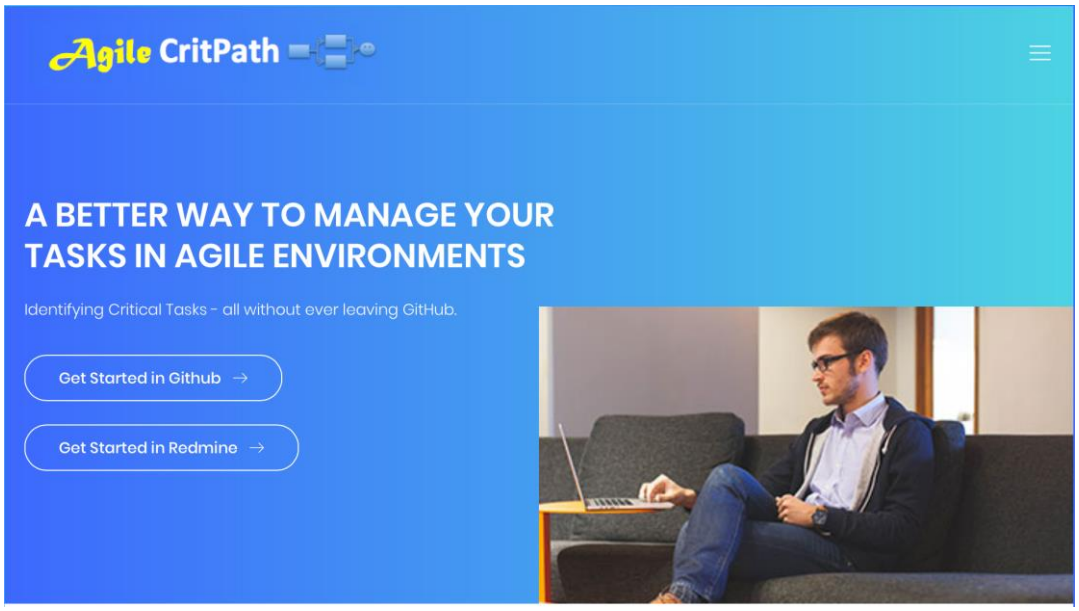


Figura 5.6: Tela inicial da ferramenta *AgileCritPath*

Ao selecionar a opção “Iniciar com *GitHub*”, por exemplo, o sistema apresenta um formulário onde o usuário pode consultar um repositório previamente cadastrado ou informar dados de um novo repositório, Figura 5.7. O usuário deve ter acesso ao repositório pelo *GitHub* para conseguir realizar a consulta.

A ferramenta *AgileCritPath* consulta todas as tarefas do repositório ou pesquisa somente as tarefas de uma iteração. Com a lista de tarefas e dependências, o sistema exibe os caminhos encontrados na rede de tarefas.

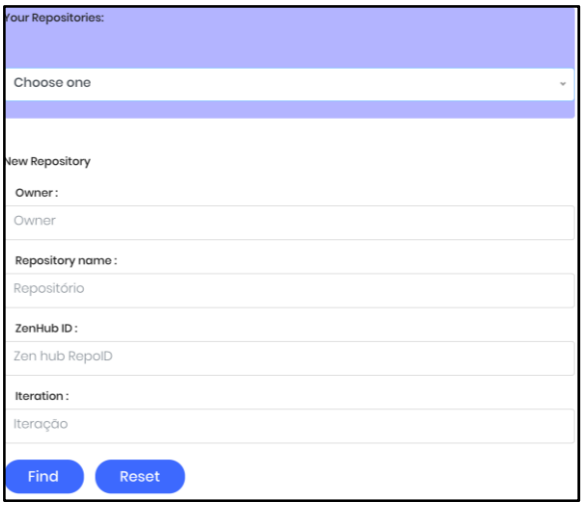


Figura 5.7: Dados de entrada do repositório do *GitHub* a ser consultado

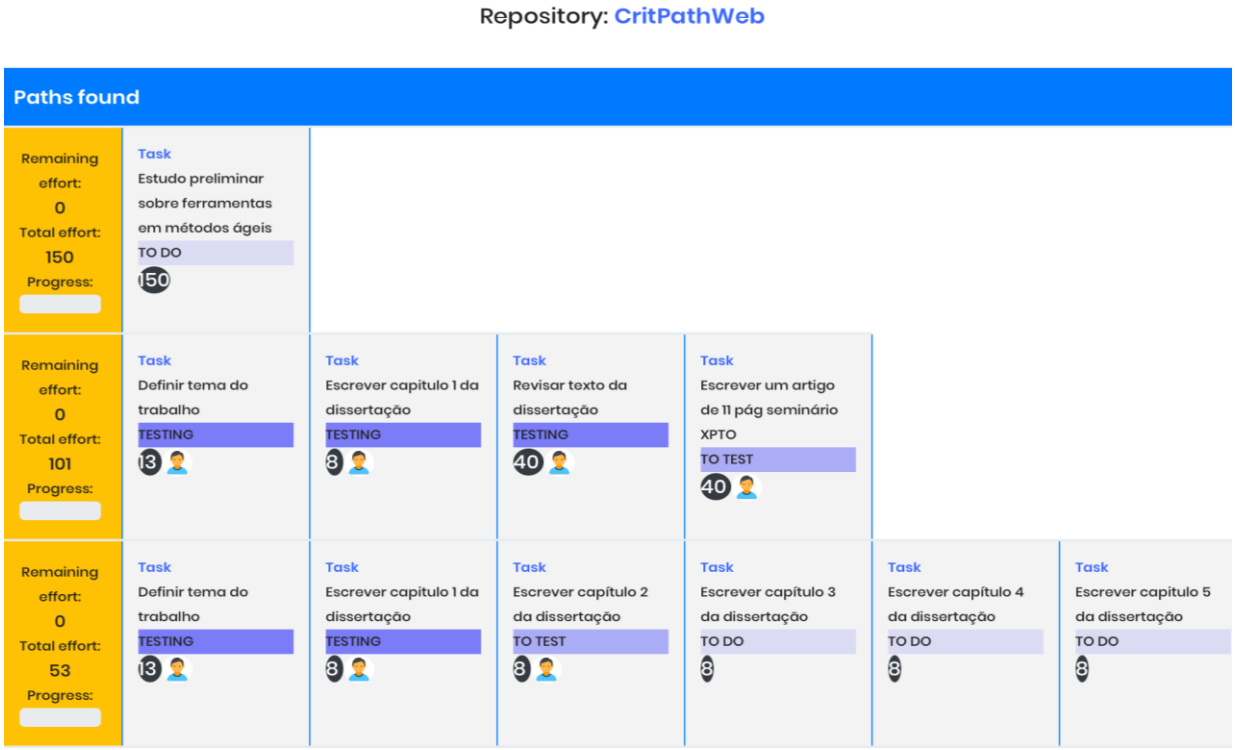


Figure 5.8: Apresentação do resultado

A Figura 5.8 mostra o resultado da consulta dos caminhos de uma rede de tarefas de um repositório do *GitHub*.

Os caminhos encontrados são exibidos na ordem decrescente, de forma que o Caminho Crítico seja exibido no topo da lista. Na ferramenta são exibidos todos os caminhos encontrados para que a equipe possa acompanhar também os caminhos que têm risco de se tornar críticos, caminhos "*quase críticos*". Junto à listagem dos caminhos é exibido o status da tarefa, o responsável pela a execução da tarefa e o esforço de cada tarefa que compõe o caminho (ver Figura 5.8).

O planejamento no ambiente ágil é diferente das abordagens usadas em modelos tradicionais de desenvolvimento de software orientados por planos (Badampudi et al., 2013). Em vez de empregar planos em projetos baseados em um conjunto de fatores e restrições predefinidos, os modelos ágeis contam com fator humano para se auto-organizarem. Nesse trabalho buscamos oferecer um apoio sistêmico para que as informações estejam de forma acessível para todos da equipe e que as decisões sobre a ordem de execução das tarefas possam ser tomadas a qualquer momento.

Em um planejamento ágil, as tarefas podem estar estimadas em tamanho relativo. Logo, se as tarefas estiverem estimadas em esforço, os caminhos encontrados da rede de tarefas irão representar o maior esforço e não estarão relacionados com unidades de tempo, como horas ou dias.

5.3. Limitações

Quanto ao desenvolvimento da ferramenta, identificamos as seguintes limitações:

- A ferramenta foi desenvolvida considerando apenas o relacionamento Término-Início (TI). Apesar de esse relacionamento ser o mais utilizado, acreditamos que tratar os demais relacionamentos daria mais completude de planejamento;
- A ferramenta trata apenas dependências entre tarefas. Durante o estudo de Mapeamento Sistemático verificamos que existem outros tipos de dependências das quais não estamos tratando.
- Algumas informações obtidas através do Método de Caminho Crítico em projetos tradicionais não foram consideradas na aplicação como: cálculo de folgas, antecipações e esperas. Em ambientes ágeis, nem sempre as tarefas estão estimadas em dias ou horas. Para preservar as características dos processos de cada organização estamos considerando que as estimativas

poderão ter valores relativos e estes valores podem estar relacionados à complexidade e/ou tempo.

5.4. Considerações sobre o Capítulo

O Método do Caminho Crítico é uma técnica já consolidada em modelos de gestão de projetos tradicionais. A proposta desse trabalho é utilizar práticas do método em um ambiente ágil.

Neste capítulo apresentamos a ferramenta *AgileCritPath* desenvolvida para concretizar a ideia de aplicarmos o uso do Método do Caminho Crítico em projetos de desenvolvimento de software em organizações.

A ferramenta foi desenvolvida em formato *OpenSource* e seu código fonte está disponível no *GitHub*: https://github.com/RachelVital/CritPath.

Neste capítulo apresentamos a arquitetura e forma de uso da ferramenta. Para a validação da ferramenta, aplicamos o modelo de aceitação tecnológica TAM (*Technology Acceptance Model*) que será apresentado no próximo Capítulo.

6. Avaliação da ferramenta *AgileCriticalPath*

Neste capítulo será apresentada a avaliação da ferramenta AgileCriticalPath retratada no capítulo anterior. Na avaliação buscamos analisar a eficiência e a adequação da tecnologia em um ambiente industrial utilizando o Modelo de Aceitação de Tecnologia (TAM).

6.1. Introdução

Esse estudo foi realizado para validarmos a aceitação da ferramenta *AgileCritPath* em uma empresa de desenvolvimento de software. O objetivo principal desse estudo é analisar a eficiência e adequação técnica da ferramenta através do Modelo de Aceitação de Tecnologia (TAM) (Lee et al., 2003). O modelo TAM foi utilizado nesta pesquisa devido ao aporte teórico e empírico que vem recebendo em suas aplicações (Saga and Zmud, 1994). Na aplicação do modelo buscamos obter feedbacks de potenciais usuários e incentivar a adoção da tecnologia em empresas de desenvolvimento de software.

Pesquisas associadas à adoção de tecnologias e a avaliação dos seus impactos nas atividades são avaliações importantes na área de engenharia de software. O Modelo de Aceitação de Tecnologia TAM foi proposto por Davis em 1989 e sugere a aceitação de uma nova tecnologia de TI dependendo de duas variáveis: (1) Facilidade de Uso Percebida e a (2) Utilidade Percebida. Para Davis, as pessoas tendem a usar ou não uma tecnologia com o objetivo de melhorar seu desempenho no trabalho – utilidade percebida. Porém, mesmo que essa pessoa entenda que uma determinada tecnologia é útil, sua utilização poderá ser prejudicada se o uso for muito complicado, de modo que o esforço não compense o uso – facilidade percebida.

6.2. Objetivos e indicadores

O principal objetivo da aplicação do modelo TAM foi avaliar a aceitação e adequação da utilização de conceitos do Método do Caminho Crítico em projetos ágeis na indústria. Para organização do objetivo desse estudo, utilizamos o paradigma GQM (*Goal-Question-Metric*) (Solingen et al., 2002):

O objetivo deste estudo foi:

Analisar a ferramenta *AgileCriticalPath*

Com o propósito de caracterizar

Com respeito à facilidade de uso e a utilidade percebida

Do ponto de vista de profissionais da prática

No contexto do planejamento e acompanhamento das tarefas de projetos ágeis em uma empresa de desenvolvimento de software

Para alcançar esse objetivo operacionalizamos os indicadores:

(1) **Facilidade de Uso Percebida – *Perceived ease of use***: o grau que a pessoa acredita que utilizando a tecnologia específica facilita o seu trabalho.

(2) **Utilidade percebida – *Perceived usefulness***: o grau que a pessoa acredita que a tecnologia específica melhora seu desempenho no trabalho.

Dessa forma, para esse estudo de caso foi criado um questionário pós-uso da aplicação com escalas de 6 pontos, tendo como base os questionários aplicados por (Laitenberger and Dreyer, 1998) e (Denger et al., 2004). Nas questões, utilizamos uma escala Likert de 6 pontos: (1) Discordo Totalmente, (2) Discordo Amplamente, (3) Discordo Parcialmente, (4) Concordo Parcialmente, (5) Concordo Amplamente, (6) Concordo Totalmente. Para as questões associadas ao indicador Facilidade de Uso Percebida em Q6 e Q7 utilizamos as escalas de 1 a 6 sendo 1 “Muito Difícil” e 6 “Muito Fácil”. Não foi utilizada uma escala de sete pontos contendo um valor neutro ou intermediário, pois, segundo Laitenberger and Dreyer (1998) este valor neutro não fornece informações sobre para qual direção o participante está inclinado.

Os aspectos avaliados estão exibidos na Tabela 6.1.

Tabela 6.1: Avaliação realizada através do modelo de aceitação TAM

UTILIDADE PERCEBIDA	Q1. De 1 a 6, sendo 1 “Discordo Totalmente” e 6 “Concordo Totalmente”. O uso da ferramenta *AgileCritPath* aumenta a produtividade da equipe
	Q2. De 1 a 6, sendo 1 “Discordo Totalmente” e 6 “Concordo Totalmente”. A utilidade da ferramenta *AgileCritPath* aumenta o desempenho da equipe

	Q3. De 1 a 6, sendo 1 “Discordo Totalmente” e 6 “Concordo Totalmente”. O uso da ferramenta *AgileCritPath* aumenta a qualidade do meu trabalho e do trabalho da minha equipe.
	Q4. De 1 a 6, sendo 1 “Discordo Totalmente” e 6 “Concordo Totalmente”. As vantagens de usar a ferramenta superam as desvantagens
	Q5. De 1 a 6, sendo 1 “Discordo Totalmente” e 6 “Concordo Totalmente”. A tecnologia é útil no meu trabalho
FACILIDADE DE USO PERCEBIDA	Q6. De 1 a 6, sendo 1 “muito difícil” e 6 “muito fácil”, avalie a dificuldade em identificar as dependências das tarefas da iteração
	Q7. De 1 a 6, sendo 1 “muito difícil” e 6 “muito fácil”, avalie o uso da ferramenta AgileCriticalPath na identificação das atividades críticas da iteração
	Q8. De 1 a 6, sendo 1 “Discordo totalmente” e 6 “Concordo Totalmente”, A facilidade do uso da ferramenta permite melhorar o desempenho da equipe durante o planejamento e execução das tarefas da iteração?
	Q9. De 1 a 6, sendo 1 “Discordo totalmente” e 6 “Concordo Totalmente”, considero utilizar a ferramenta *AgileCriticalPath* no planejamento e execução da iteração?
EXTRAS	Q10. Você identifica benefícios no uso da ferramenta na sua equipe e/ou organização?
	Q11. Você identifica limitações no uso da ferramenta?

Incluímos no questionário questões abertas com o objetivo de coletar informações sobre os benefícios e as limitações observadas pelos participantes, nas questões Q10 e Q11.

Esse estudo foi dividido nas seguintes etapas:

Figura 6.1: Etapas do estudo de aplicação do modelo de aceitação TAM

6.3. Planejamento

A fase de Planejamento foi iniciada em Agosto/2018. Durante esse período iniciamos estudos para a aplicação do Modelo de Aceitação de Tecnologia TAM durante a fase de validação da ferramenta proposta.

Na fase de planejamento realizamos as atividades de seleção da organização e indivíduos, definição dos procedimentos do estudo e definições de ferramentas que seriam utilizadas. Definição do Formulário de Consentimento e Desimpedimento de Participação (apêndice D), Formulário de Caracterização dos Participantes (apêndice E), Avaliação do modelo TAM (tabela 3.3) e definimos os procedimentos para o estudo.

Avaliação do modelo TAM foi elaborada com questões direcionadas para medir as variáveis: (1) Facilidade de Uso Percebida e a (2) Utilidade Percebida conforme sugerida pelo modelo. As questões detalhadas a tabela 6.1 estão divididas da seguinte forma:

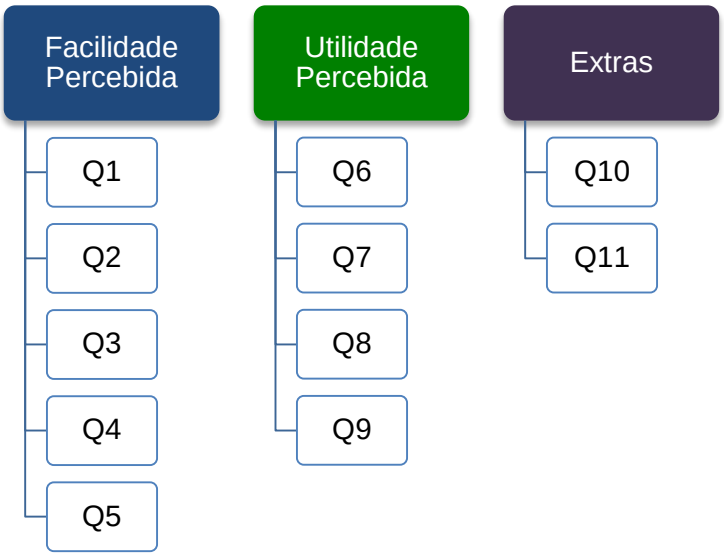


Figura 6.2: Distribuição das Perguntas por objetivos

Conforme Figura 6.2, o indicador de Facilidade Percebida será avaliado nas questões Q1 a Q5. A Utilidade Percebida será avaliada nas questões Q6 até Q9. No formulário de avaliação incluímos duas questões abertas para coletarmos informações sobre os benefícios e limitações da ferramenta observando ponto de vista dos participantes.

6.4. Estudo Piloto

Para garantir a validade do estudo aplicamos o modelo definido como Piloto com os alunos da disciplina de Qualidade de Software entre os meses de setembro/2018 a novembro/2018. Para a aplicação do Piloto na disciplina executamos os seguintes passos:

a. **Seleção dos indivíduos** – os participantes do estudo piloto foram os alunos da disciplina de Qualidade de software ministrada pelo Prof. Toacy no curso de Bacharelado em Ciências da Computação da UFRJ.
b. **Preparação** – durante a disciplina os alunos tiveram aulas sobre assuntos relevantes para a execução dos projetos:
   - Processos de Software (Toacy) 06 e 08.08
   - MPS/CMMI (Toacy) 06 a 08.08
   - Métodos Ágeis (Ulisses) 13.08
   - Scrub e BPMN/CMMN (Toacy) 15.08
   - Explicar o trabalho e as ferramentas (Ulisses) 20.08
   - Planejamento de Projetos Ágeis (Rachel) 20.08

- Regras em Projetos de Software (Ulisses) 22.08
- Gestão de Conhecimento em Projetos de Software (Gláucia) 27.08
- Qualidade de Código (Sonarqube, etc) (Ulisses) 29.08
- JHipster (Ulisses) 03.09

c. **Definição do ambiente e ferramentas utilizadas** - Para a realização do estudo durante a disciplina definimos o ambiente e ferramentas que os alunos iriam utilizar em seus projetos. O *GitHub* foi utilizado com a plataforma para controle de versões e publicação dos projetos. O *ZenHub* como uma solução para visualização das tarefas no quadro *Kanban* e as dependências entre as tarefas seriam inseridas no *Zenhub*. Além disso, foi utilizada a ferramenta *AgileCriticalPath* para identificação do caminho crítico ao longo da iteração.

d. **Execução** – O projeto desenvolvido na disciplina foi dividido em 3 iterações. As equipes receberam instruções do modelo do Processo de Desenvolvimento de Software a ser utilizado e o modelo do Processo de Negócio. Com essas informações eles conseguiram definir a maior parte das dependências que existem entre as tarefas. O piloto aconteceu para validarmos o algoritmo utilizado para calcular os caminhos da rede de tarefas.

e. **Aprendizado** – O estudo Piloto ajudou na validação da ferramenta *AgileCritPath*. Por conta da inexperiência dos participantes não achamos viável aplicar os formulários de avaliação definidos para a aplicação do modelo de aceitação TAM.

6.5. Procedimentos da avaliação

Para a execução do modelo de aceitação, definimos alguns procedimentos:

a. **Seleção da empresa e projetos** – a empresa selecionada para aplicação do estudo foi uma empresa de desenvolvimento de software. A empresa atua na àrea de engenharia de software há 20 anos e seu processo de desenvolvimento de software é baseado em práticas ágeis. O projeto utilizado na validação do estudo foi selecionado pela própria organização.

b. **Preparação do estudo** – A empresa selecionada utiliza o Redmine como ferramenta de gestão de tarefas. Para a execução do estudo

desenvolvemos na ferramenta *AgileCritPath* a integração com o Redmine.

6.6. Execução do estudo

A execução do estudo ocorreu em duas etapas nos meses de novembro e dezembro de 2018.

A execução do estudo ocorreu *in vivo*. Estudos *in vivo* são estudos que envolvem pessoas em seu próprio ambiente de trabalho em condições realistas (Travassos and Barros, 2003). Estudos de caso feitos em ambiente industrial é um tipo importante de estudo *in vivo*, visto que estes permitem a análise de um processo particular no contexto de um ciclo de vida de software (Shull et al., 2001).

Para garantir que o estudo não impactasse o processo de desenvolvimento de software observamos as tarefas realizadas no dia a dia da equipe e incluímos as atividades necessárias para a utilização da ferramenta, conforme Figura 6.3.

Figura 6.3: Atividades realizadas durante a Iteração

Na Figura 6.3 mostra as atividades já realizadas na organização. As atividades em vermelho são as atividades que incluímos para que os conceitos do Método do Caminho Crítico pudessem ser utilizados na organização.

As atividades estão divididas em dois momentos distintos: a reunião de Planejamento e nas reuniões diárias (Daily Meetings). Na reunião de planejamento da iteração, a equipe passou a incluir as dependências das tarefas. Durante a execução

da iteração, a equipe realiza as reuniões diárias (*daily Scrum*). Assim como sugerido no Scrum(Deemer et al., 2012), as reuniões diárias tem duração de 15 minutos, onde a equipe discute sobre o que foi feito nas últimas 24 horas, o plano para as próximas 24 horas e quais impedimentos podem ocorrer. A reunião é realizada na frente do quadro *Kanban*. A visualização dos caminhos calculados pela ferramenta *AgileCriticalPath* passou a ser verificado durante as reuniões diárias.

Durante a execução do estudo conseguimos obter alguns *insigths* dos participantes que serão apresentados na análise dos resultados.

Ao final da iteração, os participantes preencheram o Formulário de Consentimento e Desimpedimento de Participação (apêndice D), Formulário de Caracterização dos Participantes (apêndice E) e a Avaliação do modelo TAM (tabela 3.3). Os formulários foram preenchidos individualmente e sem qualquer contato entre os participantes, como esperado pelo Termo de Consentimento no primeiro apêndice, assinados por todos. Assim, procura-se trazer maior confiabilidade no resultado obtido e, também, diminuir o risco de viés nos dados da pesquisa.

6.7. Análise e Interpretação dos Resultados

Participantes

O universo da pesquisa foi uma equipe com seis funcionários de uma empresa de desenvolvimento de software. A participação na pesquisa foi livre e voluntária. O questionário foi enviado a todos e tivemos um retorno de quatro respostas.

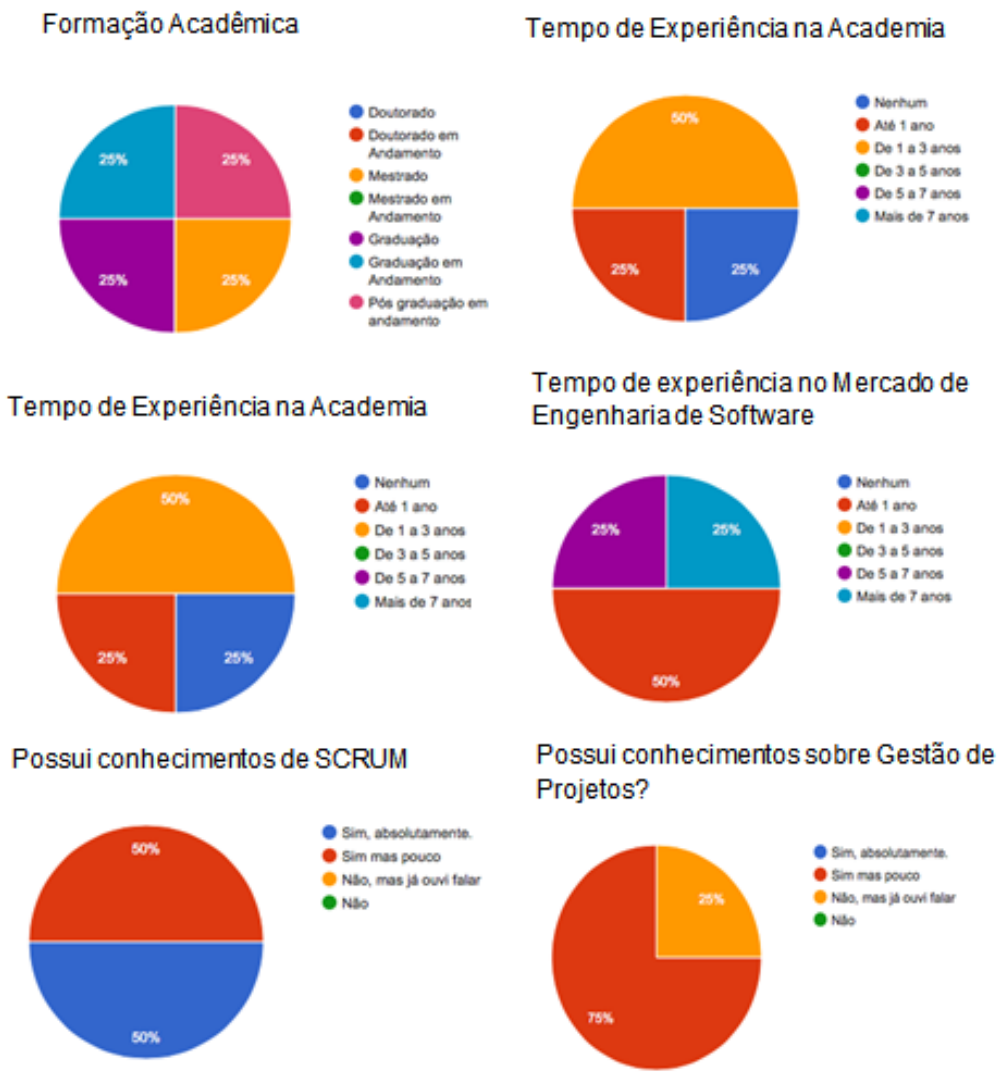


Figura 6.4: Caracterização dos Participantes

Conforme Figura 6.4 os participantes, em geral, possuem graduação completa, sendo 1 participante com a graduação em andamento. O tempo de experiência da academia varia bastante entre o grupo pesquisado, já que a equipe é formada por profissionais: sêniores (2), junior (1) e estagiário (1). Todos os participantes conhecem o básico de gestão de projetos e possuem conhecimentos sobre Scrum.

Procedimento de coleta das informações

Durante o planejamento e execução da iteração os participantes tiveram acesso à ferramenta e nas reuniões diárias foram apresentadas para a equipe as tarefas que faziam parte do caminho crítico. A iteração durou 2 semanas e no final os participantes foram solicitados a responder as perguntas sobre a avaliação da ferramenta.

Os formulários foram divulgados pela Internet no ambiente Google Form, o prazo de resposta dos formulários foram de 7 dias e aconteceu em janeiro/2019. Nesse período foi enviado um lembrete convidando-os para responder a pesquisa, antes de encerrar a coleta dos dados.

Resultados

Os resultados da avaliação estão apresentados na tabela 6.2.

Tabela 6.2: Resultado da Avaliação

		(1) Discordo Totalmente	(2) Discordo Amplamente	(3) Discordo Parcialmente	(4) Concordo Parcialmente	(5) Concordo Amplamente	(6) Concordo Totalmente
UTILIDADE PERCEBIDA	Q1. O uso da ferramenta *AgileCritPath* aumenta a produtividade da equipe				**75%**		**25%**
	Q2. O uso da ferramenta *AgileCritPath* aumenta o desempenho da equipe			**25%**	**50%**		**25%**
	Q3. O uso da ferramenta *AgileCritPath* aumenta a qualidade do meu trabalho e do trabalho da minha equipe.				**50%**	**25%**	**25%**
	Q4. As vantagens de usar a ferramenta superam as desvantagens				**25%**	**25%**	**50%**
	Q5. A tecnologia é				**25%**		**75%**

	útil no meu trabalho						
FACILIDADE DE USO PERCEBIDA	Q6. De 1 a 6, sendo 1 “muito difícil” e 6 “muito fácil”, avalie a dificuldade em identificar as dependências das tarefas da iteração			**50%**	**25%**		**25%**
	Q7. De 1 a 6, sendo 1 “muito difícil” e 6 “muito fácil”, avalie o uso da ferramenta AgileCriticalPath na identificação das atividades críticas da iteração			**50%**	**25%**		**25%**
	Q8. O uso da ferramenta permite melhorar o desempenho da equipe durante o planejamento e execução das tarefas da iteração?				**50%**		**50%**
	Q9. Considero utilizar a ferramenta *AgileCriticalPath* no planejamento e execução da iteração?			**25%**	**25%**		**50%**

Quanto ao critério de avaliação da Utilidade Percebida, os participantes concordaram que o uso da ferramenta poderia melhorar a produtividade e qualidade dos trabalhos dos membros da equipe. Todos viram a ferramenta como algo útil para a execução das atividades.

Quanto à Facilidade Percebida de Uso, a equipe avaliou a identificação das

dependências entre as tarefas como uma atividade razoavelmente difícil com 50% e a identificação das tarefas do Caminho Crítico também como uma atividade razoavelmente difícil. Mesmo apontando dificuldades na identificação das dependências e avaliação do caminho crítico, ainda sim, 75% das respostas consideram utilizar a ferramenta durante o planejamento e execução das tarefas. Apesar de ter um esforço um maior na parte de identificação das dependências, a equipe reconheceu os benefícios que o uso da ferramenta pôde proporcionar.

Nas duas questões descritivas (Q10 e Q11), os participantes identificaram os benefícios e limitações da ferramenta. Os principais benefícios destacados foram: a visualização das dependências entre as tarefas como sendo um facilitador para identificar quais tarefas deve ser priorizada, além de melhorar a visualização do esforço total planejado para concluir as tarefas do caminho crítico. Como limitação da ferramenta: foi colocado que a ferramenta poderia exibir informações do estimado x realizado das tarefas em andamento e poderia informar mais métricas para acompanhamento do progresso do trabalho.

Durante o uso da ferramenta nas reuniões diárias conseguimos capturar observações realizadas pela equipe como:

- A equipe observou que o uso da ferramenta permite que todos da equipe possam ter a visão de quais tarefas são críticas;
- As dependências de Conhecimento e Recurso existem no ambiente estudado, porém, não foi explicitada pela equipe durante o planejamento.
- A equipe achou interessante deixar as tarefas críticas com os desenvolvedores mais experientes no projeto, principalmente quando os prazos de execução dessas tarefas estiverem apertados;
- Visualizar o caminho crítico também é uma forma de avaliar os pontos mais complexos do projeto, já que as estimativas consideram complexidade;
- A equipe observou que a prioridade que eles classificaram no momento do planejamento das tarefas não fazia sentido, em alguns casos, eles poderiam ter tido um ganho maior priorizando as tarefas do caminho crítico;
- A equipe avaliou o Método do Caminho Crítico como um instrumento complementar ao quadro *Kanban*, pois no quadro *Kanban* eles não

conseguiam acompanhar quais eram as dependências entre as tarefas.

6.8. Ameaça à Validade

Nesse estudo conseguimos identificar possíveis pontos de ameaça à validade do estudo:

1. Uma limitação do modelo TAM, segundo Benbasat and Barki (2007) e Chuttur (2009), seria o seu médio alcance, que fornece uma ponte potencialmente útil da aceitação, mas que se limita ao ambiente onde foi aplicado. No caso desse estudo, o modelo foi aplicado em uma única empresa.
2. Aplicar o estudo em mais de um projeto e com mais participantes poderia nos fornecer melhores resultados.
3. A possibilidade dos participantes não entenderem alguma questão também constitui uma ameaça potencial à validade.
4. O pesquisador responsável pelo estudo trabalha na empresa onde o estudo foi aplicado. Consideramos isso como uma ameaça à validade, mas não como uma circunstância que torna o estudo impraticável.
5. Os resultados do estudo também apresentam algumas limitações, incluindo o viés do pesquisador na seleção do estudo e na avaliação da qualidade.

6.9. Considerações sobre o Capítulo

O Modelo de Aceitação de Tecnologia TAM é o modelo amplamente aplicado que prevê a aceitação de uma nova tecnologia pelo usuário. Um critério importante desse estudo foi avaliação quanto à usabilidade e viabilidade da ferramenta *AgileCritPath* na indúsria. Usabilidade é um dos aspectos relacionados à qualidade de uso de sistemas, sendo um dos mais importantes critérios de aceitação para aplicações interativas em geral, e, em particular, para aplicações Web (Insfran and Fernandez, 2008). Devido às características dessas aplicações, a importância da usabilidade é ainda maior, pois “aplicações Web são aplicações interativas, centradas no usuário e baseadas em hipermídia, onde a interface com o usuário desempenha um papel central” (Olsina et al., 2006).

Vários autores destacam que a aceitação de uma tecnologia está relacionada à qualidade e utilização dos sistemas, à qualidade das informações e à satisfação do usuário (Petter et al., 2012). Segundo Torres (2009), pode haver casos onde uma tecnologia é implantada dentro do prazo, com os valores orçados e os critérios de qualidade e objetivo atingidos, mas que não resulta em benefícios para os clientes ou melhora o desempenho da empresa.

O objetivo geral do uso do modelo de aceitação foi avaliar o potencial da ferramenta *AgileCritPath* no ambiente industrial. O sistema *AgileCritPath* permite que organizações utilizem as informações de dependências entre as tarefas para melhorar a priorização das tarefas em ambientes ágeis identificando quais as dependências que podem bloquear a execução das tarefas.

Nesse trabalho os desenvolvedores reconhecem as vantagens de utilizar a ferramenta, porém alguns perceberam dificuldades em identificar as dependências entre as tarefas e visualizar o caminho crítico mesmo com o apoio da ferramenta. Ao mesmo tempo, o autor também reconhece que existe um esforço maior da equipe para identificar as dependências e gerenciá-las, existe também certo nível de desconhecimento no escopo das tarefas o que pode dificultar a identificação de algumas dependências entre as tarefas.

7. Conclusão

Neste capítulo será apresentada a conclusão desta dissertação. Primeiro são apresentadas algumas considerações gerais sobre o trabalho e, em seguida, as principais contribuições da pesquisa. Finalmente, as ameaças à validade e algumas oportunidades de trabalhos futuros são delineadas.

7.1. Introdução

Esta dissertação apresentou a proposta de uso de conceitos do Método do Caminho Crítico em ambientes ágeis. Primeiro, foram discutidos os conceitos centrais dos assuntos relacionados ao trabalho. A partir da experiência da autora na área de gestão de projeto, iniciamos algumas pesquisas que nos mostraram a evidência do problema que as organizações enfrentam ao gerir as tarefas em projeto de software que utilizam métodos ágeis.

Durante a pesquisa, conseguimos identificar alguns trabalhos relacionados que reforçam a importância do tema na academia e na indústria. A partir dos trabalhos relacionados vimos os potenciais benefícios apresentados na proposta da dissertação. Para sistematizar a ideia central de gerenciar as dependências entre as tarefas, desenvolvemos a aplicação *AgileCritPath* que funciona de forma autônoma podendo ser integrada a qualquer software de gestão de tarefas.

A aplicação *AgileCritPath* é uma ferramenta web desenvolvida em Java utilizando o framework Spring Boot. A ferramenta foi implementada de forma modularizada e atualmente possui integração ao *GitHub* e a ferramenta permite importar arquivos do Redmine. A proposta principal da ferramenta *AgileCritPath* é fornecer informações de dependências entre as tarefas sem alterar o processo de desenvolvimento da organização. Nossa proposta é oferecer uma solução que seja aderente a qualquer metodologia ágil e que apoie equipes nas tomadas de decisões de forma fácil e rápida. A ferramenta foi desenvolvida em formato de código fonte aberto e está disponível no *GitHub*.

A avaliação do uso da ferramenta ocorreu através do uso do modelo de aceitação tecnológica TAM. Durante avaliação conseguimos também obter insights sobre o uso da tecnologia e incentivar a adoção da ferramenta na indústria.

7.2. Contribuições

Esta pesquisa de dissertação tem o objetivo de trazer contribuições para a área de Engenharia de Software e especificamente para apoio a processos de desenvolvimento de software em ambiente ágeis.

De acordo com os objetivos definidos na pesquisa, podemos listar os objetivos alcançados, conforme mostrado na Tabela 7.1.

Tabela 7.1: Objetivos Alcançados

Objetivo Geral	Apoiar a equipe de desenvolvimento na identificação das tarefas do caminho crítico		**Dissertação**
Objetivo Específico I	Identificar causas/motivos de falha na gestão de projetos ágeis;		**Revisão da Literatura** **Experiência do autor em Gestão de Projetos**
Objetivo Específico II	Identificar as dependências entre tarefas que impactam o desenvolvimento do projeto em ambientes ágeis;		**Mapeamento Sistemático**
Objetivo Específico III	Avaliar a utilização da análise do caminho crítico como estratégia para identificação de tarefas críticas em ambientes ágeis;		**Estudo de Observação**
Objetivo Específico IV	Propor uma estratégia e ferramenta que permita a equipe de desenvolvimento ter informações sobre as tarefas críticas;		**Ferramenta desenvolvida**  https://github.com/RachelVital/CritPath
Objetivo Específico V	Avaliar a estratégia proposta.		**Avaliação TAM**

Outras contribuições foram realizadas durante o período desta pesquisa como:

- Estágio nas disciplinas de Qualidade nas turmas 2016.3, 2017.3 e 2018.3;
- Artigos pulicados:

  2017 **- Escaping from the Time Box towards Continuous Planning: An Industrial Experience**

  Autores: Breno Bernard Nicolau de França**, Rachel Vital Simões**, Valéria Silva, Guilherme Horta Travassos

  2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering

  2019 - **AGILECRITPATH: Identifying Critical Tasks in Agile Environments**

  Autores: **Rachel Vital Simões**, Glaucia Melo dos Santos, Toacy Oliveira, Paulo Alencar e Don Cowan

  SEKE 2019 – International Conference on Software Engineering and Knowledge Engineering.

7.3. Limitações

As limitações estão discutidas ao longo dos capítulos da dissertação. Elas apresentam ameaças à validade não apenas nos estudos isolados, mas também da dissertação.

7.4. Trabalhos Futuros

Considerando os estudos realizados na dissertação e analisando as conclusões apresentadas em cada capítulo foi possível sugerir algumas oportunidades de trabalhos futuros que serão discutidas nesta subseção:

- **Estudo do Mapeamento Sistemático**: Validar os procedimentos realizados no estudo de Mapeamento Sistemático por um segundo pesquisador, para dessa forma eliminar possíveis vieses da pesquisa.
- **Desenvolvimento da Aplicação**: A aplicação desenvolvida ficou com algumas limitações que podem ser desdobradas em trabalhos futuros como:
  - Preparar a aplicação para tratar os outros tipos de dependências como: Término-Término (TT), Início-Início (II) e Início –Término (IT).
  - Desenvolver a integração da ferramenta *AgileCriticalPath* com os demais sistemas de gestão de tarefas.

- Estender a ferramenta para tratar outros tipos de dependências que existem no processo de desenvolvimento de software.
- Evoluir a ferramenta para que algumas dependências possam ser definidas a partir de algum tipo de automação, por exemplo, extrair as dependências das tarefas a partir do modelo de processo definido na organização.
- Implementar as funcionalidades para autorização e autenticação dos usuários.

- **Modelo de Aceitação da Tecnologia**: identificamos também algumas limitações no Modelo de Aceitação de Tecnologia TAM aplicado nesse trabalho. Essas limitações podem ser desdobradas nos seguintes trabalhos futuros:
  - Sugerimos a aplicação do modelo de aceitação em outras empresas de desenvolvimento de software.
  - No estudo realizado consideramos apenas dependências existentes entre as tarefas da iteração. Esse mesmo estudo pode ser aplicado em ambientes com múltiplos projetos, assim poderíamos tratar também dependências entre projetos.
  - Avaliar o uso da tecnologia em ambientes de desenvolvimento de software Contínuos e em ambientes DevOps.
  - Avaliar a evolução do modelo aplicado na avaliação TAM para o modelo denominado TAM2 (Venkatesh and Davis, 2000), onde é proposta uma nova representação do modelo TAM. Apesar do modelo TAM2 não ter sido utilizado, ele parece ser um bom exemplo de novas possibilidades de composição fatorial a partir do corpo essencial do TAM.
  - Sugerimos também um estudo mais aprofundado de outras teorias para investigação de fenômenos relacionados à aceitação de tecnologia.

7.5. Considerações Finais

Este capítulo resume a pesquisa realizada ao longo desta dissertação, incluindo as contribuições para a indústria e para a comunidade acadêmica, bem como um conjunto de possíveis oportunidades de trabalho futuro para melhorar o conhecimento geral sobre o tema entre os profissionais de engenharia de software.

Através dos estudos realizados, a experiência do pesquisador na área de desenvolvimento de software, e juntamente com o tempo gasto com reuniões no grupo de Processos de Engenharia de Software Prisma levou-nos a concluir que, se as organizações realmente querem diminuir o risco de atraso de suas entregas, são necessárias ferramentas que apoiem o processo de desenvolvimento de software, principalmente, em ambientes ágeis e com constantes mudanças.

Por fim, para garantir a efetividade da proposta apresentada nessa dissertação desenvolvemos uma ferramenta para que pudéssemos avaliar o estudo em cenários reais de desenvolvimento de software. Através das respostas obtidas, nós conseguimos avaliar que apesar do esforço adicional para identificar as dependências, o uso do Método do Caminho Crítico em Ambientes Ágeis se mostrou viável na organização avaliada.

REFERÊNCIAS BIBLIOGRÁFICAS

Ahuja, H.N., Dozzi, S.P., Abourizk, S.M., 1994. AOA Crtitical Path Analysis In Hira N. Ahuja, **Project Management Techniques in Planning and Controlling Construction Projects**, 2 ed., chapter 6, New York, USA, John Wiley & Sons, INC.

Alaidaros, H., Omar, M., Romli, R., 2018. Identification of criteria affecting software project monitoring task of Agile Kanban method, **AIP Conference Proceedings 2016**. v. 2016 Issue 1 AIP Publishing, p. 020021.

Amrit, C., Van Hillegersberg, J., 2008. Detecting coordination problems in collaborative software development environments, **Information Systems Management**, v. 25, pp. 57–70.

Anderson, D.J., 2003. **Agile management for software engineering: Applying the theory of constraints for business results** 2ed. New Jersey . Prentice Hall Professional.

Aslam, W., Ijaz, F., 2018. A Quantitative Framework for Task Allocation in Distributed Agile Software Development**. IEEE Access, v6**, 15380–15390. https://doi.org/10.1109/ACCESS.2018.2803685

Badampudi, D., Fricker, S.A., Moreno, A.M., 2013. Perspectives on Productivity and Delays in Large-Scale Agile Projects, **Agile Processes in Software Engineering and Extreme Programming XP 2013,** Lecture in Business Information Processing, vol 149, pp. 180-194 Springer, Berlin, Heidelberg.

Barlow, J.B., Giboney, J.S., Keith, M.J., Wilson, D.W., Schuetzler, R.M., 2011. Overview and guidance on agile development in large organizations, **Communications of the Association for Information Systems**, v. 29 n. 2, pp. 25-44 Available at

SSRN: https://ssrn.com/abstract=1909431 or http://dx.doi.org/10.2139/ssrn.1909431 (accessed 1.20.19).

Barros-Justo, J.L., Benitti, F.B., Cravero-Leal, A.L., 2018. Software patterns and requirements engineering activities in real-world settings: A systematic mapping study. **Computer Standards & Interfaces**, v. 58, pp. 23–42.

Benbasat, I., Barki, H., 2007. Quo vadis TAM?, **Journal of the Association for Information Systems**, v. 8: Iss. 4 n. 7. Available at: http://aisel.aisnet.org/jais/vol8/iss4/7 (accessed 1.20.19)

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A., 2018. Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings**. IEEE Transactions on Software Engineering,** v. 44, Iss: 10, pp. 932–950. Available at: https://doi.org/10.1109/TSE.2017.2730870

Boehm, B., 2002. Get ready for agile methods, with Care. **Computer**, v. 35, pp. 64–69.

Boehm, B., Turner, R., 2004. Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods, **26th International Conference on Software Engineering**. ICSE 2004. Proceedings**.**. IEEE, pp. 718–719.

Cao, L., Mohan, K., Xu, P., Ramesh, B., 2004. How extreme does extreme programming have to be? Adapting XP practices to large-scale projects, in: System Sciences, 2004. **37th Annual Hawaii International Conference on System Sciences,** Proceedings of the, pp. 10.

Cataldo, M., Herbsleb, J.D., 2013. Coordination breakdowns and their impact on development productivity and software failures. **IEEE Trans. Softw. Eng.** 39, 343–360.

Cataldo, M., Herbsleb, J.D., Carley, K.M., 2008. Socio-technical congruence: a framework for assessing the impact of technical and work dependencies

on software development productivity, **Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**. ACM, pp. 2–11, New York, NY, USA.

Chuttur, M., 2009. Overview of the technology acceptance model: Origins, developments and future directions, **Sprouts: Working Papers on Information Systems**, pp. 9-37. Available at:http://sprouts.aisnet.org/9-37 (accessed 1.21.19).

Cohn, M., 2005. **Agile estimating and planning**. v.1, Upper Saddler River, NJ, ed. Pearson Education.

Colin, J., Vanhoucke, M., 2015. A comparison of the performance of various project control methods using earned value management systems. **Expert Syst. Appl.** 42, 3159–3175.

Conforto, E.C., Amaral, D.C., 2010. Evaluating an agile method for planning and controlling innovative projects. **Project Management Journal**, v. 41, Iss: 2, pp. 73–80, published: April 1, 2010.

Corona, E., Pani, F.E., 2013. A review of lean-Kanban approaches in the software development. **WSEAS Transactions on Information Science and Applications**. v. 10, Is. 1, pp. 1–13.

Crowston, K., Osborn, C.S., 1998. A coordination theory approach to process description and redesign, **Organizing Business Knowledge: The MIT Process Handbook**, Malone, T. W., Crowston, K. & Herman, G. (Eds.), pp. 335-370. Cambridge, MA: MIT Press. Available from http://ccs.mit.edu/papers/pdf/wp204.pdf.

Deemer, P., Benefield, G., Larman, C., Vodde, B., 2012. The scrum primer version 2.0 [on-line]. Available: http://www.scrumprimer.com/ (accessed 2.5.19).

De França, B.B.N., Simões, R.V., Silva, V., Travassos, G.H., 2017. Escaping from the time box towards continuous planning: an industrial experience,

in: Rapid Continuous Software Engineering (RCoSE), 2017 **IEEE/ACM 3rd International Workshop on**. IEEE, pp. 43–49.

Denger, C., Ciolkowski, M., Lanubile, F., 2004. Investigating the active guidance factor in reading techniques for defect detection, **International Symposium on Empirical Software Engineering**, 2004. ISESE'04. Proceedings. 2004 International Symposium on. IEEE, pp. 219–228, Redondo Beach, CA, USA, USA.

Despa, M.L., 2014. Comparative study on software development methodologies, **Database Systems Journal**, v. 3/2014 J. 5, pp. 37–56.

Duggan, J., Byrne, J., Lyons, G.J., 2004. A task allocation optimizer for software construction. **IEEE Software**, v. 21, Iss. 3 pp. 76–82.

Espinosa, J.A., Slaughter, S.A., Kraut, R.E., Herbsleb, J.D., 2007. Team knowledge and coordination in geographically distributed software development. **Journal of Management Information Systems**, v. 24, pp. 135–169.

Faraj, S., Sproull, L., 2000. Coordinating expertise in software development teams. **Management Science**, v. 46, pp.1554–1568.

Grinter, R.E., 1996. **Understanding dependencies: A study of the coordination challenges in Software Development**, PhD Thesis. University of California, Irvine, 1996.

Hazır, Ö., 2015. A review of analytical models, approaches and decision support tools in project monitoring and control. **International Journal of Project Management** v. 33, pp. 808–815.

Heikkilä, V.T., Paasivaara, M., Lassenius, C., 2013. ScrumBut, But Does it Matter? A Mixed-Method Study of the Planning Process of a Multi-team Scrum Organization, **2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement**. Presented at the 2013 ACM / IEEE International Symposium on Empirical Software

Engineering and Measurement, pp. 85–94. https://doi.org/10.1109/ESEM.2013.27

Highsmith, J., 2003. Agile project management: Principles and tools. **Cutter Consortium Reports**, Cutter Consortium, pp. 1-37 Arlington, MA.

Hoegl, M., Weinkauf, K., Gemuenden, H.G., 2004. Interteam coordination, project commitment, and teamwork in multiteam R&D projects: A longitudinal study. **Organization Science**, v. 15, n. 1, pp. 38–55.

Insfran, E., Fernandez, A., 2008. A systematic review of usability evaluation in web development, **International Conference on Web Information Systems Engineering**. Springer, pp. 81–91.

Jiang, Y., Jiang, J., 2008. Contextual resource negotiation-based task allocation and load balancing in complex software systems. **IEEE Transactions on Parallel and Distributed Systems**, v. 20, n. 5, pp. 641–653.

Kaliprasad, M., 2005. Agile project management: How to succeed in the face of changing project requirements. **Cost Engineering.** v. 47, ed. 10 pp. 29.

Kelley Jr, J.E., Walker, M.R., 1959. Critical-path planning and scheduling, IRE-**AIEE-ACM Computer Conference. ACM**, v. 9, n. 3, pp. 160–173.

Kerzner, H., Kerzner, H.R., 2017. Project management: a systems approach to planning, scheduling, and controlling. 20 ed., New Jersey, **John Wiley & Sons**, 2017.

Korkala, M., Maurer, F., 2014. Waste identification as the means for improving communication in globally distributed agile software development. **Journal of System and Software**, v. 95, pp.122–140.

Kraut, R.E., Streeter, L.A., 1995. Coordination in software development. **Communications of the ACM**, v. 38, pp. 69–82.

Laitenberger, O., Dreyer, H.M., 1998. Evaluating the usefulness and the ease of use of a web-based inspection data collection tool, **Proceedings Fifth**

International Software Metrics Sysmposium. IEEE, pp. 122, Bethesda, MD, USA, USA.

Lee, Y., Kozar, K.A., Larsen, K.R., 2003. The technology acceptance model: Past, present, and future. **Communications of the Association for Information Systems**, v. 12, n. 50.

Livrari, J., Livrari, N., 2011. The relationship between organizational culture and the deployment of agile methods, **Information and Software Technology**, v. 53, n. 5, pp. 509-520.

Lomas, C.D.W., Wilkinson, J., Maropoulos, P.G., Matthews, P.C., 2006. Measuring design process agility for the single company product development process. **International Journal of Agile Manufacturing**, v. 9, pp. 105–112.

Malone, T.W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C.S., Bernstein, A., Herman, G., 1999. Tools for inventing organizations: Toward a handbook of organizational processes. **Management Science**, v. 45, n. 3, pp. 425–443.

Marks, M.A., Mathieu, J.E., Zaccaro, S.J., 2001. A temporally based framework and taxonomy of team processes. **Academy of Management** v. 26, n. 3, pp. 356–376.

Martakis, A., Daneva, M., 2013. Handling requirements dependencies in agile projects: A focus group with agile software development practitioners, **Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on.** IEEE, pp. 1–11, Paris, France.

McChesney, I.R., Gallagher, S., 2004. Communication and co-ordination practices in software engineering projects. **Information and Software Technology**, v. 46, pp.473–489.

Milicic, A., El Kadiri, S., Perdikakis, A., Ivanov, P., Kiritsis, D., 2014. Toward the definition of domain concepts and knowledge through the application of

the user story mapping method. **International Journal of Product Lifecycle Management**, v. 7, pp. 3–16.

Nidumolu, S., 1995. The effect of coordination and uncertainty on software project performance: residual performance risk as an intervening variable. **Information Systems Research**, v. 6, pp. 191–219.

Olsina, L., Covella, G., Rossi, G., 2006. Web quality, in: Web Engineering. **Web Engineering Springer**, pp. 109–142.

Pai, M., McCulloch, M., Gorman, J., Pai, N., Enanoria, W., Kennedy, G., Tharyan, P., Colford JM, J., 2004. Systematic reviews and meta-analyses: an illustrated, step-by-step guide. **The National Medical Journal of India**, v. 17, pp. 86–95.

Patton, J., Economy, P., 2014. User story mapping: discover the whole story, build the right product. 1 **ed. O'Reilly Media**, Inc, USA.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology** vol. 64, pp. 1–18.

Petter, S., DeLone, W., McLean, E.R., 2012. The past, present, and future of" IS Success." **Journal of the Association for Information Systems**. v. 13, pp.341-362.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., Still, J., 2008. The impact of agile practices on communication in software development. **Empirical Software Engineering**, v. 13, pp. 303–337.

Pries-Heje, L., Pries-Heje, J., 2011. Why Scrum works: A case study from an agile distributed project in Denmark and India, **Agile Conference**, 2011. IEEE, pp. 20–28, Salt Lake City, UT, USA.

Ramasubbu, N., Bharadwaj, A., Tayi, G.K., 2015. Software process diversity: conceptualization, measurement, and analysis of impact on project performance, **MIS Quarterly**, v. 39, n. 4, pp. 787-807.

Rhee, C., Liang, Y.D., Dhall, S.K., Lakshmivarahan, S., 1994. Efficient algorithms for finding depth-first and breadth-first search trees in permutation graphs. **Information Processing Letters**, v. 49, pp. 45–50.

Rose, K.H., 2010. Effective project management: Traditional, agile, extreme. 1 ed, **Wiley Online Library**.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. **Empirical Software Engineering** v. 14, pp. 131–164.

Saga, V.L., Zmud, R.W., 1994. The Nature and Determinants of IT Acceptance, Routinization, and Infusion, in: **Proceedings of the IFIP TC8 Working Conference on Diffusion, Transfer and Implementation of Information Technology**. Elsevier Science Inc., New York, NY, USA, pp. 67–86.

Saynisch, M., 2010. Beyond frontiers of traditional project management: An approach to evolutionary, self-organizational principles and the complexity theory—results of the research program. **Project Management Journal banner,** v. 41, pp. 21–37.

Shen, M., Tzeng, G.-H., Liu, D.-R., 2003. Multi-criteria task assignment in workflow management systems, in: **System Sciences**, 2003. Proceedings of the 36th Annual Hawaii International Conference on. IEEE, p. 9–pp.

Shull, F., Carver, J., Travassos, G.H., 2001. An empirical methodology for introducing software processes, in: **ACM SIGSOFT Software Engineering Notes. ACM**, pp. 288–296, Vienna, Austria.

Solingen, R. van, Basili, V., Caldiera, G., Rombach, H.D., 2002. Goal Question Metric (GQM) Approach, in: **Encyclopedia of Software Engineering. American,** John Wiley & Sons, Inc..

Špundak, M., 2014. Mixed agile/traditional project management methodology–reality or illusion?. **Procedia-Social and Behavioral Sciences**, v. 119, pp. 939–948.

Staudenmayer, N., 1997. Interdependency: Conceptual, practical, and empirical issues. **Massachusetts Institute of Technology**

Stettina, C.J., Hörz, J., 2015. Agile portfolio management: An empirical perspective on the practice in use. **International Journal of Project Management** v. 33, pp. 140–152.

Stevenson, W.J., Hojati, M., Cao, J., 2007. Operations management., **eng.uwi.tt** McGraw-Hill/Irwin, 8th edition Boston.

Strode, D.E., 2016. A Dependency Taxonomy for Agile Software Development Projects. **Information Systems Frontiers**, v. 18, pp. 23–46. https://doi.org/10.1007/s10796-015-9574-1

Suomalainen, T., Kuusela, R., Tihinen, M., 2015. Continuous planning: an important aspect of agile and lean development. International. **Journal of Agile Systems Management** v. 8, pp.132–162. https://doi.org/10.1504/IJASM.2015.070607

Sutherland, J., Schwaber, K., 2013. The Scrum guide. the definitive guide to Scrum: The rules of the game. **ScrumGuides Com**.

Torrecilla-Salinas, C.J., Sedeño, J., Escalona, M.J., Mejías, M., 2015. Estimating, planning and managing Agile Web development projects under a value-based perspective. **Information and Software Technology** v. 61, pp. 124–144.

Travassos, G.H., Barros, M.O., 2003. Contributions of in virtuo and in silico experiments for the future of empirical studies in software engineering, in: **2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering**. pp. 117–130.

Trkman, M., Mendling, J., Krisper, M., 2016. Using business process models to better understand the dependencies among user stories. **Information and Software Technology**, v. 71, pp.58–76.

Ujigawa, K., Updegrove, D., 2016. “Agile” CCPM: Critical Chain for Software Development. **TOCICO Theory of Constraints**, New Zeland.

Venkatesh, V., Davis, F.D., 2000. A theoretical extension of the technology acceptance model: Four longitudinal field studies. **Management Science**. v. 46, n. 2, pp.186–204.

VersionOne, 2018. **12th Annual State of Agile Report** [WWW Document]. URL https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report (accessed 7.16.18).

VersionOne, 2016. **10th annual state of Agile development survey**. URL: http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10th-Annual-State-of-Agile-Report.pdf (accessed 7.16.18)

Wageman, R., 1995. Interdependence and group effectiveness. **Administrative Science Quarterly**, Sage Publications, Inc. v. 40, n. 1, pp. 145–180.

Wagstrom, P., Herbsleb, J., 2006. Dependency forecasting in the distributed agile organization. **Communications of the ACM** v. 49, pp.55–56.

Williams, L., Cockburn, A., 2003. Agile Software Development: It’s about Feedback and Change. **IEEE Computer** v. 36, pp. 39–43.

Apêndice A – Formulário de Exclusão

Estudo de Mapeamento Sistemático

Ano	Autores	Título	Razão da Exclusão
2018	Alhazmi, A., Huang, S.	**A Decision Support System for Sprint Planning in Scrum Practice**.	O assunto difere do objetivo da Dissertação. Apesar de o artigo tratar dependências entre tarefas, o principal objetivo é melhorar a estimativa das tarefas durante os *Planning Poker*
2017	Elamin, R., Osman, R.	**Towards Requirements Reuse by Implementing Traceability in Agile Development**	O assunto difere do objetivo da dissertação. O artigo tem como objetivo tratar reutilização de requisitos
2017	Silvax, A., Silva, A., Araújo, T. Perkusich, M,, Dilorenzo, E.	**Ordering the product backlog in agile software development projects: A systematic literature review**	O artigo trata a questão de dependência muito superficialmente. O principal objetivo do artigo é oferecer um mecanismo para priorização dos itens do *backlog*.
2016	Krishnan, B.S., Kovvuri, H., Balasubramani, U.M.	**Effective management of work in a geographically dispersed team using Tasks in Agile methodology**	O artigo é curto e não apresenta a solução com clareza.
2016	Grimaldi, P., Perrotta, L., Corvello, V., Verteramo, S.	**An agile, measurable and scalable approach to deliver software applications in a large enterprise**	O foco do artigo não está dentro do escopo do trabalho. O artigo apresenta uma definição de equipe em um ambiente terceirizado para utilização do SAFe
2015	Scheerer, A., Bick, S., Hildenbrand, T., Heinzl, A.	**The effects of team backlog dependencies on agile multiteam systems: A graph theoretical approach**	O artigo trata as dependências que existem entre os itens do backlog e destaca a influência das dependências na priorização dos backlog. Apesar do artigo falar sobre dependência o objetivo do artigo difere da proposta apresentada na dissertação.

2015	Heikkilä, V.T., Paasivaara, M., Rautiainen, K., (...), Toivola, T., Järvinen, J.	**Operational release planning in large-scale scrum with multiple stakeholders - A longitudinal case study at F-secure Corporation**	A proposta difere do objetivo do estudo. O artigo apresenta uma forma de distribuição de equipe para planejamento e execução de projetos utilizando Scrum em larga escala.
2013	Staron, M., Meding, W., Hoglund, C., (...), Nilsson, J., Hansson, J.	**Identifying implicit architectural dependencies using measures of source code change waves**	O assunto difere do objetivo dessa dissertação. Nesse artigo, o autor trata a identificação de dependências arquiteturais através de código fonte.
2013	Strode, D.E.	**Extending the dependency taxonomy of agile software development**	O artigo foi excluído porque já estamos usando uma versão publicada em 2016 do estudo.
2013	Daneva, M., Van Der Veen, E., Amrit, C., (...), Ramteerthkar, U., Wieringa, R.	**Agile requirements prioritization in large-scale outsourced system projects: An empirical study**	O assunto difere do objetivo da dissertação por não apresentar soluções para o problema. O artigo apenas caracteriza o problema através de estudos empíricos.
2010	Koru, A.G., El Emam, K.	**The theory of relative dependency: Higher coupling concentration in smaller modules**	O estudo difere do objetivo da proposta dessa dissertação. O trabalho relaciona a complexidade do software com a probabilidade de defeitos.
2010	Gomez, A., Rueda, G., Alarcón, P.P.	**A systematic and lightweight method to identify dependencies between user stories**	O artigo trata as dependências entre *user stories*, porém o objetivo do artigo difere da proposta apresentada na dissertação.

Apêndice B – Formulários de identificação das dependências entre tarefas por iteração

Estudo de Exploratório

Formulário de identificação de Dependências da Iteração 2.12

Situação	Título	Dependência	Tipo de Dependência	Atividade de Processo
Done	[2.12.0] Erro *NullPointer* ao liberar pagamento de pedido de posicionamento em HOMOL			Implementar UC
Done	[2.12.0] Erro *NullPointer* ao liberar pagamento de pedido de posicionamento em HOMOL	21723 TI	Processo	Testar UC
Done	[Acesso ao sistema] - Cliente não consegue escolher perfil no PTVV	21612 TI	Processo	Implementar UC
Done	[Acesso] - Mudanças no Perfil Faturamento	21763 TI	Processo	Testar UC
Done	[Acesso] [Pedido Posicionamento] - Mudanças no Perfil Faturamento	21612 TI	Atividade	Implementar UC
Done	[Acesso] Sistema Esta exibindo incorretamente menu alfandegado	21612 TI	Processo	Implementar UC
Done	[Acesso] Sistema Esta exibindo incorretamente menu alfandegado	21734 TI;	Processo	Testar UC
Done	[Busca Posicionamento] - Ao acessar a tela de busca de pedidos. com perfis cliente / despachante os filtros avançados devem vir preenchidos	21603 TI;	Processo	Implementar UC
Done	[Busca Posicionamento] - Ao acessar a tela de busca de pedidos. com perfis cliente / despachante os filtros avançados devem vir preenchidos	21655 TI;	Processo	Testar UC
Done	[Conclusão Pedido Posicionamento] - Alterações no pedido posicionamento	21653 TI	Atividade	Implementar UC
Done	[Conclusão Pedido Posicionamento] - Alterações no pedido posicionamento	21694 TI;	Processo	Testar UC
Done	[Conclusão vistoria N4] - Corrigir ocorrências de conclusão da vistoria	21653 TI	Atividade	Implementar UC

Done	[Conclusão vistoria N4] - Corrigir ocorrências de conclusão da vistoria	21691 TI	Processo	Testar UC
Done	[Controle Acesso] Criar script de acesso (ver RN246 e RN250)			Implementar UC
Done	[Controle Acesso] Criar script de acesso (ver RN246 e RN250)	21612 TI	Processo	Testar UC
Done	[Criação pedido Posicionamento] - Erro ao criar pedido além da capacidade	21598 TI;	Atividade,	Implementar UC
Done	[Criação pedido Posicionamento] - Erro ao criar pedido além da capacidade	21730 TI;	Processo	Testar UC
Done	[Criação Pedido Posicionamento] - Para contêineres com pedidos já criados ou não concluídos não deve ser permitido criar pedidos.	21589 TI	Atividade	Implementar UC
Done	[Criação Pedido Posicionamento] - Para contêineres com pedidos já criados ou não concluídos não deve ser permitido criar pedidos.	21667 TI;	Processo	Testar UC
Done	[Criação posicionamento exportação] - Erro ao tentar salvar pedido.	21598 TI	Atividade	Implementar UC
Done	[Criação posicionamento exportação] - Erro ao tentar salvar pedido.	216664 TI;	Processo	Testar UC
Done	[Criação Posicionamento] - DetachedEntity ao criar pedido posicionamento	21589 TI; 21598 TI	Atividade	Implementar UC
Done	[Criação Posicionamento] - Pagador não correntista não cria pedido posicionamento			Testar UC
Done	[Criar Pedido Posicionamento Importação] - Ao salvar pedido de posicionamento com 2 contêineres. O sistema exibe mensagem de erro	21598 TI	Atividade	Implementar UC
Done	[Criar Pedido Posicionamento Importação] - Ao salvar pedido de posicionamento com 2 contêineres. O sistema exibe mensagem de erro	21679 TI	Processo	Testar UC
Done	[Detalhes pedido posicionamento] - alteração de layout	21604 TI;	Atividade	Implementar UC
Done	[Detalhes pedido posicionamento] - alteração de layout	21662 TI	Processo	Testar UC
Done	[GradeHoraria] Alterar job para criação de vagas GradeHoraria de Posicionamento (ver RN250)	21602 TT	Atividade	Implementar UC
Done	[GradeHoraria] Alterar tela CONFIGURAÇÃO DE GRADE HORÁRIA para GradeHoraria de Posicionamento (ver RN250)			Implementar UC

Done	[GradeHoraria] Alterar tela GERAÇÃO DE GRADE HORÁRIA para GradeHoraria de Posicionamento (ver RN250)	21608 TT	Processo	Implementar UC
Done	[GradeHoraria] Alterar tela PARAMETRIZAÇÃO DA GRADE HORÁRIA para GradeHoraria de Posicionamento (ver RN250)			Implementar UC
Done	[Importação Siscomex] O Sistema exibe erro de CE não existente no arquivo			Implementar UC
Done	[Incidente 214421] Imagens dobradas RX portal TVV			Implementar UC
Done	[Incluir Posicionamento] - Validar existência da viagem na criação do pedido posicionamento.	21598 TI	Atividade	Implementar UC
Done	[Incluir Posicionamento] - Validar existência da viagem na criação do pedido posicionamento.	21721 TI	Processo	Testar UC
Done	[Inclusão Posicionamento] Ocultar fieldset			Implementar UC
Done	[Inclusão Posicionamento] Ocultar fieldset	21761 TI	Processo	Testar UC
Done	[Integração N4 x Ptvv] - Alteração de Log Pedido posicionamento.	21596 TI	Atividade	Implementar UC
Done	[Integração N4 x Ptvv] - Alteração de Log Pedido posicionamento.	21726 TI	Processo	Testar UC
Done	[Integração N4 x PTVV] - Problema de envio de e-mail			Implementar UC
Done	[Integração N4 x PTVV] - Problema de envio de e-mail	21724 TI	Processo	Testar UC
Done	[Navis] Configurar Navis			Implementar UC
Done	[OS 214463] Erro na geração de grade horária via job	21611 TI	Atividade	Implementar UC
Done	[OS 214463] Erro na geração de grade horária via job	21648 TI	Processo	Testar UC
Done	[Parametrização Grade Horaria posicionamento] - Alteração de layout	21608 TI	Atividade	Implementar UC
Done	[Parametrização Grade Horaria posicionamento] - Alteração de layout	21644 TI	Processo	Testar UC
Done	[Pedido Posicionamento] - Exibição de ocorrências para os pedidos de posicionamento	21591 TI	Atividade	Implementar UC
Done	[Pedido Posicionamento] - Geração de ocorrências para os pedidos de posicionamento	21671 TI	Processo	Testar UC
Done	[Pedido Posicionamento] Cliente logado como despachante salvar não consegue concluir pedido	21653 TI	Processo	Testar UC

Done	[Pedido Posicionamento] Cliente logado como despachante salvar não consegue concluir pedido	21598 TI;	Atividade	Implementar UC
Done	[Pedido Posicionamento] Implementar na criação do pedido comportamento da seção Empresas para o Programador igual ocorre para o Administrador de Sistemas	21598 TI	Atividade	Implementar UC
Done	[Pedido Posicionamento] Implementar na criação do pedido comportamento da seção Empresas para o Programador igual ocorre para o Administrador de Sistemas	21685 TI	Processo	Testar UC
Done	[Posicionamento] - Suporte a desenvolvimento			Testar UC
Done	[Posicionamento] - Suporte a teste			Implementar UC
Done	[PRODUÇÃO] Erro ao importar nota fiscal na tela de edição de pedido de recepção			Implementar UC
Done	[Sugerir Data Posicionamento] - Controlar acesso ao botão sugerir data na tela de detalhes pedido posicionamento	21612 TI	Atividade	Implementar UC
Done	[Sugerir Data Posicionamento] - Controlar acesso ao botão sugerir data na tela de detalhes pedido posicionamento	21794 TI	Processo	Testar UC
Done	[Sugerir Data Posicionamento] - Perfil Faturamento não deve poder sugerir data			Testar UC
Done	[Sugestão Data] - Erro ao sugerir data usando grade negativa como programador	21598 TI;	Atividade	Implementar UC
Done	[Sugestão Data] - Erro ao sugerir data usando grade negativa como programador	21728 TI	Processo	Testar UC
Done	[Tela Criação Posicionamento] - alterações de campo finalidade	21642 TI	Atividade	Implementar UC
Done	[Tela Criação Posicionamento] - alterações de campo finalidade	21646 TI	Processo	Testar UC
Done	[Tela Criação Posicionamento] - alterações de layout na tela de criação de posicionamento	21591TI	Atividade	Implementar UC
Done	[Tela Criação Posicionamento] - alterações de layout na tela de criação de posicionamento	21642 TI	Processo	Testar UC
Done	Ajuste na integração no PTVV - N4 de acordo com novo entendimento			Implementar UC
Done	Ajuste na integração no PTVV - N4 de acordo com novo entendimento	21793 TI	Processo	Testar UC

Done	Alterar a *procedure* de criação de Pedido Posicionamento para agrupar os contêineres por viagem	21601 TI	Atividade	Implementar UC
Done	Alterar a *procedure* de criação de Pedido Posicionamento para agrupar os contêineres por viagem	21814 TI	Processo	Testar UC
Done	Alterar a *procedure* de Liberação de Pagamento do Pedido de Posicionamento para agrupar contêineres por Viagem	21602 TI	Atividade	Implementar UC
Done	Alterar a *procedure* de Liberação de Pagamento do Pedido de Posicionamento para agrupar contêineres por Viagem	21815 TI	Processo	Testar UC
Done	Alterar NavisMiddleware (ver RN251)			Implementar UC
Done	Bugs na tela de detalhes (Ocorrência. envio de e-mail e cliente correntista)	21604 TI	Atividade	Implementar UC
Done	Bugs na tela de detalhes (Ocorrência. envio de e-mail e cliente correntista)	21693 TI	Processo	Testar UC
Done	Criar entidades Java PedidoPosicionamento	21590 TI	Atividade,	Implementar UC
Done	Criar estrutura inicial da Tela Criação	21589 TT	Atividade	Implementar UC
Done	Criar parâmetros de configuração LIMITE_HORARIO_POSICIONAMENTO_D1_CORRENTISTA. LIMITE_HORARIO_POSICIONAMENTO_D1_NAO_CORRENTISTA			Implementar UC
Done	Criar procedure convivência GTVV: criacaoPedidoPosicionamento (ver RN248)			Implementar UC
Done	[Incidente 214463] PTVV - ERRO AO IMPORTAR PLANILHA DE UPLOAD DE HORARIO DTC			
Done	Criar *procedure* convivência GTVV: liberarPagamentoPedidoPosicionamento	21601 TT	Atividade	Implementar UC
Done	Criar *scripts* de banco de dados PedidoPosicionamento			Implementar UC
Done	Criar serviço NavisCriacaoEventoPosicionamentoService (ver RN251)	21599 TT	Atividade,	Implementar UC
Done	Implementar botão 'Liberar Pagamento' (ver RN249)	21589 TI	Atividade,	Implementar UC
Done	Implementar botão Salvar (ver RN248)	21591 TI; 21589 TI; 21594 TT,	Processo, Atividade, Atividade,	Implementar UC

		21592 TT, 21596 TT	Atividade, Atividade	
Done	Implementar comportamento da combo Categoria	21603TI	Processo	Implementar UC
Done	Implementar comportamento seção Documentos/Contêineres (Tela Criação/Edição) (ver RN247)	21591 TI; 21589TI	Processo, Atividade	Implementar UC
Done	Implementar comportamento seção Empresas (Tela Criação/Edição) (ver RN248)	21591 TI; 21589TI	Processo, Atividade	Implementar UC
Done	Implementar consumidor de mensagens do Navis (ver RN254)	21599 TT	Processo	Implementar UC
Done	Implementar PedidoPosicionamentoConteinerQuery (ver RN247)			Implementar UC
Done	Implementar RN254			Implementar UC
Done	Implementar seção Sugerir Horário ao criar pedido (ver RN248 e RN250)	21591 TI; 21589 TI	Atividade, Atividade	Implementar UC
Done	Implementar seção Sugerir Horário na tela de detalhes (ver RN248 e RN250)	21591 TI; 21589 TI	Atividade, Atividade	Implementar UC
Done	Implementar Tela Busca	21589 TI	Processo	Implementar UC
Done	Implementar Tela Detalhes	21603 TT; 21589 TI	Atividade, Processo	Implementar UC
Done	Problema com Activemq Produção			Implementar UC
Done	Suporte a produção			Implementar UC
Done	Validar botão 'Liberar Pagamento' (ver RN249)	21605 TI	Processo	Testar UC
Done	Validar criação pedido posicionamento	21598 TI;	Processo	Testar UC
Done	Validar Envio de mensagem STOW GRP para o N4			Testar UC
Done	Validar geração e configuração da grade horaria			Testar UC
Done	Validar *procedure* convivencia GTVV: criacaoPedidoPosicionamento (ver RN248)	21601 TI	Processo	Testar UC
Done	Validar *procedure* convivencia GTVV: liberarPagamentoPedidoPosicionamento	21602 TI	Processo	Testar UC
Done	Validar resposta do n4 de conclusão da vistoria	21629 TI	Processo	Testar UC
Done	Validar resposta do n4 de conclusão do posicionamento	21629 TI	Processo	Testar UC
Done	Validar se o *contêiner* é do Cliente nos Pedido de Posicionamento	21664 TT	Processo	Implementar UC

	de Exportação			
Done	Validar se o contêiner é do Cliente nos pedidos de Posicionamento de Exportação	21863 TI	Processo	Testar UC
Done	Validar seção Sugerir Horário ao criar pedido (ver RN248 e RN250)	21596 TI; 21598TI; 21601 TI	Processo. Atividade, Atividade	Testar UC
Done	Validar seção Sugerir Horário na tela de detalhes (ver RN248 e RN250)	21596 TI; 21598TI; 21601 TI	Processo, Atividade, Atividade	Testar UC
Done	Validar Tela Detalhes posicionamento	21604 TI	Processo	Testar UC
Done	Validar Tela Busca posicionamento	21603 TI	Processo	Testar UC

Formulário de identificação de Dependências da Iteração 2.13

Situação	Título	Dependência	Tipo de Dependência	Atividade de Processo
Done	[Tela Faturamento] [Liberar Pagamento] Configurar Acesso	21712 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Configurar Acesso	21722 TI	Processo	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Criar classes (FiltrosVO. TaxaVO. Bean e Service)	21722 TI	Processo	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Criar estrutura inicial da tela	21722 TI	Processo	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Criar seção Resultados	21703 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Criar seção Resultados	21722 TI, 21702 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar botões Limpar Campos e Buscar	21706 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar botões Limpar Campos e Buscar	21722 TI	Processo	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Gerar Boletos (Serviço RECEPCAO)	21709 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Gerar Boletos (Serviço RECEPCAO)	21722 TI, 21747 TT, 21708 TI, 21707	Processo, Atividade, Atividade, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço POSICIONAMENTO)	21752 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço POSICIONAMENTO)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC

Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço RECEPCAO)	21710 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço RECEPCAO)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço RETIRADA DTC)	21715 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço RETIRADA DTC)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço RETIRADA Importação)	21714 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar Liberar Pagamento (Serviço RETIRADA Importação)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar recuperarTaxas (Serviço POSICIONAMENTO)	21708 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar recuperarTaxas (Serviço POSICIONAMENTO)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar recuperarTaxas (Serviço RECEPCAO)	21707 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar recuperarTaxas (Serviço RECEPCAO)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar recuperarTaxas (Serviço RETIRADA)	21713 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar seção Navio Viagem	21705 TI	Processo	Testar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar seção Navio Viagem	21722 TI, 21702 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] [Liberar Pagamento] Implementar seções Filtros de Consulta e Dados do Pedido	21704 TI	Processo	Testar UC

Done	[Tela Faturamento] [Liberar Pagamento] Implementar seções Filtros de Consulta e Dados do Pedido	21722 TI, 21702 TI	Processo, Atividade	Implementar UC
Done	[Tela Faturamento] Implementar recuperarTaxas (Serviço RETIRADA)	21722 TI, 21701 TI	Processo, Atividade	Implementar UC
Done	Ajustar Navis.plugin OwsePedidoPosicionamento			Implementar UC
Done	Prepara branches			Criar Branch
Done	Refactoring método empresaService.validarCNPJ			Implementar UC

Formulário de identificação de Dependências da Iteração 2.14

Situação	Título	Dependência	Tipo de Dependência	Atividade de Processo
Done	[Criação Pedido Posicionamento Importação] - Validação exclusiva para exportação esta sendo aplicada para pedidos de importação.	21849 TI,	Processo	Implementar UC
Done	Alterar a lógica de recuperação de GH em função do tipo do Pedido	21849 TI	Processo	Implementar UC
Done	Alterar o diagrama de estado para adaptar o novo fluxo de reprogramação			
Done	Validar a lógica de recuperação de GH em função do tipo do Pedido	21926 TI	Processo	Testar UC
Done	Adaptar o serviço do NavisMiddleware que envia ICU para o ArgoService do N4	21839 TI	Processo	Implementar UC
Done	Ajustar fluxo Liberar Pagamento para especificar qual DocCobranca está sendo pago	21839 TI	Processo	Testar UC
Done	Ajustar fluxo Liberar Pagamento para especificar qual DocCobranca está sendo pago	21846 TI, 21849 TI	Atividade, Processo	Implementar UC
Done	Ajustar o fluxo de Criação para identificar possível tentativa de escapar de multa de reprogramação	21841 TI	Processo	Testar UC

Done	Ajustar o fluxo de Criação para identificar possível tentativa de escapar de multa de reprogramação	21860 TI, 21849 TI	Atividade, Processo	Implementar UC
Done	Ajustar tela CONFIGURAÇÃO DE GRADE HORÁRIA	21831 TI	Processo	Testar UC
Done	Ajustar tela CONFIGURAÇÃO DE GRADE HORÁRIA	21849 TI	Processo	Implementar UC
Done	Ajustar tela GERACAO DE GRADE HORÁRIA	218832 TI	Processo	Testar UC
Done	Ajustar tela GERACAO DE GRADE HORÁRIA	21849 TI	Processo	Implementar UC
Done	Ajustar tela PARAMETRIZAÇÃO DA GRADE HORÁRIA	21849 TI	Processo	Implementar UC
Done	Alterar a versão do trunk para 2.14.0	21849 TI	Processo	Alterar versão do Trunk
Done	Alterar rotina de geração de grade horária (Manual e Job)	21849 TI	Processo	Implementar UC
Done	Alterar rotina de geração de grade horária (Manual e Job)	21830 TI	Processo	Testar UC
Done	Botão Cancelar Contêiner	21820TT; 21849 TI,	Atividade, Processo	Implementar UC
Done	Botão Cancelar Pedido	21821TT; 21822 TT; 21825 TT, 21849 TI	Processo, Atividade, Atividade, Negocio	Implementar UC
Done	Botão Reprogramação de Pedido Posicionamento	21837 TT; 21838 TT; 21840TT. 21849 TI	Processo, Atividade, Atividade, Atividade, Atividade	Implementar UC
Done	Configurar o Navis para Cancelar Pedido Posicionamento	21849 TI, 21827 TI	Processo, Atividade	Implementar UC
Done	Conviver cancelamento Ptvv X N4	21818 TI, 21819 TI	Processo	Testar UC
Done	Criar as colunas passivel_multa e tem_multa na tabela TB_POSICIONAMENTO_CONTEINER	21849 TI	Processo	Implementar UC
Done	Criar o branch 2.13.0 a partir do trunk			Criar Branch

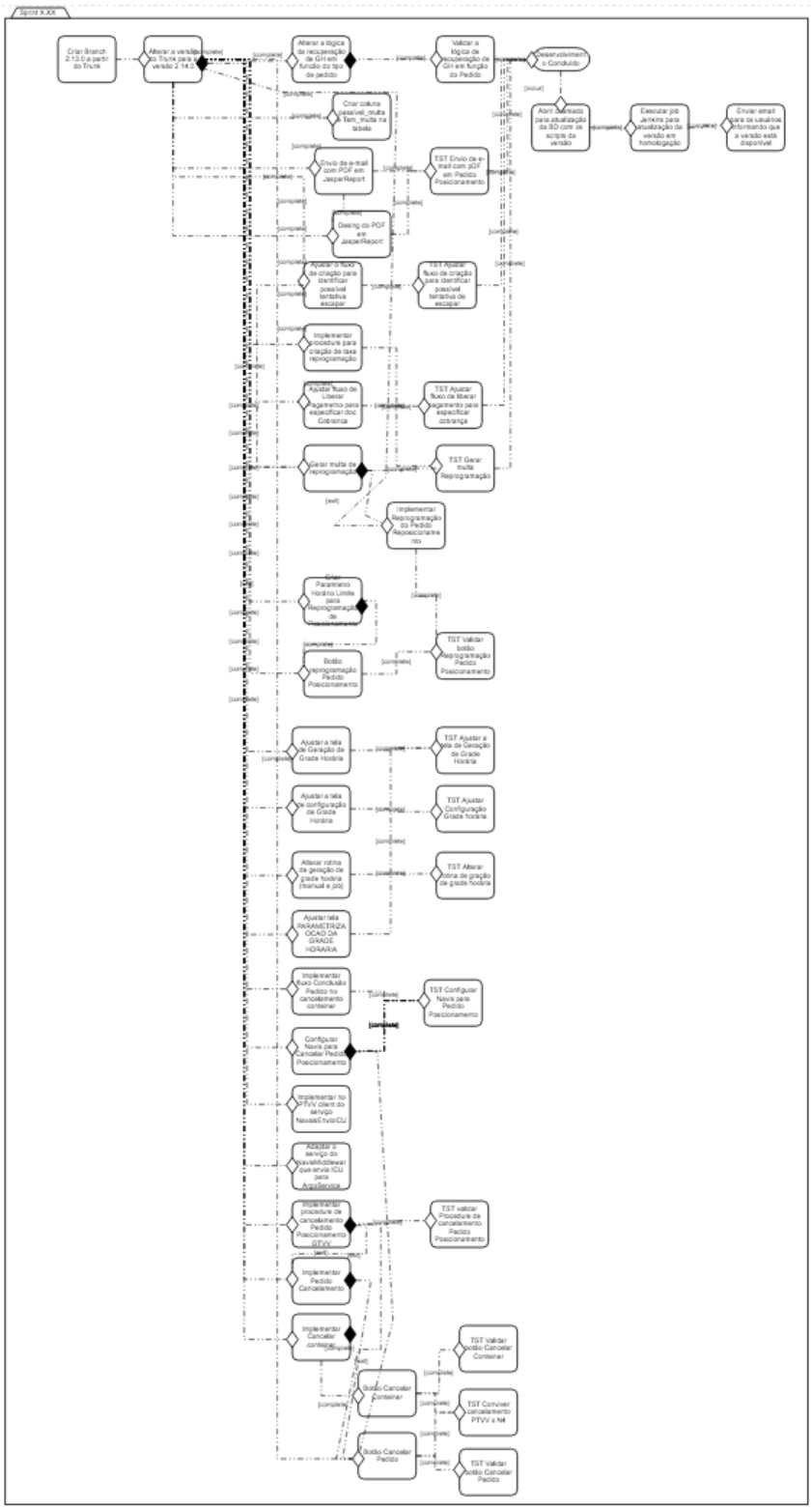
Done	Criar parâmetro 'Horário limite para Reprogramação de Posicionamento sem multa'	21849 TI	Processo	Implementar UC
Done	Design do PDF em JasperReport	21849 TI	Processo	Implementar UC
Done	Envio de email / PDF do pedido Posicionamento	21844 TI	Processo	Testar UC
Done	Envio de Email com PDF do Pedido Posicionamento	21843 TI, 21849 TI	Atividade, Processo	Implementar UC
Done	Gerar Multa de Reprogramação	21840 TT	Atividade	Implementar UC
Done	Gerar Multa de Reprogramação	21838 TI	Processo	Testar UC
Done	Remover serviço NavisCriacaoPedidoPosicionamentoICU e usar NavisEnvioICU			Implementar UC
Done	Implementar Cancelar Contêiner	21849 TI, 21821 TI	Processo, Atividade	Implementar UC
Done	Implementar Cancelar Pedido	21822TT	Atividade	Implementar UC
Done	[Navis] Tratar erro referente ao processamento do evento PESAGEM_DESCARGA	21849 TI,	Processo	Implementar UC
Done	Implementar fluxo Conclusão Pedido no cancelamento de contêiner (se houver pelo menos um Contêiner Concluído no Pedido)	21849 TI	Processo	Implementar UC
Done	Implementar no PTVV client do serviço NavisEnvioICU	21849 TI	Processo	Implementar UC
Done	Implementar procedure de Cancelamento Pedido Posicionamento no GTVV	21849 TI	Processo	Implementar UC
Done	Implementar *procedure* de criação de taxa de reprogramação	21846 TI; 21849 TI	Atividade, Processo	Implementar UC
Done	Implementar Reprogramação de Pedido Posicionamento	21838 TI; 21849 TI	Atividade, Processo	Implementar UC
Doing	Testar checkAll na liberação de pagamento de posicionamento	21915 TI	Processo	Testar UC
Done	Validar Botão Cancelar Contêiner	21818 TI	Processo	Testar UC
Done	[POSICIONAMENTO] Relatório de Vagas na Grade	21002 TI	Processo	Testar UC

Done	[Cancelar Contêiner exportação] - Sistema não retorna as grades horarias utilizadas pelo pedido ao cancelar Pedido	21898 TI	Processo	Testar UC
Done	[Cancelar Contêiner exportação] - Sistema não retorna as grades horarias utilizadas pelo pedido ao cancelar Pedido	21849 TI,	Processo	Implementar UC
Done	[Cancelar Contêiner Recepção] - Sistema não realiza cancelamento de pedido recepção	21883 TI,	Processo	Testar UC
Done	[Cancelar Contêiner Recepção] - Sistema náo realiza cancelamento de pedido recepção	21849 TI,	Processo	Implementar UC
Done	[Cancelar Pedido Posicionamento] - Exibição do botão cancelar pedido	21882 TI	Processo	Testar UC
Done	[Cancelar Pedido Posicionamento] - Exibição do botão cancelar pedido	21849 TI,	Processo	Implementar UC
Done	[Criação Pedido Posicionamento Importação] - Validação exclusiva para exportação está sendo aplicada para pedidos de importação.	21903 TI	Processo	Testar UC
Done	[ICU] Corrigir envio pro N4	21849 TI,	Processo	Implementar UC
Done	[Integração Portal Único Exportação] Problema no XML de ENTREGA CARGA POR CONTÊINER	21771 TI	Processo	Testar UC
Done	[Integração Portal Único Exportação] Problema no XML de ENTREGA CARGA POR CONTÊINER	21849 TI	Processo	Implementar UC
Done	[Liberar Pagamento Posicionamento] - Sistema exibe erro ao tentar liberar pagamento	21918 TI	Processo	Testar UC
Done	[Liberar Pagamento Posicionamento] - Sistema exibe erro ao tentar liberar pagamento	21849 TI,	Processo	Implementar UC
Done	[DESENVOLVIMENTO] Problemas de Produção			
Done	[Liberar Pagamento] [Posicionamento]- Botão liberar pagamento esta sendo exibido para pedidos cancelados.	21908 TI	Processo	Testar UC
Done	[Liberar Pagamento] [Posicionamento]- Botão liberar pagamento esta sendo exibido para pedidos cancelados.	21849 TI,	Processo	Implementar UC
Done	[ANALISE] Problemas de Produção			

Done	[Pedido de Posicionamento] Inconsistência ao cancelar (fechar) modal de sugerir data	21849 TI,	Processo	Implementar UC
Done	[POSICIONAMENTO] Colocar números das Programações de manuseio na ocorrência do pedido de posicionamento	21878 TI	Processo	Testar UC
Done	[POSICIONAMENTO] Colocar números das Programações de manuseio na ocorrência do pedido de posicionamento	21849 TI,	Processo	Implementar UC
Done	[POSICIONAMENTO] Novos filtros de consulta de pedido	21879 TI	Processo	Testar UC
Done	[POSICIONAMENTO] Novos filtros de consulta de pedido	21849 TI,	Processo	Implementar UC
Done	[POSICIONAMENTO] Relatório de Vagas na Grade	21849 TI,	Processo	Implementar UC
Done	[REFACTORING] Refatorar o metodo validaEmpresaAtiva	21849 TI,	Processo	Implementar UC
Done	Aplicar alterações numeroPedidoService em 2.13 e 2.14	21849 TI,	Processo	Implementar UC
Done	Bug ao cancelar Reprogramação	21906 TI	Processo	Testar UC
Done	Bug ao cancelar Reprogramação	21849 TI,	Processo	Implementar UC
Done	CheckAll do prompt de cancelamento de contêineres não funciona	21849 TI,	Processo	Implementar UC
Done	Validar Botão Cancelar Pedido	21819 TI	Processo	Testar UC
Done	Validar Botão Reprogramação de Pedido Posicionamento	21835 TI	Processo	Testar UC
Done	Criar uma classe EmailPedidoPosicionamentoService	21849 TI,	Processo	Implementar UC
Done	Implementar checkAll na liberação de pagamento de posicionamento	21849 TI,	Processo	Implementar UC
Done	Problema [Produção] de envio de PreAdvise	21849 TI,	Processo	Implementar UC
Done	Tela de detalhes está permitindo programar contêineres cancelados	21849 TI,	Processo	Implementar UC
Done	Testar [Pedido de Posicionamento] Inconsistência ao cancelar (fechar) modal de sugerir data	21884 TI	Processo	Testar UC

Done	Validar procedure de Cancelamento Pedido Posicionamento no GTVV	21822TI	Processo	Testar UC
Done	[Planning] Reunião de planejamento das tarefas da sprint 2.14.0			

Apêndice C – Rede de tarefas executadas na iteração 2.14



Apêndice D – TAM - Formulário de Consentimento e Desimpedimento de Participação

AVALIAÇÃO TAM

OBJETIVO

Esse estudo visa a observar a efetividade do uso do Método do Caminho Crítico em Ambientes Ágeis. Será utilizada uma ferramenta *AgileCriticalPath* na qual o aluno deverá interagir para entender quais tarefas podem impactar as entregas, proporcionando uma experiência diferente dentro do contexto de métodos ágeis.

DEFINIÇÕES

Caminho Crítico: Método do Caminho Crítico é uma abordagem que permite encontrar as tarefas que não têm flexibilidade de datas, o atraso nessas tarefas retarda a entrega final do produto.

Tarefas Críticas: São as tarefas que compõem o Caminho Crítico.

IDADE

Eu declaro ter mais de 18 anos de idade e concordar em participar desse estudo experimental conduzido pela aluna de mestrado *Rachel Vital Simões* pela COPPE/UFRJ.

PROCEDIMENTO

As atividades a serem realizadas no estudo são de caráter individual e voluntário. Para melhorar a efetividade do estudo, será realizada uma apresentação inicial sobre os conceitos e técnicas utilizadas.

Após os procedimentos iniciais o participante deverá executar as tarefas propostas e preencher os formulários com o relato sobre a experiência no estudo. É de extrema importância que não haja discussão entre os participantes sobre os modelos avaliados.

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será divulgado. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do experimento.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA.

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado. Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

Pesquisador Responsável: Rachel Vital Simões (COPPE/UFRJ)

Professores Responsáveis: Prof. Toacy Oliveira (COPPE/UFRJ)

Nome do Participante (em letra de forma): _____

Assinatura: _____ Data: _____

Apêndice E – TAM - Caracterização do Participante

AVALIAÇÃO TAM

Este formulário faz parte do estudo de avaliação da ferramenta *AgileCriticalPath*.

Este estudo está inserido nas atividades de pesquisa e desenvolvimento relacionadas à dissertação de mestrado da aluna Rachel Vital Simões, na linha de Engenharia de Software do Programa de Engenharia de Sistemas e Computação – PESC (COPPE/UFRJ), sob orientação do prof. Toacy Cavalcante de Oliveira.

Esta pesquisa adota os princípios éticos e científicos que norteiam a pesquisa científica em engenharia de software. Desta forma, dados pessoais e sensíveis não são solicitados, a participação é totalmente voluntária. Todos os resultados serão apresentados de forma agregada, sem a possibilidade de identificação do respondente.

O objetivo principal desse estudo é analisar a eficiência e adequação técnica da ferramenta através do Modelo de Aceitação de Tecnologia (TAM). Os resultados desta pesquisa são de relevância para a indústria de software.

Esse questionário visa a identificar os participantes do estudo quanto a sua experiência e seu domínio nos assuntos relacionados. O tempo esperado de resposta pode variar de 5 a 15 minutos.

A participação do profissional de software nesta avaliação é voluntária. No entanto, é muito importante e relevante a sua participação!

Caso tenha dúvidas ou sugestões, por favor, entre em contato através do e-mail rachelvital@cos.ufrj.br.

Código do Participante: _____ Data: _____

Por favor, NÃO inclua nenhum detalhe que poderá identificá-lo.

Perfil do participante

1) Formação Acadêmica:

( ) Doutorado

( ) Doutorado em Andamento

( ) Mestrado

( ) Mestrado em Andamento

( ) Graduação

( ) Graduação em Andamento

( ) Outro: _____

Ano de ingresso: _____ Ano de conclusão (de previsão): _____

2) Quanto tempo de experiência na <u>academia</u> você possui na área de Engenharia de Software?

( ) Nenhum
( ) Até 1 ano
( ) De 1 a 3 anos
( ) De 3 a 5 anos
( ) De 5 a 7 anos
( ) Mais de 7 anos

3) Quanto tempo de experiência no <u>mercado</u> você possui na área de Engenharia de Software?

( ) Nenhum
( ) Até 1 ano
( ) De 1 a 3 anos
( ) De 3 a 5 anos
( ) De 5 a 7 anos
( ) Mais de 7 anos

4) Você possui conhecimentos sobre o conceito de Gestão de Projetos?

( ) Sim, absolutamente.
( ) Sim, mas pouco.
( ) Não, mas já ouvi falar sobre.
( ) Não

5) Você conhece os conceitos básicos do Scrum ?

( ) Sim, absolutamente.
( ) Sim, mas pouco.
( ) Conheço apenas um deles.
( ) Não

6) Por favor, descreva brevemente o que você sabe sobre métodos ágeis.

_____

_____

_____

_____

7) Por favor, cite os métodos ágeis que você julgue mais populares.

_____

_____

_____

8) Na escala dos 5 pontos abaixo:

0 = nenhum
1 = estudei em aula ou em livro
2 = pratiquei em projetos em sala de aula

3 = usei em projetos pessoais

4 = usei em projetos na indústria

Marque uma opção, indicando o grau de sua experiência em:

Gestão de Projetos	0	1	2	3	4
Gestão Ágil de Projetos	0	1	2	3	4
Lean	0	1	2	3	4
Métodos Ágeis	0	1	2	3	4
SCRUM	0	1	2	3	4
XP	0	1	2	3	4

Agradecemos sua colaboração!