UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

UM MÉTODO BASEADO EM MODELO PROBABILÍSTICO PARA AUXÍLIO NA DETECÇÃO DE PROBLEMAS NA UTILIZAÇÃO DO SCRUM EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE

MESTRANDO
MIRKO BARBOSA PERKUSICH

ORIENTADOR
HYGGO ALMEIDA

CAMPINA GRANDE
AGOSTO - 2013

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Um Método Baseado em Modelo Probabilístico para Auxílio na Detecção de Problemas na Utilização do Scrum em Projetos de Desenvolvimento de Software

Mirko Barbosa Perkusich

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Processos de Desenvolvimento de Software

Hyggo Almeida
(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

P451m Perkusich, Mirko Barbosa.

Um método baseado em modelo probabilístico para auxílio na detecção de problemas na utilização do *Scrum* em projetos de desenvolvimento de Software / Mirko Barbosa Perkusich. – Campina Grande, 2013.

141 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2013.

"Orientação: Prof. Dr. Hyggo Almeida".

Referências.

1. Projeto de Desenvolvimento de Software. 2. Arcabouço Ágil. 3. Redes Bayesianas. 4. Scrum. I. Almeida, Hyggo. I. Título.

CDU 004.415.2(043)

"UM MÉTODO BASEADO EM MODELO PROBABILÍSTICO PARA AUXÍLIO À DETECÇÃO DE PROBLEMAS NA UTILIZAÇÃO DO SCRUM EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE"

MIRKO BARBOSA PERKUSICH

DISSERTAÇÃO APROVADA EM 30/08/2013

HYGGO OLIVEIRA DE ALMEIDA, D.Sc, UFCG
Orientador(a)

JOSEANA MACÊDO FECHINE RÉGIS DE ARAÚJO, D.Sc, UFCG
Examinador(a)

RODRIGO DE BARROS PAES, Dr., UFAL
Examinador(a)

CAMPINA GRANDE - PB

UNIVERSIDADE FEDERAL
DE ALAGOAS

Prezado Coordenador do Programa de Pós-Graduação em Ciência da Computação da UFCG,

Declaro que participei da banca examinadora da Dissertação de Mestrado de Mirko Barbosa Perkusich, no dia 30/08/2013, via teleconferência.

Declaro ainda ter emitido, na ocasião, o conceito APROVADO para o trabalho em questão.

Maceió, 30 de agosto de 2013.

Rodrigo de Barros Paes

Instituto de Computação/UFAL

Resumo

Há uma taxa elevada de projetos de desenvolvimento de software que não alcançam seus objetivos. Quanto antes os problemas no projeto forem detectados, maior sua probabilidade de sucesso e menor será o prejuízo financeiro e de tempo. A utilização de metodologias e arcabouços ágeis em projetos de desenvolvimento de software vem se popularizando por possuir práticas, regras e princípios leves que agilizam a detecção de problemas e permitem mudanças frequentes nos projetos. O arcabouço ágil mais popular é o *Scrum*. Mesmo o *Scrum* tendo práticas, regras e princípios simples, profissionais encontram dificuldades em aplicá-lo na indústria, principalmente em equipes acostumadas ao modelo tradicional de gerência de projetos. Para auxiliar o *ScrumMaster* no seu papel de facilitar a aplicação do *Scrum*, um método para detectar problemas na sua aplicação em projetos de desenvolvimento de software é apresentado nesta dissertação. Trata-se de um método cíclico que utiliza um modelo probabilístico que serve para prover dados referentes ao projeto para o *ScrumMaster*. Dada a capacidade de modelar incertezas e sua flexibilidade para modificações, *Redes Bayesianas* foram utilizadas para implementar o modelo. O modelo foi validado a partir de testes em cenários, e o método a partir de um estudo de caso em dois projetos de uma empresa. Os resultados obtidos demonstram que a utilização do método é capaz de detectar problemas na aplicação do *Scrum* com custo-benefício positivo e útil para guiar a equipe na busca por excelência.

Palavras chave: Ágil, Redes Bayesianas, Scrum

Abstract

There is a high rate of software development projects that fail. Whenever problems can be detected ahead of time, projects may have better chances of success, and therefore save money and time. Recently, the usage of agile methodologies and frameworks have been increasing due to its lightweight practices, rules and principles that allow frequent changes during project execution. *Scrum* is the most popular agile framework. Even though it is composed of simple practices, rules and principles, professionals find it hard to apply it in the industry, specially, in teams used to traditional project management. To help *ScrumMasters* to fulfill their duty of facilitating the usage of *Scrum*, this dissertation presents a method to detect problems regarding its application in software development projects. The method is cyclic and uses a probabilistic model to present project data to the *ScrumMaster*. Given its capacity to handle uncertainties and flexibility to modifications, *Bayesian Networks* were used to implement the model. The model was validated using scenarios to test it and the method through a case study in two projects in a company. The results show that using the method helps to detect problem in software development projects using *Scrum* with a positive cost-benefit and useful to guide the team to achieve excellence.

Keyworks: Agile, Bayesian Networks, Scrum

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Em 2008, El Emam e Koru [12] publicaram o resultado de um estudo empírico acerca da taxa de sucesso em projetos de desenvolvimento de software. Esse estudo foi realizado entre 2005 e 2007 e concluiu que entre 24% e 34% dos projetos de desenvolvimento de software falham: entre 11,5% e 15,5% são cancelados; e entre 16% e 22% são concluídos, mas os objetivos não são atingidos.

Esse estudo [12] conclui que há desentendimento entre os *desenvolvedores* e outras *partes interessadas*: há falta de comprometimento do gerenciamento sênior e de habilidades de gerenciamento. Dentre os projetos concluídos, mas cujo objetivos não são atingidos, o maior problema detectado é satisfazer o planejamento de tempo do projeto. Geralmente, os projetos não cumprem o cronograma. Esse estudo mostra que a principal causa disso é a imprecisão nas estimativas de duração das atividades do projeto.

Complementando as conclusões do estudo de El Emam e Koru, Boehm [7] identificou os seis principais motivos que causam a falha de projetos de software, os quais são: requisitos incompletos; falta de envolvimento do usuário; falta de recursos; expectativas irrealistas; falta de apoio executivo; e mudanças de requisitos e especificações. A maioria desses problemas é causada por falhas de comunicação e entendimento entre *desenvolvedores* e *clientes*. Portanto, para aumentar as chances de sucesso de projetos de software é de fundamental importância buscar soluções que proporcionem mecanismos para melhorar a comunicação entre os atores envolvidos.

Neste sentido, as metodologias ágeis são focadas na colaboração entre *desenvolvedores* e *clientes* e em responder às mudanças de requisitos [27], ou seja, essas metodologias, teorica-

mente, resolvem os principais problemas em projetos de software identificados por El Emam e Koru, e Boehm. Desta forma, processos e arcabouços de desenvolvimento de software ágeis, especialmente o *Scrum* [43; 37], têm se popularizado na indústria de desenvolvimento de software.

O *Scrum* é um arcabouço ágil iterativo e incremental que aperfeiçoa a previsibilidade e controle de riscos usando equipes auto-gerenciáveis e multifuncionais. Os três pilares do *Scrum* são: transparência, inspeção e adaptação. Ao final de cada iteração, é entregue um incremento funcional do produto para ser avaliado pelos *clientes*. Dessa forma, todas as partes envolvidas no processo de desenvolvimento do produto tornam-se cientes do estado atual da construção do mesmo, e inspeções e adaptações podem ser realizadas para que o produto correto seja construído ao final do projeto. O *Scrum* foca em satisfazer os *clientes* e maximizar o valor do produto. Além disso, por o *Scrum* ser um arcabouço, é necessário utilizar processos e técnicas adicionais para gerenciar projetos. Por exemplo, é comum complementar o *Scrum* utilizando técnicas de engenharia de *Extreme Programming* [10].

1.1 Problemática

A natureza iterativa e incremental do *Scrum* pressiona a equipe de desenvolvimento a melhorar as práticas de engenharia e os gerentes a otimizar o retorno de investimento [38]. Alguns *clientes* e *desenvolvedores* não compreendem ou não estão acostumados com alguns princípios do *Scrum*, tais como: colaboração entre pessoas de negócio e *desenvolvedores*, auto-organização e desenvolvimento incremental do produto. No *Scrum*, os *desenvolvedores* não são vistos apenas como recursos, e o papel dos gerentes do projeto é liderar - não gerenciar - a equipe de desenvolvimento. Schwaber [38] acredita que um dos maiores desafios para aplicar o *Scrum* ocorre por causa da mudança de alguns hábitos tradicionais de gerenciamento de projeto, tais como: qualidade do projeto não pode ser diminuída e gerentes não têm poder de controlar-e-comandar.

Dada a característica de fornecer transparência, a utilização correta do *Scrum* constantemente expõe os problemas das práticas e técnicas empregadas no projeto. Desta forma, o *Scrum* é difícil de ser aplicado pois expõe as disfunções da empresa. Por isto, é comum empresas, ao aplicarem o *Scrum*, ignorarem ou modificarem as regras do mesmo com o intuito

de tornarem os problemas invisíveis. Este comportamento ocorre pois resistência é encontrada quando mudanças são introduzidas na rotina de trabalho de qualquer empresa. Por outro lado, há diversos aspectos, tais como: regras do *Scrum*, práticas de engenharia e comportamento da equipe a serem considerados ao aplicar o *Scrum* e é difícil para o *ScrumMaster* (pessoa responsável por garantir que o Scrum seja entendido e aplicado) monitorá-los e prever a evolução da saúde do projeto caso algum aspecto seja corrigido. A utilização incorreta do *Scrum* pode ser causada por resistência a mudanças das partes envolvidas no projeto ou pela dificuldade de monitorar os diversos aspectos envolvidos no projeto.

1.2 Estado da Arte de Modelos de Processos

Uma forma de complementar o *Scrum* consiste em utilizar modelos de processos. Modelos de processos de software vêm se tornando populares na comunidade de engenharia de software [22]. Neste contexto, o objetivo é utilizar tecnologias, pessoas e ferramentas para trabalhar em conjunto com o intuito de melhorar as chances de sucesso de projetos de desenvolvimento de software. Esses modelos são focados no processo de desenvolvimento, manutenção e evolução de software. Eles podem representar processos da forma que estão implementados ou como planejados para implementação futura. Esses modelos são abstrações que representam apenas os fatores julgados importantes pelo desenvolvedor do modelo. Esses fatores podem ser definidos de forma empírica ou a partir da opinião de especialistas [28]. De acordo com um estudo realizado em 2008 [46], as técnicas mais utilizadas para modelar processos de software são *system dynamics* e simulação de eventos discretos. Algumas aplicações de modelos de processos de software são: auxiliar na tomada de decisões, no gerenciamento de risco e na detecção de potenciais problemas no processo de desenvolvimento da equipe.

Bai et al. [3] utilizaram uma técnica híbrida para modelar processos de desenvolvimento de software do ponto de vista das *partes interessadas* com o objetivo de aumentar a confiança na entrega do projeto. Os pesquisadores identificaram as *partes interessadas* e suas respectivas perspectivas na modelagem contínua do processo. Dentre as *partes interessadas*, pode-se citar: cliente, engenheiro de processo, especialista e gerente de processo. Dentre as perspectivas, pode-se citar: modelagem da força de trabalho, composição da equipe, análise

da justificativa de negócios do projeto e arquitetura do sistema. Para tal, os pesquisadores utilizaram técnicas de modelagem discretas e contínuas para que o modelo do processo possa se adaptar a mudanças internas do processo e causadas pelo ambiento externo. O modelo obteve sucesso quando aplicado ao ISPW-6 [21] utilizando Little-JIL [31].

Fan e Yu [13], Hearty et al. [16], Houston et al. [18], Büyüközkan e Ruan [8] e Jeet et al. [20] modelaram processos de software para auxiliar no gerenciamento de riscos. Fan e Yu usaram *redes Bayesianas* para construir um modelo que possibilita prever potenciais riscos, identificar fontes de riscos e auxiliar no ajuste dinâmico de recursos. Hearty et al. usaram *redes Bayesianas* para modelar o (*Extreme Programming*)*XP* e avaliar riscos. Além disto, o modelo proposto por Hearty et al. permite estimar, quantitativamente, o esforço necessário para finalizar um projeto. Houston et al. usaram *system dynamics* e *simulação estocástica* para modelar fatores de risco e simular seus efeitos para auxilar nas atividades de gerenciamento de risco. Büyüközkan e Ruan usaram um operador *fuzzy* especial - integral de Choquet - que permite modelar vários efeitos de importância e interações entre os riscos. Jeet et al. [20] usaram *redes Bayesianas* para estimar o atraso na entrega de um projeto de desenvolvimento de software dado o impacto da falta de produtividade da equipe de desenvolvimento. Jeet et al. utilizaram entrevistas e dados históricos para construir o modelo.

Abouelela e Benedicenti [1] e Jeet et al. [19] modelaram processos de software para auxiliar no gerenciamento da qualidade. Abouelela e Benedicenti usaram *redes Bayesianas* para prever a taxa de defeitos em uma liberação (*release*) de produto em um projeto *XP*. Além disto, o modelo proposto possibilita prever a data de finalização do projeto. Desta forma, o modelo proposto permite determinar se o projeto *XP* vai obter sucesso ou falhar. As entradas do modelo são: informações de requisitos; tamanho do time; uso de *programação em par*; uso de *desenvolvimento dirigido por testes*; disponibilidade do *cliente*; velocidade; e número de defeitos. Jeet et al. usou *redes Bayesianas* para detectar a quantidade de defeitos em um projeto de desenvolvimento de software. Para o modelo proposto, são usados a taxa de defeitos e julgamentos dos gerentes de projetos para prever e auxiliar no controle do número de defeitos.

Pendharkar et al. [33], Kouskouras e Georgiou [23] e Spasic e Onggo [41] modelaram processos de software para auxiliar no planejamento do projeto. Pendharkar et al. usaram *redes Bayesianas* para prever o esforço necessário para completar um projeto de desenvolvi-

mento de software. Para tal, o modelo combina conhecimento histórico do projeto e estimativas subjetivas de especialistas. Kouskouras e Georgiou construíram um modelo de evento de simulação discreta para estimar vários detalhes de um projeto de software, tais como: datas de entregas e métricas de qualidade. Desta forma, além desse modelo auxiliar no planejamento de projeto, ele auxilia no seu controle e monitoramento. Spasic e Onggo utilizaram *agent-based simulation* para modelar o processo de desenvolvimento de software de uma empresa avaliada como nível 3 CMMI e que segue a metodologia *Rational Unified Process*. O objetivo do modelo é estimar a data de finalização do projeto dados os componentes do sistema envolvidos e recursos humanos disponíveis. Spasic e Onggo validaram o modelo comparando os resultados calculados pelo mesmo com resultados reais de projetos passados e obtiveram sucesso.

Melis [28], Wu e Yan [45] e Kuppuswami et al. [24] modelaram processos para avaliar a influência do uso de algumas práticas *XP* em projetos *XP*. Melis usou *simulação de eventos discretos* e *system dynamics* para modelar o processo de desenvolvimento *XP* com o intuito de avaliar o uso de *programação em par* e *desenvolvimento dirigido por testes* na evolução de projetos *XP*. Wu e Yan usaram *system dynamics* para modelar o processo de desenvolvimento *XP* para avaliar a influência do *programação em par* no resultado de um projeto *XP* com o intuito de demonstrar para alunos de graduação a importância da prática. Kuppuswami et al. usaram *system dynamics* para modelar o processo de desenvolvimento *XP*, contendo todas as práticas [5], com o intuito de avaliar o esforço do desenvolvimento do projeto. O objetivo é verificar se o uso de *XP* diminui o custo para desenvolver o projeto, comparado com metodologias tradicionais.

Nagy et al. [30] usaram *redes Bayesianas* para construir um modelo para avaliar fatores chave de um projeto ágil e calcular a saúde do projeto. Esse modelo auxilia na tomada de decisões mais rapidamente, e dessa forma, permite detectar potenciais problemas mais cedo. O modelo proposto utiliza apenas o conhecimento de especialistas e os autores não deixam claro como esse conhecimento foi usado na construção da *rede Bayesiana*. Além disto, o modelo não foi validado.

1.3 Objetivos da Pesquisa

Neste trabalho propõe-se um método baseado em modelo probabilístico para auxiliar na detecção de problemas na aplicação do *Scrum* em projetos de desenvolvimento de software. O objetivo do método é que sua utilização exponha os problemas na aplicação do *Scrum* de forma a auxiliar o *ScrumMaster* a liderar a equipe na busca por excelência. A justificativa para a utilização do método é diminuir a probabilidade de projetos não alcançarem seus objetivos por falha na aplicação do *Scrum* e processos e técnicas auxiliares necessários. O método proposto utiliza a abordagem de modelos de processos para representar um projeto de desenvolvimento de software usando o *Scrum* e as técnicas e processos complementares necessários. A técnica escolhida para construir o modelo foi baseada em *Redes Bayesianas*. Desta forma, os objetivos específicos do método são:

1. construir modelo, utilizando *Rede Bayesiana*, para identificar problemas na aplicação do *Scrum*. O modelo deve considerar fatores chave na aplicação do *Scrum*, tais como: princípios e regras do *Scrum* e técnicas e processos complementares necessários. Além disto, dada a impossibilidade de representar todos os projetos *Scrum* de desenvolvimento de software, o modelo deve ser facilmente alterado para representar qualquer projeto *Scrum* de desenvolvimento de software;

2. se adequar aos princípios e regras do *Scrum*;

3. ser útil para auxiliar o *ScrumMaster* a guiar a equipe na busca por excelência;

4. ter custo-benefício de utilização positivo.

Como apresentado anteriormente, *Redes Bayesianas* foram utilizadas com sucesso para modelar incertezas em projetos de desenvolvimento de software e é uma técnica que possibilita modelar fatores e seus relacionamentos, a partir do grafo acíclico direcionado e as tabelas de probabilidade. Apesar desta pesquisa ter objetivos diferentes dos casos citados, enfrenta-se os mesmos problemas de modelar incertezas dada a subjetividade na definição de mensurar cada parâmetro do modelo inerentes em modelagem de processos. Dada a sua capacidade de modelar incertezas, capacidade de realizar previsões e lidar com dados não disponíveis, *Redes Bayesianas* foram utilizadas neste trabalho.

1.4 Relêvancia da Pesquisa

A contribuição principal dessa pesquisa é a proposição de um método pioneiro, validado em dois projetos em uma empresa, para auxiliar na detecção de problemas na aplicação do *Scrum* em projetos de desenvolvimento de software. Desta forma, dado que a utilização de metodologias ágeis, especialmente o *Scrum*, em projetos de desenvolvimento de software vem crescendo, essa pesquisa tem o potencial de ter impacto relevante na indústria de software, o que já vem sendo respaldado com publicações de impacto [34].

Outras contribuições promovidas por este trabalho podem ser destacadas, como a detecção de fatores chave em projetos de desenvolvimento de software utilizando o *Scrum* e uma nova metodologia para construir *redes Bayesianas* a partir da análise estatística dos dados coletados de especialistas por meio de um questionário, o que também já foi aceito como contribuição do trabalho [35]. Os fatores chave foram identificados a partir da experiência do pesquisador que tem mais de dois anos de atividade como *ScrumMaster* e possui as certificações de *Certified Scrum Master* e *Certified Scrum Product Owner* da *Scrum Alliance*, pesquisa na literatura e consultas a especialistas com experiência e trabalhando em diversos países.

1.5 Estrutura da Dissertação

Nesta dissertação, os capítulos foram organizados de modo a minimizar a dependência entre eles. Assim, o leitor pode optar por ler apenas os pontos que lhe interessa de acordo com a estrutura a seguir:

- No Capítulo 2, a fundamentação teórica da pesquisa é apresentada. Metodologia ágil, *Scrum* e *Redes Bayesianas* são introduzidos. Além disto, os conceitos de *Redes Bayesianas* utilizadas na pesquisa são apresentados.
- No Capítulo 3, o método proposto é apresentado, destacando suas etapas, cuidados necessários para aplicá-lo e limitações.
- No Capítulo 4, o modelo e seu processo de construção são apresentados.

- No Capítulo 5, a validação da modelagem a partir de testes em cenários, discussões e resultados da mesma são apresentadas.
- No Capítulo 6, a validação do método a partir de um estudo de caso em dois projetos em uma empresa, discussões e resultados associados são apresentados.
- Por fim, no Capítulo 7 são apresentadas as considerações finais, sendo discutidos resumidamente os principais tópicos elencados neste trabalho, bem como as ameaças à validade e trabalhos em andamento e sugestões para trabalhos futuros.

Além dos capítulos descritos anteriormente, são disponibilizados quatro apêndices. No Apêndice A, apresenta-se a lista de estados para todos os nós da *Rede Bayesiana*. No Apêndice B, apresentam-se as perguntas e respectivos glossários do questionário disponibilizado para coletar dados dos especialistas. No Apêndice C, apresentam-se os dados coletados a partir do questionário. No Apêndice D, apresenta-se a análise estatística do estudo empírico (aplicação do questionário).

Capítulo 2

Fundamentação Teórica

2.1 Metodologia Ágil

Em 2001, um grupo de engenheiros de software se reuniu para definir uma nova metodologia com princípios e valores que se adaptem à crescente taxa de mudanças de requisitos e expectativas dos clientes [44]. O resultado dessa reunião foi a criação da *Agile Alliance* e do *Manifesto Ágil* [27]. O *Manifesto Ágil* expressa os valores da metodologia ágil: indivíduos e interações mais que processos e ferramentas; software em funcionamento mais que documentação abrangente; colaboração com o cliente mais que negociação de contratos; e responder a mudanças mais que seguir um plano. Além disso, o *Manifesto Ágil* fornece doze princípios para guiar equipes ágeis:

- Satisfazer o cliente a partir da entrega contínua e adiantada de software com valor agregado.
- Mudanças nos requisitos são bem vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

- Construir projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- Software funcionando é a medida primária de progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Contínua atenção à excelência técnica e boa arquitetura aumenta a agilidade.
- Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.
- As melhores arquiteturas, requisitos e modelos emergem de equipes auto-organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

2.2 Scrum

O *Scrum* foi criado no início dos anos 90 e, atualmente, é o arcabouço ágil mais utilizado na indústria [43]. A proporção de projetos ágeis que utilizam *Scrum* com relação à outros arcabouços, processos e métodos ágeis é apresentada na Figura 2.1. Este arcabouço ágil foca em construir produtos complexos e pode ser complementado com processos e técnicas [39]. Por exemplo, é comum combinar *Scrum* com técnicas do *XP* [10] e com *Kanban* [2] - processo chamado de *Scrumban* [25]. O *Scrum* utiliza uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar riscos. Ao final de cada iteração - chamada de *sprint* -, um incremento funcional do produto é entregue para ser avaliado pelos *stakeholders*. Três pilares sustentam o *Scrum*: transparência, inspeção e adaptação. Ao final de cada iteração, é entregue um incremento funcional do produto para ser avaliado pelos *clientes*. Aspectos significativos do processo (definição de "pronto"e critérios de aceitação, por

exemplo) devem ser visíveis (transparentes) para os envolvidos. Os envolvidos devem inspecionar os artefatos e progresso relativo ao objetivo frequentemente para detectar variações indesejáveis. Caso algum desvio indesejável seja detectado no processo, o mesmo deve ser modificado para se adaptar às circunstâncias. Uma visão geral do *Scrum* é apresentada na Figura 2.2.

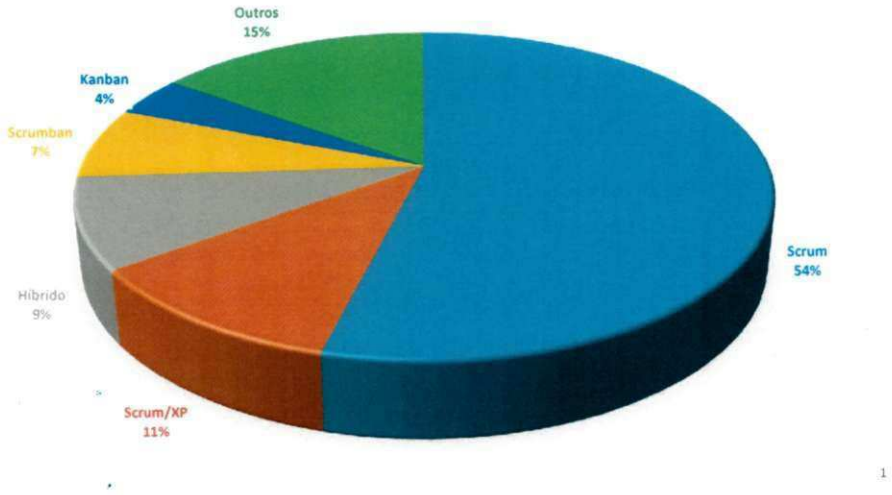

Figura 2.1: Proporção de projetos ágeis que utilizam *Scrum* com relação à outros arcabouços, processos e métodos ágeis

Os dois artefatos principais são: *backlog do produto* e *backlog da sprint*. O *backlog do produto* é uma lista ordenada que representa todas as funcionalidades, requisitos, melhorias e correções de bugs do produto. A ordenação é baseada em valor, risco, prioridade e necessidade. Além disto, o *backlog do produto* é evolutivo. Ou seja, ele se adapta às mudanças requisitadas pelos *clientes* durante a execução do projeto. O *backlog da sprint* é composto por itens do *backlog do produto* alocados para a iteração e o planejamento para alcançar o objetivo da iteração. Além dos artefatos já citados, no *Scrum* há artefatos de monitoramento

de progresso popularmente utilizados, tais como: *sprint burndown*, *release burndown graph* e *release burndown bar* [36].

Os três papéis essenciais são: *dono do produto (product owner)*, *ScrumMaster* e *desenvolvedor*. *O dono do produto* é responsável por maximizar o valor do produto e o trabalho dos desenvolvedores. Ele faz a interface entre a equipe técnica e a equipe de negócios envolvida no projeto e é o único responsável por gerenciar o *backlog do produto*. O *ScrumMaster* age como um líder-servidor e é responsável por garantir que as teorias, práticas e regras do *Scrum* sejam seguidas. O *desenvolvedor* é responsável por realizar qualquer trabalho necessário para entregar o incremento do produto ao final da iteração, tais como: construir arquitetura, programar e testar. Além destes papéis, na prática, há a figura do *cliente*. O *cliente* representa os interesses de negócio do projeto. Geralmente, um projeto *Scrum* é composto por uma equipe de *desenvolvedores* - entre cinco e nove integrantes -, um *ScrumMaster*, um *dono do produto* e diversos *clientes*.

No *Scrum* há quatro reuniões essenciais: *planejamento*, *diária*, *retrospectiva* e *revisão*. As *reuniões de planejamento* ocorrem no início de cada iteração com o objetivo de planejar o trabalho e objetivo da mesma. As *reuniões diárias* ocorrem diariamente com o objetivo de inspecionar o andamento da iteração e sincronizar o trabalho para diminuir os riscos. As *reuniões de revisão* ocorrem ao final de cada iteração com o objetivo de inspecionar o produto produzido na iteração e, se necessário, adaptar o backlog do produto às mudanças requisitadas pelos *clientes*. As *reuniões de retrospectiva* ocorrem após as reuniões de revisão com o objetivo de inspecionar a iteração com relação às pessoas, relacionamentos, processos e ferramentas; identificar problemas e criar um plano para aplicar as melhorias.

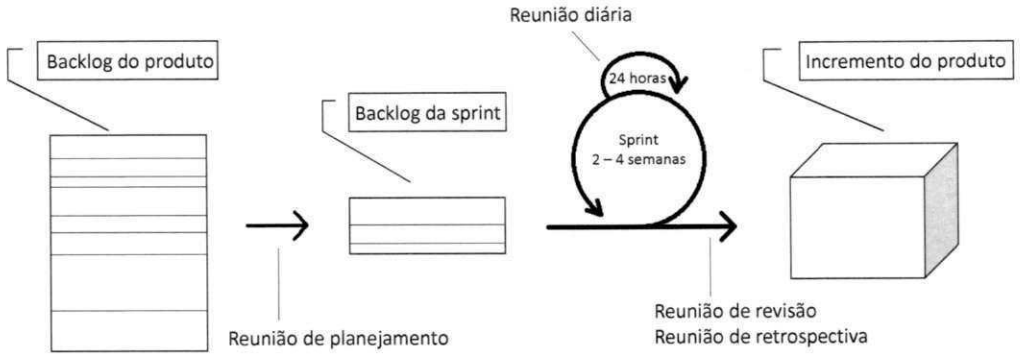

Figura 2.2: Visão geral do *Scrum*

2.3 Redes Bayesianas

Redes Bayesianas pertencem à família de modelos gráficos probabilísticos e são usadas para representar incertezas de um domínio [6]. Elas representam variáveis aleatórias e suas dependências condicionais a partir de um grafo acíclico direcionado (GAD). As variáveis aleatórias são representadas pelos nós e as dependências condicionais pelos arcos. Além disto, os nós são associados com funções de probabilidade que têm como entrada os conjuntos de valores dos nós pais e calculam a probabilidade da variável representada pelo nó. Os nós são representados por círculos. Os arcos são representados por setas. As funções de probabilidade, geralmente, são representadas por tabelas. Nota-se que apesar dos arcos representarem a direção da conexão causal entre as variáveis, informações podem propagar em qualquer direção no grafo [32]. Um exemplo de *rede Bayesiana* é apresentado na Figura 2.3.

Formalmente, uma *rede Bayesiana* B é um grafo acíclico direcionado que representa a distribuição conjunta de probabilidade sobre o conjunto de variáveis aleatórias V [15]. A rede é definida pelo par $B = \{G, \Theta\}$. G é o grafo acíclico direcionado cujo os nós $X_1, \ldots, X_n$ representam variáveis aleatórias, e os arcos representam dependências diretas

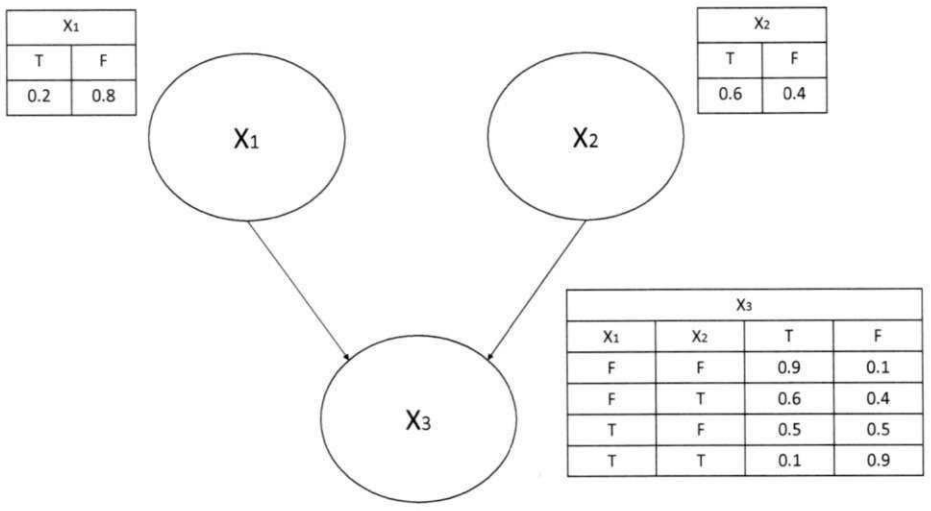

Figura 2.3: Exemplo de *rede Bayesiana*

entre essas variáveis. O grafo G codifica suposições de independência, nas quais cada variável X_i é independente dos seus *não-dependentes* dados seus *pais* em G. Θ representa o conjunto de parâmetros da rede. Esse conjunto contém o parâmetro $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$ para cada x_i em X_i condicionado por π_i, o conjunto de parâmetros de X_i em G. Equação 2.1 apresenta a distribuição conjunta definida por B sobre V .

$$P_B(X_1, \ldots, X_n) = \prod_{i=1}^{n} P_B(x_i|\pi_i) = \prod_{i=1}^{n} \theta_{X_i|\pi i} \tag{2.1}$$

2.4 Conceitos Aplicados à Pesquisa

O *Scrum* foi criado com a intenção de ser aplicado em projetos de desenvolvimento de software. Dada a subjetividade de definir e mensurar cada parâmetro do modelo inerente em modelagem de processos, para modelar a utilização do *Scrum* é necessário representar as incertezas. Dada sua capacidade de modelar incertezas, realizar previsões e lidar com dados não disponíveis, *Rede Bayesiana* é uma técnica ideal para tal. Os nós da *rede Bayesiana* representam fatores relevantes para projetos *Scrum* de desenvolvimento de software, os arcos representam os relacionamentos entre os fatores, e as funções de probabilidade quantificam os relacionamentos.

Capítulo 3

O Método

Como discutido no Capítulo 2, o *Scrum* promove uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar riscos. Há três papéis nesse arcabouço: *dono do produto (product owner)*, *ScrumMaster* e *desenvolvedor*. Destes três, o *ScrumMaster* é o responsável por garantir que as teorias, práticas e regras do *Scrum* sejam seguidas a partir de liderança servidora. Além disto, ele deve guiar a equipe para garantir a qualidade dos entregáveis. Ao final de cada iteração, além da *reunião de revisão* cujo objetivo é validar o incremento do produto produzido durante a iteração, há uma reunião para inspecionar a iteração com relação às pessoas, relacionamentos, processos e ferramentas; identificar problemas e criar um plano para aplicar as melhorias. Essa reunião chama-se *reunião de retrospectiva* e é liderada pelo *ScrumMaster*. Ou seja, o dia-a-dia do *ScrumMaster* resume-se a garantir que as teorias, práticas e regras do *Scrum* sejam seguidas além de liderar a equipe à excelência.

O método proposto utiliza um modelo probabilístico, é cíclico e composto por quatro etapas: (i) análise do modelo, (ii) alimentação do modelo, (iii) análise das saídas do modelo e (iv) aplicação de ações corretivas e preventivas. O método é apresentado na Figura 3.1. Apesar do método utilizar um modelo probabilístico, a complexidade do modelo não é perceptível pelo usuário, ou seja, os usuários não precisam de conhecimentos específicos sobre as tecnologias utilizadas para construir o modelo. A duração dos ciclos deve ser alinhada com as iterações, mas não precisa ser a mesma. Por exemplo, a duração de um ciclo da aplicação do método pode ser igual à duração da iteração, ou maior. Dado que o método deve ser aplicado em um processo empírico (no caso, o *Scrum*), a definição de sua duração do ciclo deve ser empírica, ou seja, a duração do ciclo deve ser definida durante sua execução

baseando-se no que a equipe do projeto considera como a que tem melhor retorno de investimento. O importante é que a etapa (iii) seja realizada durante uma *reunião de retrospectiva*. O esforço adicional para a aplicação do método ocorre durante as etapas (i) e (ii), dado que a etapa (iii) pode ser considerada como um complemento ou técnica adicional utilizada na *reunião de retrospectiva*, e a etapa (iv) faz parte do dia-a-dia do *ScrumMaster*.

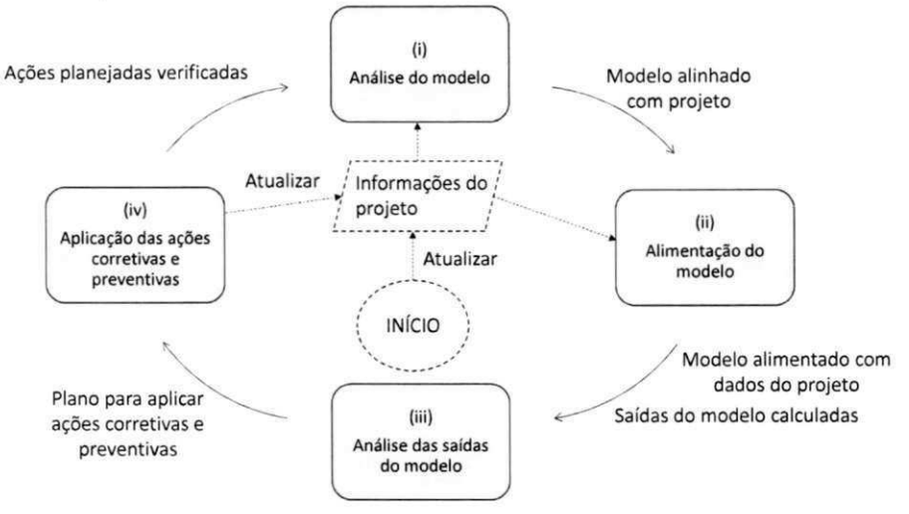

Figura 3.1: Fluxo com etapas do método

Sua aplicação deve ser liderada pelo *ScrumMaster*, o qual deve solicitar, sempre que necessário, auxílio de outros membros da equipe durante qualquer etapa. Desta forma, a aplicação do método é um processo complementar ao *Scrum* e alinha-se com o dia-a-dia de *ScrumMasters*, pois nas etapas (i) e (ii) auxiliará a analisar o estado do projeto; na etapa (iii), o modelo servirá como uma ferramenta adicional na execução de *reuniões de retrospectiva*; e na etapa (iv) o *ScrumMaster* continuará liderando a aplicação de melhorias definidas nas *reuniões de retrospectiva*.

O objetivo do modelo é prover informações para o *ScrumMaster* de forma a ajudá-lo

a detectar os problemas na aplicação do *Scrum* e ter informações suficientes para guiar a equipe e aumentar as chances de sucesso do projeto. Dado que, dependendo das características do projeto, diferentes técnicas e processos podem ser utilizados para complementar o *Scrum*, torna-se impossível modelar todos os projetos de software que utilizam *Scrum*. Nesta pesquisa, técnicas e processos complementares ao *Scrum* considerados melhores práticas foram identificados e adicionados às regras e princípios do *Scrum* para que um modelo genérico fosse construído. Mais detalhes de como as melhores práticas foram identificadas e o modelo construído são apresentadas no Capítulo 4. Desta forma, dado que o modelo é genérico, seu objetivo é que possa ser facilmente alterado para se adequar às características específicas de qualquer projeto de desenvolvimento de software que utilize o *Scrum*.

Durante (i) a análise do modelo, a etapa na qual inicia-se o processo, procura-se detectar se o esse necessita ser alterado para se adequar ao projeto. É recomendado que informações relacionadas aos princípios e regras do *Scrum* não sejam alterados, e sim, apenas processos e técnicas complementares utilizados no modelo genérico. Mais informações sobre como alterar o modelo encontram-se no Capítulo 4. Essa etapa pode ser realizada durante uma *reunião de retrospectiva*, ou fora dela. Caso seja realizada fora da *reunião de retrospectiva*, o *ScrumMaster* deve assegurar a participação de outros membros do projeto para que essa etapa seja realizada com sucesso. Os dados desta etapa são apresentados na Figura 3.2.

Durante (ii) a alimentação do modelo, o modelo validado na etapa anterior é alimentado com dados que refletem a situação atual do projeto. O ideal é que todas as entradas do modelo estejam com valores de seus estados devidamente preenchidos. Para as entradas que puderam ser verificadas, uma evidência deve ser inserida. Para as entradas que não puderam ser verificadas, as mesmas deve ser alimentadas com incerteza igual para todas as possíveis condições. Da mesma forma que a etapa anterior, essa etapa pode ser realizada durante uma *reunião de retrospectiva*, ou fora dela. Caso seja realizada fora da *reunião de retrospectiva*, o *ScrumMaster* deve assegurar a participação de outros membros do projeto para que essa etapa seja realizada com sucesso. Os dados desta etapa são apresentados na Figura 3.3.

Durante (iii) a análise dos dados, que deve ser realizada durante a *reunião de retrospectiva*, a situação atual do projeto é discutida baseada nas saídas calculadas do modelo com os membros da equipe. O objetivo desta etapa segue um dos objetivos principais das *reuniões de retrospectiva*: definir ações corretivas e preventivas que devem ser colocadas em prática para

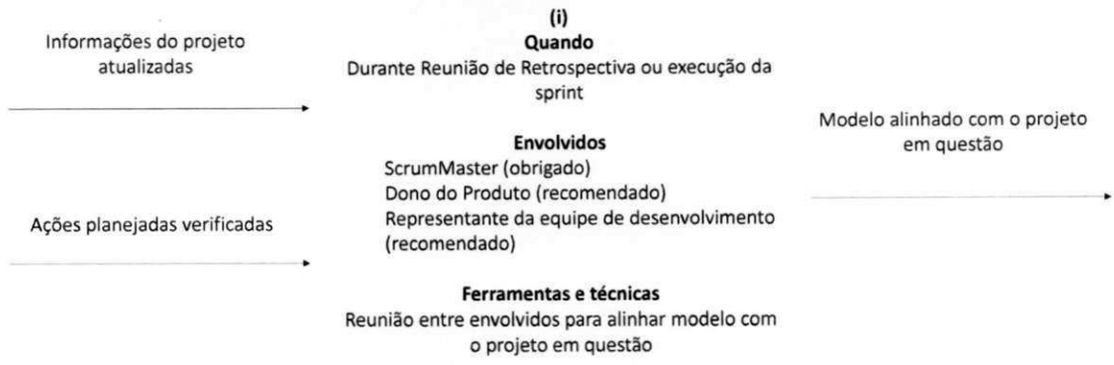

Figura 3.2: Descrição da Primeira Etapa do Método

aumentar a probabilidade de sucesso do projeto. Desta forma, esta etapa pode ser considerada como um complemento ou técnica adicional na execução de *reuniões de retrospectiva*. Além disto, dado que o modelo é capaz de fazer previsões do impacto na saúde do projeto, algumas entradas podem ser alteradas para que se avalie o impacto na saúde do projeto. Esta característica do modelo pode auxiliar na priorização das ações corretivas e preventivas. Por exemplo, dado que a equipe identifique cinco fatores que precisem de ações, esses fatores podem ter seus valores alterados, isoladamente, no modelo para que o impacto na saúde do projeto possa ser calculado. Por fim, um plano de execução das ações deve ser criado. Os dados desta etapa são apresentados na Figura 3.4.

Finalmente, o *ScrumMaster* é responsável por liderar (iv) a execução das ações definidas. Dado que utilizando ou não o método proposto nesta dissertação, um dos objetivos das *reuniões de retrospectiva* é definir um plano de ações corretivas e preventivas e o *ScrumMaster* é responsável por liderar a execução das plano de ações, a execução desse passo não

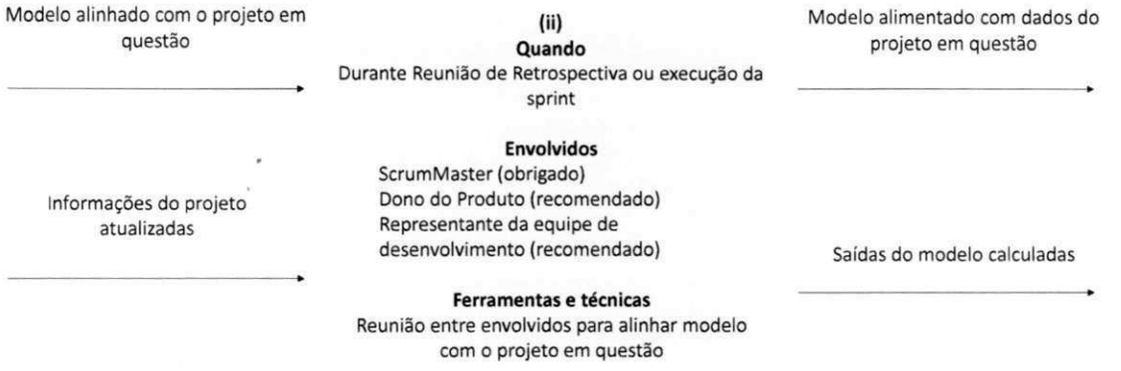

Figura 3.3: Descrição da Segunda Etapa do Método

adiciona esforço no dia-a-dia do *ScrumMaster*. A verificação das ações planejadas exigirá a atualização do estado do projeto que servirá para (i) análise do modelo e (ii) alimentação do modelo, no próximo ciclo. Os dados desta etapa são apresentados na Figura 3.5.

Os custos associados à implantação do método decorrem da necessidade de familiarização dos usuários com o método e o modelo probabilístico. Como citado anteriormente, a complexidade técnica do modelo não é perceptível pelos usuários, ou seja, não influencia nos custos. Por outro lado, é necessário treinamento para cada etapa do método. Além disso, é necessário conhecimento suficiente para compreender as entradas e saídas do modelo, para que o mesmo seja alimentado e interpretado corretamente. Dado que o modelo pode ser modificado, é necessário conhecimento o suficiente para tal. Os detalhes sobre o conhecimento necessário para modificação do modelo são apresentados no Capítulo 4.

Dado que o modelo utiliza dados que avaliam as pessoas envolvidas no projeto, alguns cuidados devem ser tomados ao se utilizar esse método. Dado que há informações relativas

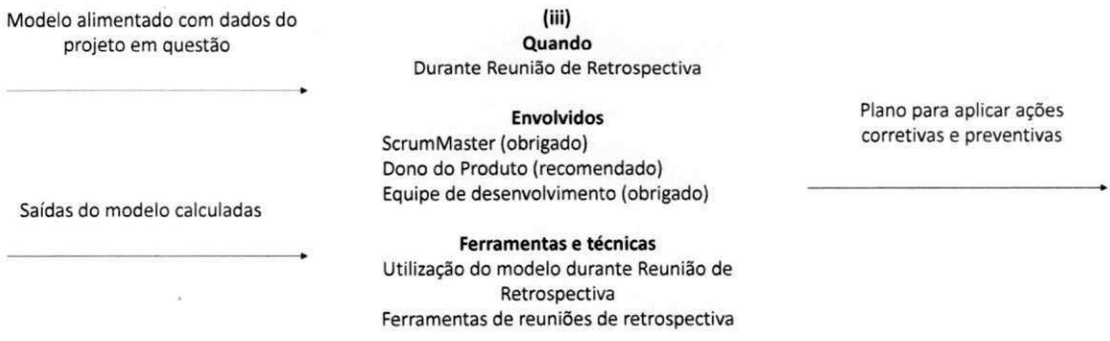

Figura 3.4: Descrição da Terceira Etapa do Método

ao desempenho da equipe, o *ScrumMaster* deve ter sensibilidade ao coletar dados para alimentar o modelo e ao apresentar os dados coletados e calculados pelo o mesmo para evitar constrangimentos e conflitos.

As limitações do método seguem as limitações do modelo: aplica-se apenas para projetos de desenvolvimento de software que utilizam *Scrum* composto por apenas uma equipe de desenvolvimento e apenas considera fatores relacionados à utilização do *Scrum* para monitorar a saúde do projeto e calcular sua probabilidade de sucesso. Problemas relacionados à recursos, condições econômicas da empresa, condições de mercado, entre outros fatores não relacionados à utilização do *Scrum* não são considerados.

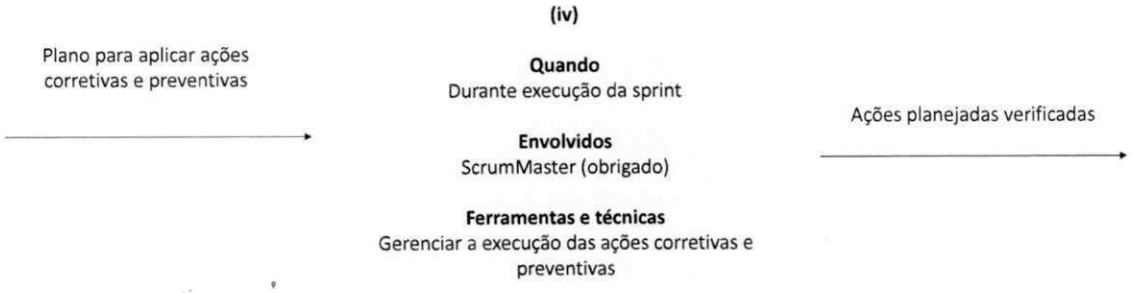

Figura 3.5: Descrição da Quarta Etapa do Método

Capítulo 4

Modelo Probabilístico Utilizado pelo Método

Trata-se de uma *Rede Bayesiana* que representa um projeto *Scrum* de desenvolvimento de software composto por uma equipe. O principal objetivo do modelo é identificar problemas na aplicação *Scrum* e guiar a equipe com o intuito de aumentar a probabilidade de sucesso do projeto. O modelo engloba princípios e regras do *Scrum* e técnicas e processos considerados como melhoras práticas para sua complementação. O modelo não engloba problemas que podem ocorrer no projeto que não estão diretamente relacionados à utilização *Scrum*. Como exemplos de fatores que não são englobados pelo modelo, pode-se citar: mudanças na demanda de negócios, estabilidade econômica da empresa, gerenciamento de aquisições e perda de membros da equipe. O modelo é apresentado na Figura 4.1.

O modelo apresentado neste trabalho é baseado nas regras e princípios do *Scrum* e melhores práticas na indústria, mas nem todas as equipes utilizam o *Scrum* da forma pura e seguem as melhores práticas identificadas. Desta forma, o modelo, na sua forma original, não representa todas as equipes que utilizam *Scrum* em projetos de desenvolvimento de software. Por outro lado, dado que a complexidade do modelo não é perceptível para o usuário, ele pode ser modificado, com baixo custo, para se adaptar às necessidades das equipes.

Caso o modelo precise ser alterado, recomenda-se que: (i) o grafo seja modificado para refletir as alterações, (ii) para as Tabelas de Probabilidade dos Nós (TPN) afetadas pelas modificações, os nós pais sejam ordenados com relação à magnitude relativa da influência sobre o nó filho de forma decrescente, (iii) o Algoritmo 1 seja utilizado para gerar as expres-

sões ponderadas, (iv) as expressões ponderadas sejam utilizadas na ferramenta utilizada para construir a *Rede Bayesiana* para gerar as respectivas TPN. Dessa forma, compreender grafos e expressões ponderadas são os únicos conhecimentos técnicos necessários para alterar o modelo, ou seja, assume-se que qualquer membro da equipe de um projeto de software tem conhecimento necessário para tal sem precisar de treinamento específico.

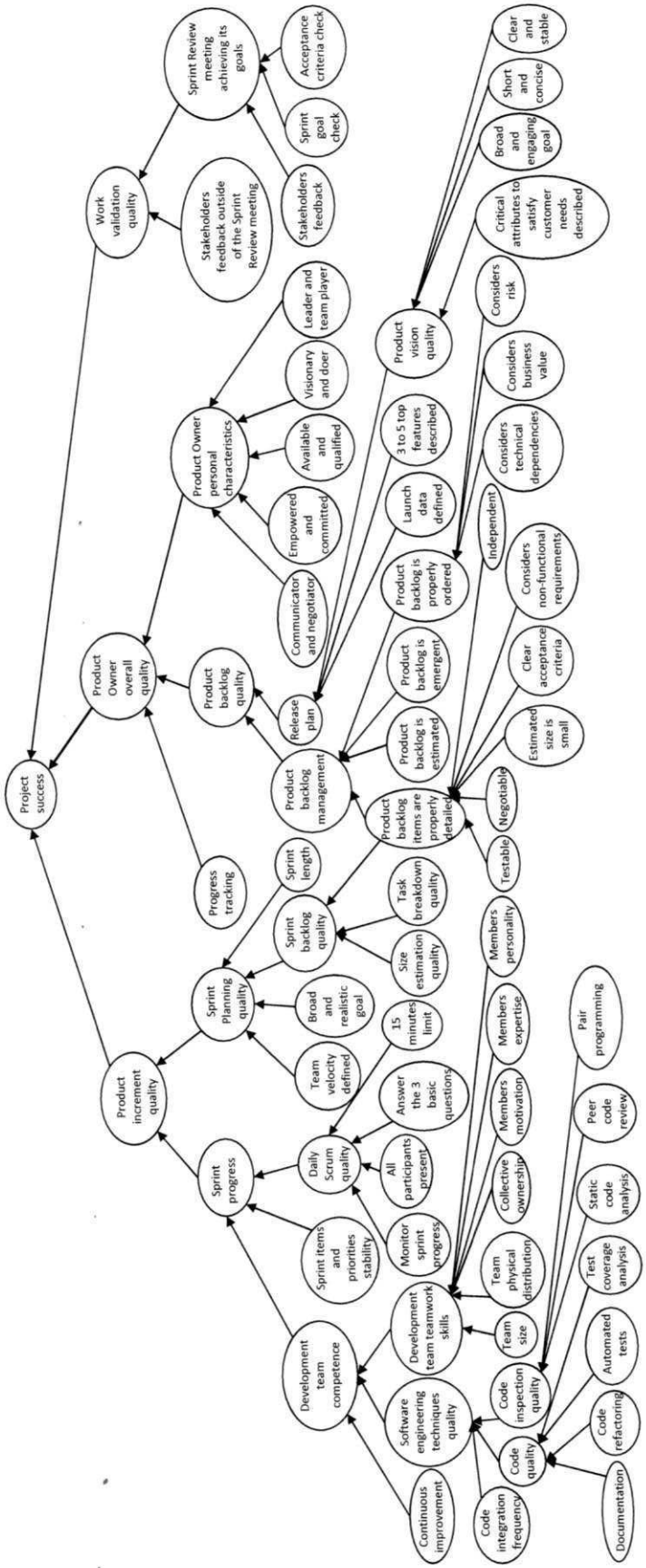

Figura 4.1: Modelo completo

4.1 Construção da *Rede Bayesiana*

O problema de definir o modelo pode ser dividido em: identificar os fatores relevantes para projetos *Scrum* de desenvolvimento de software e seus relacionamentos, e quantificar os relacionamentos. O GAD que compõe a *Rede Bayesiana* representa os fatores relevantes identificados e seus relacionamentos. As TPN que compõem a *Rede Bayesiana* representam a intensidade relativa entre os relacionamentos que apontam para o mesmo nó. Desta forma, o problema de construir a *Rede Bayesiana* pode ser dividido em: construir o GAD e construir as TPN. Para construir o GAD, a versão inicial foi definida a partir de resultados de pesquisa na literatura. Após a definição do GAD inicial, este foi apresentado para um grupo de especialistas e foi refinado. Para construir as TPN, divulgou-se um questionário para coletar dados de especialistas ao redor do mundo. Os dados coletados foram estatisticamente analisados e utilizados para ordenar os relacionamentos de cada TPN dado suas magnitudes relativas e gerar expressões ponderadas. Finalmente, as expressões ponderadas foram utilizadas para gerar as TPN. O processo utilizado para solucionar esses problemas é apresentado na Figura 4.2.

Figura 4.2: Processo para construção do modelo

4.1.1 Construção do Grafo Acíclico Direcionado (GAD)

No GAD, cada nó representa um fator relevante identificado, e há um arco entre os nós sempre que eles se relacionarem. Nos arcos, o fator sendo influenciado é o apontado pela seta. Além disto, cada fator é composto por possíveis estados com probabilidades associadas. DestFa forma, cada fator representa um conjunto de tuplas $N = \{(s_1, p_1), \ldots, (s_{|N|}, p_{|N|})\}$, em que s_i é um possível estado e p_i é a probabilidade associada ao estado. O conjunto de fatores é representado por $F = \{N_1, \ldots, N_{|F|}\}$. O conjunto de arcos é representado por $R = \{(N_j, N_k) \mid N_j \subset F \wedge N_k \subset F\}$, em que N_j é o ponto inicial e N_k é o ponto final do arco. Desta forma, o problema resume-se em identificar todos os elementos de F e R. Para identificar todos os elementos do conjunto F, necessita-se identificar todos os fatores N_a, e para cada fator N_a, identificar todos os possíveis estados s_i e suas probabilidades associadas p_i, em que $a \leq |F|$ e $i \leq |N_a|$. Finalmente, para identificar todos os elementos de R, necessita-se identificar todos f_j e f_k, em que f_j e $f_k \in F$.

Para definir o GAD, dividiu-se o problema em dois subproblemas: identificar os elementos de F e R, e identificar os elementos de N. O primeiro subproblema refere-se a identificar os nomes dos fatores para definir os nós do GAD e identificar os relacionamentos entre os fatores para conectar os nós. O segundo subproblema refere-se a identificar os possíveis estados de cada nó do GAD e as probabilidades associadas a cada estado.

Para resolver o primeiro subproblema, foi realizada pesquisa na literatura para identificar os elementos iniciais de F e R. Ao encerrar a pesquisa, a lista inicial de fatores foi apresentada para onze especialistas que a refinaram em discussões individuais com o pesquisador.

Com a identificação dos elementos finais de F e R, foi possível identificar os elementos N. Segundo os especialistas, três estados por fator é o suficiente para que os fatores sejam monitorados. Desta forma, para cada fator (elemento de F) identificado, definiu-se três possíveis estados, que seguem uma escala ordinal.

Pesquisa na Literatura

Inicialmente, identificou-se as variáveis (elementos) dependentes de F. E, a partir das variáveis dependentes, refinou-se a definição de F até que todas as variáveis dependentes e independentes fossem identificadas.

Dado que o modelo representa um projeto e deve ser utilizado para aumentar suas chances de sucesso, definiu-se *Project success* como o nó de mais alto nível da *rede Bayesiana*. Desta forma, *Project success* é o nó de saída da *rede Bayesiana* e representa as chances de sucesso do projeto. Para este nó, definiu-se três estados: *Bad*, *Moderate* e *Good*. Quanto maior a probabilidade de que o estado de *Project success* seja *Good*, maiores as chances de sucesso do projeto sendo avaliado.

Há quatro dimensões nos projetos que podem ser gerenciadas: escopo, custo, qualidade e tempo [29]. Destas dimensões, o *Scrum* engloba: escopo, qualidade e tempo. Segundo Pichler [36], em projetos *Ágeis*, a data final do projeto deve ser fixada no seu início. Apesar disto, segundo o *Manifesto Ágil* [27], mudanças no escopo durante o projeto são bem-vindas. Como, em projetos *Ágeis*, o escopo é alterado frequentemente, cronogramas não são criados. Desta forma, no *Scrum*, há três indicadores que podem ser analisados para medir o progresso do projeto dado as características do processo: qualidade do produto desenvolvido durante as *sprints*, da validação do incremento do produto ao final de cada *sprint* e do trabalho do *dono do produto*. O *dono do produto* responsabiliza-se pelo escopo de cada *sprint* e a sequência na qual as funcionalidades do produto são implementadas. Sua função é maximizar o valor do produto e do trabalho da equipe. Os *desenvolvedores* são responsáveis por implementar o produto durante a *sprint* e, juntamente com os outros membros da equipe, responsáveis pela sua qualidade. Ao final das *sprints*, durante a *reunião de revisão*, os *clientes* e *dono do produto* são responsáveis por validar o trabalho desenvolvido durante a *sprint* [39]. Este conjunto de atividades engloba o gerenciamento de qualidade, escopo e tempo de um projeto *Scrum*. Desta forma, identificou-se mais três elementos de F. A *rede Bayesiana* resultante é apresentada na Figura 4.3.

Os elementos encontrados e apresentados na Figura 4.3 podem ser refinados. No *Scrum*, como dito anteriormente, o trabalho é validado pelo *dono do produto* e *clientes* durante as *reuniões de revisão*. Durante esta reunião, o *dono do produto* verifica se o trabalho realizado durante a *sprint* está conforme o esperado e, os *clientes*, colaborativamente com a equipe, analisam se o produto correto está sendo construído [39]. Desta forma, adicionam-se dois elementos em F: *Acceptance criteria check* e *Stakeholders feedback*, ambos filhos do nó *Work validation Quality* na *rede Bayesiana*.

Para que os *clientes* e o *dono do produto* possam guiar o desenvolvimento do produto,

Figura 4.3: Quatro primeiros nós identificados da *rede Bayesiana*

é necessário que os *desenvolvedores* produzam um incremento do produto ao final de cada *sprint* [39]. O produto desenvolvido ao final da *sprint* é resultado das atividades executadas durante a *sprint* e durante o seu planejamento. Desta forma, adicionam-se dois elementos em F: *Sprint Planning* e *Sprint progress*, ambos filhos do nó *Product increment quality*.

O planejamento da *sprint* ocorre durante a *reunião de planejamento*. Durante essa reunião, os itens do *backlog da sprint*, que são extraídos do *backlog do produto*, são negociados entre os *dono do produto* e os *desenvolvedores* [39]. A qualidade dos itens alocados para a *sprint* dependem da qualidade dos itens descritos no *backlog do produto*. Além disto, a capacidade da equipe de transformar os itens do *backlog do produto* em tarefas técnicas e de estimá-los também influenciam na qualidade do *backlog do produto*. A *velocidade* da equipe é um indicador que pode auxiliar a equipe durante a *reunião de planejamento*, pois dá a equipe informações sobre a quantidade de trabalho que essa é capaz de completar por *sprint* [10]. Finalmente, o *objetivo da sprint*, que deve ser amplo e realista, é definido durante a *reunião de planejamento* para dar foco à equipe durante a *sprint* [36]. Desta forma, adicionam-se seis elementos em F: *Broad and realistic goal*, *Team velocity* e *Sprint backlog quality*, em que todos são filhos do nó *Sprint Planning quality*; e, *Product backlog items are properly detailed*, *Size estimation quality* e *Task breakdown quality*, onde todos são filhos do nó *Sprint backlog quality*.

Durante a *sprint*, há três fatores principais que influenciam seu progresso: *reuniões diárias*, a estabilidade do *backlog da sprint* e a competência da equipe de desenvolvimento. As *reuniões diárias* são essenciais para satisfazer o princípio *Scrum* que afirma que a equipe deve constantemente inspecionar-se e adaptar-se [39]. Além disto, como dito anteriormente,

no *Scrum*, o escopo da *sprint* é definido na *reunião de planejamento*. Durante a *sprint*, apenas as tarefas técnicas podem ser alteradas pelos *desenvolvedores*. O *dono do produto* ou outras *partes interessadas* não podem interferir no escopo da *sprint* durante sua execução [10]. Finalmente, a competência da equipe define a qualidade das funcionalidades implementadas. Desta forma, adicionam-se três elementos em F: *Daily Scrum quality*, *Sprint items and priorities stability* e *Development team competence*, ambos filhos do nó *Sprint progress*.

As *reuniões diárias* têm as funções de melhorar a comunicação, minimizar a necessidade de outra reuniões, e ajudar na identificação e remoção de impedimentos [39]. Além disto, essas reuniões promovem tomadas rápidas de decisões e expande o conhecimento de todo o projeto na equipe. Elas são limitadas em quinze minutos e não devem ser vistas apenas como reuniões de verificação de progresso. Durante essas reuniões, todos os membros da equipe devem estar presentes, explicar o que foi feito desde a última reunião, o que vai ser feito até a próxima reunião e levantar impedimentos [10]. Desta forma, adicionam-se três elementos em F: *15 minutes limit*, *All participants present* e *Answer the 3 basic questions*, em que todos são filhos do nó *Daily Scrum quality*.

Dado que no *Scrum*, a *equipe de desenvolvimento* é auto-gerenciável e precisa produzir o produto de forma incremental, sua competência depende da sua capacidade de trabalhar em equipe, suas práticas de engenharia de software e sua capacidade de melhorar continuamente [10]. Sua capacidade de trabalhar em equipe depende da: personalidade dos membros, conhecimento técnico dos membros, motivação da equipe, distribuição física da equipe, número de membros e capacidade dos membros em compartilhar o código e design do software. As práticas de engenharia podem ser resumidas em: técnicas de validação (ou inspeção), qualidade e integração do código. Desta forma, adicionam-se doze elementos em F: *Development team teamwork skills*, *Software engineering techniques quality* e *Continuous improvement*, onde todos são filhos do nó *Development team competence*; *Members personality*, *Members expertise*, *Members motivation*, *Team physical distribution*, *Team size* e *Collective ownership*, onde todos são filhos do nó *Development team teamwork skills*; e, *Code inspection quality*, *Code quality* e *Code integration frequency*, em que todos são filhos do nó *Software engineering techniques quality*.

Cohn [10] sugere que equipes *Scrum* utilizem algumas práticas de engenharia de software

da metodologia *Extreme Programming* [5]. Para inspecionar o código, recomenda-se que as equipes utilizem *programação em pares*. Por outro lado, muitas equipes *Ágeis* utilizam *revisão de código pelos pares* [40]. Para garantir a qualidade do código, recomenda-se que a equipe utilize *Desenvolvimento Dirigido por Testes* [10]. Caso a equipe não utilize essa metodologia, ela deve implantar outro processo que inclua automação de testes, refatoramento do código e evolução de testes para garantir cobertura adequada do código. Desta forma, adicionam-se cinco elementos em F: *Pair programming* e *Peer code review*, onde ambos são filhos do nó *Code inspection quality*; e, *Code refactoring*, *Automated tests* e *Test coverage analysis*, todos filhos do nó *Code quality*.

O *dono do produto* é o único responsável pelo gerenciamento do escopo e todos na organização devem respeitar suas decisões [39]. Além de ter características pessoais adequadas, o *dono do produto* precisa garantir a qualidade do *backlog do produto* e acompanhar o andamento do projeto. Pichler [36] recomenda que o *dono do produto* seja: visionário e cometedor, líder e jogador de equipe, comunicador e negociador, com autonomia e comprometido, e disponível e qualificado. Desta forma, adicionam-se oito elementos em F: *Product owner personal characteristics*, *Product backlog quality* e *Progress tracking*, onde todos são filhos do nó *Product owner overall quality*; e, *Visionary and doer*, *Leader and team player*, *Empowered and committed*, *Available and qualified* e *Communicator and negotiator*, em que todos são filhos do nó *Product Owner personal characteristics*.

A qualidade do *backlog do produto* depende do seu gerenciamento e da qualidade do *plano de lançamento*. O *backlog do produto* deve ser emergente, estimado no nível necessário, ordenado e ter seus itens detalhados de forma apropriada [39]. O *plano de lançamento* depende da qualidade da *visão do produto*, e precisa ter entre três e cinco das principais funcionalidades descritas e a data de lançamento definida. Por sua vez, a *visão do produto* precisa ter um objetivo amplo e envolvente, ser claro e estável, curto e conciso e os atributos críticos necessitam satisfazer as necessidades dos clientes [36]. Desta forma, adicionam-se treze elementos em F: *Product backlog management* e *Release plan quality*, em que ambos são filhos do nó *Product backlog quality*; *Product backlog is emergent*, *Product backlog items estimated*, *Product backlog properly ordered*, e *Product backlog items are properly detailed*, em que todos são filhos do nó *Product backlog management*; *Launch date defined*, *Product vision quality* e *3 to 5 top items described*, em que todos são filhos do nó *Release*

plan quality; e, *Critical attributes to satisfy the customer needs described*, *Broad and engaging goal*, *Short and concise*, e *Clear and stable*, em que todos são filhos do nó *Product vision quality*.

Para que o *backlog do produto* seja ordenado corretamente, valor de negócio, riscos e dependências técnicas devem ser considerados [39]. Além disto, os itens do *backlog do produto* devem ser negociáveis, testáveis, os itens no topo da lista devem ser pequenos o suficiente para serem implementados em uma *sprint*, ter critérios de aceitação completos e claros, independentes e considerar requisitos não-funcionais [10]. Desst forma, adicionam-se nove elementos em F: *Considers technical dependencies*, *Considers risk* e *Considers business value*, onde todos são filhos do nó *Product backlog is properly ordered*; e, *Negotiable*, *Testable*, *Estimated size is small*, *Independent*, *Clear acceptance criteria* e *Considers non-functional requirements*, em que todos são filhos do nó *Product backlog items are properly detailed*.

Com os itens identificados, concluiu-se a revisão na literatura e construiu-se a primeira versão do modelo. Neste estudo, identificou-se sessenta e sete variáveis: vinte e dois, dependentes; e, quarenta e seis independentes.

Avaliação de Especialistas

Onze especialistas foram consultados para avaliar os resultados da pesquisa na literatura. Nesse grupo incluem-se profissionais com experiência como *ScrumMaster*, *dono do produto*, gerentes de projetos *Ágeis*, *Agile coach*, consultores e líderes de equipe em empresas localizadas no Brasil, Estados Unidos da América, Europa e Índia. As consultas foram realizadas em discussões um-a-um entre o pesquisador e um especialista até se chegar em um consenso da lista final de itens e seus respectivos relacionamentos.

Algumas mudanças no modelo ocorreram depois da avaliação dos especialistas. Segundo os especialistas, em projetos reais, é muito comum *clientes* darem retorno com relação ao produto fora da *reunião de revisão*. Além disso, apesar de não ser regra do *Scrum*, é comum o *dono do produto* definir um objetivo para cada *sprint*, como é recomendando por Pichler [36]. Desta forma, adicionou-se o nó *Sprint review meeting achieving its goals*, em que o mesmo é filho do nó *Work validation quality*. Modificou-se o pai dos nós *Acceptance criteria check* e *Stakeholders feedback* de *Work validation work* para *Sprint review meeting achieving its goals*. Adicionou-se o nó *Sprint goal check*, onde o mesmo é filho do nó *Sprint review*

meeting achieving its goals. Finalmente, adicionou-se o nó *Stakeholders feedback outside of the Sprint review meeting*, em que o mesmo é filho do nó *Work validation quality*. Estas mudanças são apresentadas na Figura 4.4.

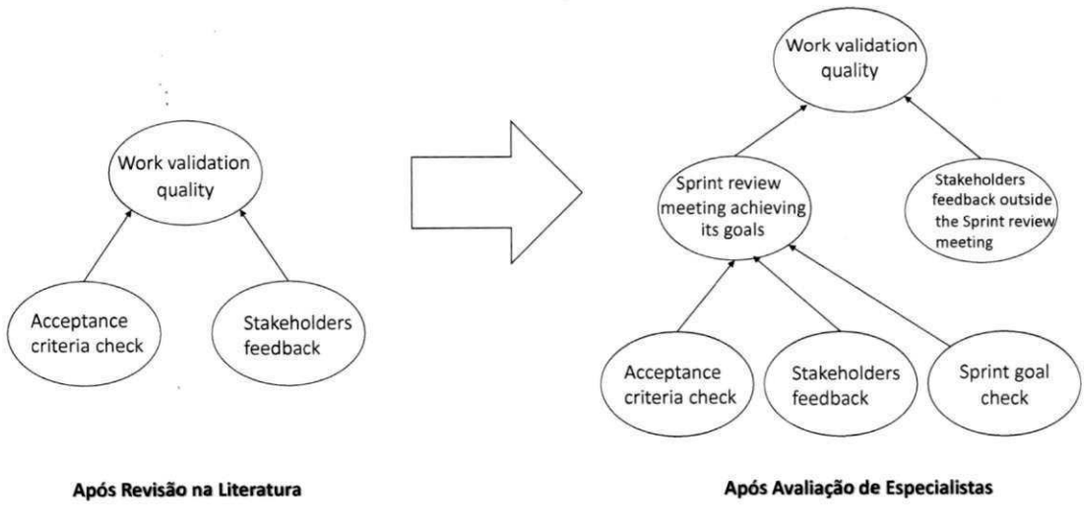

Figura 4.4: Mudanças no nó *Work validation quality*

Além disto, segundo os especialistas, a documentação do código é um fator que deve ser considerado na qualidade do código. Como o escopo é alterado frequentemente em projetos *Ágeis*, manter o código bem documentado facilita as suas mudanças. Outra consideração dos especialistas é que tamanho da *sprint* deve ser um fator considerado no planejamento da *sprint*. No *Scrum*, as *sprints* devem durar no máximo quatro semanas [39], e quanto mais alterações no escopo são esperadas, menor deve ser a *sprint*. Desta forma, adicionou-se dois nós na *rede Bayesiana*: *Documentation*, filho do nó *Code Quality*; e, *Sprint length*, filho do nó *Sprint planning quality*. Após a avaliação dos especialistas, o modelo ficou composto de setenta e três variáveis: vinte e três, dependentes; e, cinquenta independentes.

Definição dos Estados

Como dito anteriormente, além de auxiliar na definição dos nós e arcos da DAG, os especialistas foram consultados com relação ao número de estados apropriados para cada nó. Após as discussões, definiram-se que três estados era o suficiente para o usuário do

modelo monitorar a situação de cada fator. Além disto, os estados são ordinais. Dessa forma, definiu-se três tipos de estados: *Bad*, *Moderate*, *Good*. Definiu-se como *High*, o estado que representa a situação desejada para o fator associado. Os estados identificados são apresentados no Apêndice A.

4.1.2 Construção das Tabelas de Probabilidade dos Nós (TPN)

Uma TPN pode ser representada por $Pr = \{A|B\}$, em que A é uma variável dependente e B é o conjunto de nós pais. Desta forma, o conjunto $P = \{Pr_i, \ldots, Pr_{|P|}\}$ representa todas as TPN da *rede Bayesiana*. Então, o problema resume-se em identificar os elementos do conjunto P. Para identificá-los, para cada Pr_i, em que $1 \leq i \leq |P|$, necessita-se quantificar os relacionamentos entre A e B_j, com $1 \leq j \leq |B|$ e $B_j \in B$.

Como dito anteriormente, o AgenaRisk foi utilizado para construir o modelo. Uma das funcionalidades do AgenaRisk é gerar TPN a partir de dados qualitativos. Esta funcionalidade implementa as ideias apresentadas no trabalho de Fenton et al. [14]. Neste trabalho, uma forma de gerar TPN para nós com valores medidos em uma escala subjetiva (i.e escala Likert) a partir do conhecimento de especialistas é apresentada. Para tal, nós ranqueados são utilizados. Os estados dos nós ranqueados são expressos em uma escala ordinal que pode ser mapeada para uma escala numérica limitada que é contínua e ordenada monotonicamente.

Para modelar as causas entre nós ordinais, uma distribuição Normal duplamente truncada, em que todos os nós são truncados na intervalo $[0, 1]$, foi utilizada. Esta distribuição tem dois parâmetros: μ e σ^2, em que μ é a média e σ^2 é a variância. No AgenaRisk, para calcular μ, necessita-se criar uma expressão ponderada que reflita a influência dos nós pais no nó sendo considerado. Por exemplo, dado que o nó C tem dois nós pais: A e B, para calcular μ, teria-se a expressão $\mu = \mu_1 A + \mu_2 B$, em que μ_1 representa a influência relativa de A em C, e μ_2 representa a influência relativa de B em C. σ^2 define a forma da distribuição: $\sigma^2 = 5.0E^{-4}$ - em que $5.0E^{-4}$ é o menor valor possível para σ^2 - para uma distribuição fortemente enviesada e $\sigma^2 = 8$ para uma distribuição uniforme - em que 8 é o maior valor possível para σ^2, ou seja, σ^2 representa a confiança no resultado.

Desta forma, para construir as TPN usando a funcionalidade descrita pelo AgenaRisk, necessita-se gerar expressões ponderadas. Neste trabalho, primeiramente, ordenou-se a influência dos nós pais de todos os nós da *rede Bayesiana*. Dada a ordenação, criou-se um

algoritmo para gerar as expressões ponderadas.

Ordenar a influência dos nós pais de cada nó da rede

Para coletar a opinião de especialistas com o intuito de ordenar a influência dos nós pais de cada nó da rede, criou-se um questionário. Para cada TPN, criou-se uma questão. O conjunto de questões criadas compõe o questionário. A partir do questionário, coletou-se dados de quarenta profissionais da área. Com os dados coletados, para cada nó da *rede Bayesiana*, aplicou-se estatística inferencial para ordenar os nós pais dado suas magnitudes relativas. As perguntas do questionário encontram-se no Apêndice B e os dados coletados no Apêndice C.

Primeiramente, divulgou-se a primeira versão do questionário para ser pré-testada usando o SurveyTool [1]. No pré-teste, cada questão do questionário foi respondida e comentada pelo mesmo grupo de especialistas que colaborou para definir a DAG. Como resultado do pré-teste, definiu-se um formato para formular todas as questões e decidiu-se incluir, para cada questão, um glossário explicando cada termo técnico usado na mesma.

Uma das decisões mais importantes tomada durante o pré-teste foi relativa à escala utilizada para coletar a opinião dos respondentes. Inicialmente, uma escala intervalar seria utilizada, pois os resultados poderiam ser diretamente utilizados para gerar as expressões matemáticas necessárias para popular as TPN. Por outro lado, durante o pré-teste, por unanimidade, se concluiu que seria muito confuso para os participantes expressarem sua opinião com escalas intervalares e isso aumentaria a complexidade do questionário. Além disto, dado que a survey seria confusa e complexa, os resultados não teriam a acurácia necessaria para criar as expressões matemáticas. Desta forma, por unanimidade, se decidiu usar uma escala Likert de cinco pontos, indo de *Very Low Influence* até *Very High Influence*. Consequentemente, ao invés de gerar resultados que poderiam ser utilizados diretamente para gerar as expressões ponderadas, o questionário, para cada questão e após a análise estatística inferencial, definiu-se uma lista ordenada dos relacionamentos dado a magnitude relativa. O questionário foi publicado em grupos relacionados ao *Scrum* do LinkedIn, na Fan Page do Facebook da *Scrum.org* - com autorização -, e pela *ScrumAlliance* a partir da sua conta no Twitter.

[1] SurveyTool. http://www.surveytool.com

Nesse estudo empírico, os relacionamento entre os fatores são as variáveis dependentes, e os participantes que responderam todas as questões do questionário são as variáveis independentes. Então, há setenta e uma variáveis dependentes e quarenta variáveis independentes. O questionário é composto por vinte e uma questões e para cada questão formulou-se um teste de hipótese. Como o objetivo é identificar se há diferença entre as magnitudes relativas dos relacionamentos envolvidos em cada questão, as hipóteses nulas afirmam que não há diferenças.

Foi necessário utilizar dois tipos de testes de hipóteses: teste de Friedman, para as questões compostas por mais de dois relacionamentos; e teste de Wilcoxon para as demais questões. Para os casos que utilizou-se o teste de Friedman, quando a hipótese nula foi refutada, foi necessário utilizar o teste post-hoc de Wilcoxon-Nemenyi-McDonald-Thompson [17] para ordenar os relacionamentos dado as magnitudes relativas. Calculou-se a margem de erro e nível de confiança usando $SS = \frac{Z^2 p(1-p)}{C^2}$ [4], em que SS é o tamanho da amostra, Z é o valor Z, p é a porcentagem de uma escolha e C é o nível de confiança. Dado que o tamanho da amostra é quarenta, a margem de erro calculada é 13% e o nível de confiança é 90%. Os resultados do estudo empírico são apresentados no Apêndice D.

Gerar expressões ponderadas

Com os nós pais ordenados para cada nó da *rede Bayesiana*, criou-se um algoritmo para transformar as listas ordenadas em expressões ponderadas. O algoritmo é apresentado no Algoritmo 1. Dado que utilizou-se uma escala Likert de cinco pontos, essa foi utilizada para comparar a magnitude relativa dos relacionamentos. Por exemplo, um relacionamento com magnitude relativa *Very High Influence* é maior que um relacionamento com magnitude relativa *High Influence*. Por exemplo, para o nó *Code inspection quality*, a expressão calculada foi $3A + 3B + C$, em que A é o relacionamento *Peer code review - Code inspection quality*, B é *Pair programming - Code inspection quality* e C é *Static code analysis - Code inspection quality*.

Neste trabalho, utilizou-se uma escala ordinal e não é possível definir as diferenças entre os possíveis valores de uma escala Likert. Por exemplo, não se pode afirmar que a diferença entre *Very High Influence* e *High Influence* é igual a diferença entre *Moderate Influence* e *Low Influence*. Desta forma, o algoritmo ignora as diferenças na escala Likert e apenas

Data: n = the number of relationships

Result: Weighted expression to create a NPT

for $i \leftarrow n$ **to** 0 **do**

  Get the relationship with greatest relative magnitude;

  Add the relationship to the expression with the weighted value of i;

  if *there are any more relationships with the same relative magnitude* **then**

    m = the number of additional relationships with the same relative magnitude;

    current section becomes this one;

    for $j \leftarrow m$ **to** 0 **do**

      Add the relationship to the expression with the weighted value of i

    end

    Decrement i by the number of additional relationships you added to the expression;

  end

end

Algorithm 1: Algoritmo que converte os resultados de uma questão do questionário em uma expressão matemática para popular a TPN

considera se há diferença entre os relacionamentos ou não. Dado que o objetivo do modelo é apenas servir como alerta para prováveis problemas e auxiliar nas ações de melhorias para o projeto, as TPN apenas precisam ser acuradas o suficiente para atingir este objetivo.

No AgenaRisk há quatro tipos de expressões: ponderada média, ponderada máxima, ponderada mínima e mix mínimo-máximo. Para todos os nós, foi utilizada a expressão ponderada média, pois assume-se para esse modelo que não há casos em que um nó pai é essencial para enviesar o tendência central do nó ranqueado para algum dos valores extremos possíveis. Além disto, assume-se que a confiança nos resultados é alta e constante. Desta forma, para todos os nós ranqueados a variância (σ^2) é $5.0E^{-4}$.

Capítulo 5

Validação do Modelo Probabilístico

Para validar modelos de processos de desenvolvimento de software, é possível aplicá-los a cenários e comparar os resultados calculados com os esperados [11]. Desta forma, para validar a modelagem, os resultados esperados e calculados de dez cenários foram comparados. Cada cenário descreve uma situação que poderia ocorrer em um projeto *Scrum* de desenvolvimento de software. Os cenários foram criados baseados na experiência do pesquisador. Para a definição dos resultados esperados e calculados, desconsiderou-se riscos que não sejam diretamente relacionados ao processo de desenvolvimento, por exemplo: falta de recursos, mudanças dos objetivos organizacionais e catástrofes naturais. Além disso, considera-se que, em todos os cenários, o projeto é adequado para a adoção do *Scrum*.

5.1 Cenário 1

Uma empresa, experiente em usar *Scrum* em projetos de desenvolvimento de software, contrata um grupo de *desenvolvedores* que estão habituados com o método tradicional de desenvolvimento de software para formar a equipe de desenvolvimento de um projeto. Desta forma, todos os envolvidos no projeto, exceto os *desenvolvedores*, têm experiência e habilidades necessárias para trabalhar em um projeto *Scrum*. Os *desenvolvedores* não estão habituados em desenvolver o produto de forma iterativa, se auto-gerenciar e trabalhar de forma colaborativa em todos as fases de desenvolvimento do produto. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.1.

Tabela 5.1: Entradas no modelo para o Cenário 1

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	100%	0%	0%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	0%	0%	100%
All participants present	0%	100%	0%
Answer the three basic questions	0%	100%	0%
Automated tests	0%	100%	0%
Available and qualified	0%	0%	100%
Broad and engaging goal	0%	0%	100%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	0%	0%	100%
Code integration frequency	100%	0%	0%
Code refactoring	0%	0%	100%
Collective ownership	100%	0%	0%
Communicator and negotiator	0%	0%	100%
Considers business value	0%	0%	100%
Considers non-functional requirements	0%	0%	100%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%
Continuous improvement	100%	0%	0%
Critical attributes to satisfy customer needs described	0%	0%	100%
Documentation	100%	0%	0%
Empowered and committed	0%	0%	100%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	0%	0%	100%
Leader and team player	0%	0%	100%
Members expertise	0%	100%	0%
Members motivation	0%	100%	0%
Members personality	100%	0%	0%
Monitor sprint progress	100%	0%	0%
Negotiable	0%	0%	100%
Pair programming	100%	0%	0%

Peer code review	100%	0%	0%
Product backlog items estimated	0%	0%	100%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	0%	100%
Short and concise	0%	0%	100%
Size estimation quality	0%	100%	0%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	0%	100%
Stakeholders feedback	0%	0%	100%
Static code analysis	0%	0%	100%
Task breakdown quality	100%	0%	0%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	0%	100%	0%
Test coverage analysis	0%	100%	0%
Testable	0%	0%	100%
Visionary and doer	0%	0%	100%

Para este projeto, é esperado que o modelo sinalize que há problemas nas práticas da equipe de desenvolvimento que diminuem as chances de sucesso do projeto. Por outro lado, dado que as regras, princípios e melhores práticas são seguidas pela empresa e o *dono do produto*, as chances de sucesso do projeto não serão baixas pois é provável que as deficiências da equipe de desenvolvimento sejam detectadas rapidamente. Desta forma, os problemas causados pela falta de maturidade da equipe de desenvolvimento serão detectados nas *Reuniões de revisão*, e os outros envolvidos no projeto terão informações para poder ajudar os *desenvolvedores*. As saídas do modelo são apresentadas na Tabela 5.2.

Tabela 5.2: Saídas do modelo para o Cenário 1

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	81,1%	18,9%	0%
Code quality	0%	100%	0%
Daily Scrum quality	82,2%	17,8%	0%
Development team competence	75,2%	24,8%	0%

Development team teamwork skills	0%	100%	0%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	0%	100%
Product backlog quality	0%	0%	100%
Product backlog management	0%	0%	100%
Product increment quality	4,50%	71,5%	24,0%
Product owner overall quality	0%	2,00%	98,0%
Product owner personal characteristics	0%	0%	100%
Product vision quality	0%	0%	100%
Project success	0%	22,0%	78,0%
Release plan	0%	0%	100%
Software engineering techniques quality	66,8%	33,2%	0%
Sprint backlog quality	0%	47,9%	52,1%
Sprint Planning quality	0%	9,90%	90,1%
Sprint progress	14,3%	84,5%	1,20%
Sprint Review meeting achieving its goals	0%	0%	100%
Work validation quality	0%	2,60%	97,4%

Observando os dados da Tabela 5.2, pode-se concluir que o modelo atende às expectativas. Além de detectar que o projeto tem mais chances de sucesso do que de falhar - 78% de chance de ter sucesso -, a qualidade do incremento produzido durante as *sprint* é a principal responsável por reduzir as chances de sucesso do projeto. Além disto, o modelo aponta as práticas de engenharia de software, auto-gerenciamento, colaboração e planejamento iterativo como principais áreas a serem melhoradas.

5.2 Cenário 2

Uma empresa, ao perceber que muitas outras empresas estão gerenciando seus projetos de desenvolvimento de software seguindo a metodologia *Ágil*, resolve segui-la. Dado que *Scrum* é o arcabouço *Ágil* mais popular, a empresa resolve usar *Scrum* para um projeto teste. *Desenvolvedores* que estão habituados com o método tradicional de desenvolvimento de software são nomeados para integrar a equipe de desenvolvimento. Para o cargo de *dono do produto*, nomeou-se um gerente de projetos sênior habituado ao microgerenciamento e comando-e-controle. Além disto, as outras partes envolvidas no projeto não acham neces-

sário ter tantas reuniões com a equipe. Desta forma, nenhum dos envolvidos no projeto tem experiência e habilidades necessárias para trabalhar em um projeto *Scrum*, ou seja, os *desenvolvedores* não estão habituados em desenvolver o produto de forma iterativa, se auto-gerenciar e trabalhar de forma colaborativa em todos as fases de desenvolvimento do produto. Além disto, o *dono do produto* não tem a personalidade e habilidades para liderar uma equipe auto-gerenciada e não há o apoio necessário para validar o produto. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.3.

Tabela 5.3: Entradas no modelo para o Cenário 2

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	100%	0%	0%
3 to 5 top features described	100%	0%	0%
Acceptance criteria check	100%	0%	0%
All participants present	0%	100%	0%
Answer the three basic questions	0%	100%	0%
Automated tests	0%	100%	0%
Available and qualified	0%	100%	0%
Broad and engaging goal	0%	100%	0%
Broad and realistic goal	100%	0%	0%
Clear acceptance criteria	100%	0%	0%
Clear and stable	0%	100%	0%
Code integration frequency	100%	0%	0%
Code refactoring	0%	100%	0%
Collective ownership	100%	0%	0%
Communicator and negotiator	100%	0%	0%
Considers business value	0%	100%	0%
Considers non-functional requirements	0%	100%	0%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	100%	0%
Continuous improvement	100%	0%	0%
Critical attributes to satisfy customer needs described	0%	100%	0%
Documentation	100%	0%	0%
Empowered and committed	0%	0%	100%
Estimated size is small	100%	0%	0%
Independent	0%	100%	0%

Launch date defined	0%	0%	100%
Leader and team player	100%	0%	0%
Members expertise	0%	100%	0%
Members motivation	100%	0%	0%
Members personality	100%	0%	0%
Monitor sprint progress	100%	0%	0%
Negotiable	100%	0%	0%
Pair programming	100%	0%	0%
Peer code review	100%	0%	0%
Product backlog items estimated	0%	100%	0%
Product backlog is emergent	0%	100%	0%
Progress tracking	0%	100%	0%
Short and concise	0%	100%	0%
Size estimation quality	100%	0%	0%
Sprint items and priority not stability	100%	0%	0%
Sprint goal check	100%	0%	0%
Sprint length	0%	100%	0%
Stakeholder feedback outside of the Sprint Review meeting	0%	100%	0%
Stakeholders feedback	100%	0%	0%
Static code analysis	0%	0%	100%
Task breakdown quality	100%	0%	0%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	100%	0%	0%
Test coverage analysis	0%	100%	0%
Testable	0%	100%	0%
Visionary and doer	0%	100%	0%

Para esse projeto, é esperado que o modelo sinalize que há muitas chances do projeto não atender a todas as expectativas. Além da falta de maturidade da equipe de desenvolvimento, os outros envolvidos no projeto não estão preparados para trabalhar em um projeto *Ágil* e haverá problemas em todas as áreas de desenvolvimento do produto. As saídas do modelo são apresentadas na Tabela 5.4.

Tabela 5.4: Saídas do modelo para o Cenário 2

	Estados

Fator	Bad	Moderate	Good
Code inspection quality	81,1%	18,9%	0%
Code quality	0%	100%	0%
Daily Scrum quality	29,1%	70,9%	0%
Development team competence	75,4%	24,6%	0%
Development team teamwork skills	1,7%	98,3%	0%
Product backlog is properly ordered	0%	0%	0%
Product backlog items are properly detailed	0%	0%	0%
Product backlog quality	5,60%	91,2%	3,20%
Product backlog management	4,40%	94,8%	0,800%
Product increment quality	70,7%	29,3%	0%
Product owner overall quality	4,40%	93,5%	2,10%
Product owner personal characteristics	4,00%	96,0%	0%
Product vision quality	0%	100%	0%
Project success	39,9%	60,1%	%
Release plan	0%	100%	0%
Software engineering techniques quality	66,2%	33,8%	0%
Sprint backlog quality	51,5%	48,5%	0%
Sprint Planning quality	70,8%	29,2%	0%
Sprint progress	72,6%	27,4%	0%
Sprint Review meeting achieving its goals	100%	0%	0%
Work validation quality	49,7%	50,3%	0%

Observando os dados da Tabela 5.4, pode-se concluir que o modelo atende às expectativas. Além de detectar que o projeto tende a falhar - 39,9% de chance de falhar totalmente e 60,1% de atender apenas algumas expectativas -, a falta de maturidade dos envolvidos com a gestão *Ágil* é a principal responsável por reduzir as chances de sucesso do projeto. Além disto, o modelo aponta que a única razão para o projeto ter chances de alcançar algumas expectativas do projeto é o fato do *dono do produto* ter conhecimentos técnicos para gerenciar escopo e tempo, e os *clientes*, mesmo que não seja de forma ideal, avaliam o produto sendo construído.

5.3 Cenário 3

Uma empresa é contratada para desenvolver um sistema. A contratada fornece os *desenvolvedores* e o *ScrumMaster*; e, a contratante fornece o *dono do produto*. As empresas são de países diferentes e a contratada deve enviar dois integrantes para trabalhar na sede da empresa contratante. Apenas o *dono do produto* representará a contratante nas reuniões do *Scrum*, e participará apenas duas vezes por semana das *reuniões diárias*. Além disto, há uma diferença de fuso horário de seis horas entre as sedes das duas empresas, que prejudica a comunicação entre os *desenvolvedores*, e entre os *desenvolvedores* e o *dono do produto*. A empresa contratada tem experiência e competência para trabalhar em projetos *Scrum*. Por outro lado, a empresa contratante migrou para o gerenciamento *Ágil* de projetos recentemente e, historicamente, não consegue definir a visão do produto pois há muitos *clientes* e, consequentemente, muitos conflitos de interesses. Historicamente, o escopo da *sprint* costuma ser alterado frequentemente pois o *dono do produto* procura atender de imediato as solicitações dos *clientes*, e, geralmente, as funcionalidades não são escritas com todos os detalhes necessários e as definições ficam a cargo dos *desenvolvedores*. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.5.

Tabela 5.5: Entradas no modelo para o Cenário 3

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	0%	100%
3 to 5 top features described	0%	100%	0%
Acceptance criteria check	0%	100%	0%
All participants present	0%	100%	0%
Answer the three basic questions	0%	0%	100%
Automated tests	0%	0%	100%
Available and qualified	0%	100%	0%
Broad and engaging goal	0%	100%	0%
Broad and realistic goal	100%	0%	0%
Clear acceptance criteria	100%	0%	0%
Clear and stable	100%	0%	0%
Code integration frequency	0%	100%	0%
Code refactoring	0%	0%	100%

Collective ownership	0%	100%	0%
Communicator and negotiator	100%	0%	0%
Considers business value	0%	100%	0%
Considers non-functional requirements	100%	0%	0%
Considers risk	0%	100%	0%
Considers technical dependencies	0%	0%	100%
Continuous improvement	0%	0%	100%
Critical attributes to satisfy customer needs described	100%	0%	0%
Documentation	0%	0%	100%
Empowered and committed	0%	100%	0%
Estimated size is small	0%	100%	0%
Independent	0%	100%	0%
Launch date defined	100%	0%	0%
Leader and team player	0%	100%	0%
Members expertise	0%	0%	100%
Members motivation	0%	0%	100%
Members personality	0%	0%	100%
Monitor sprint progress	0%	0%	100%
Negotiable	100%	0%	0%
Pair programming	0%	0%	100%
Peer code review	0%	0%	100%
Product backlog items estimated	0%	100%	0%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	100%	0%
Short and concise	100%	0%	0%
Size estimation quality	0%	100%	0%
Sprint items and priority not stability	100%	0%	0%
Sprint goal check	100%	0%	0%
Sprint length	100%	0%	0%
Stakeholder feedback outside of the Sprint Review meeting	0%	0%	100%
Stakeholders feedback	100%	0%	0%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	0%	100%
Team physical distribution	100%	0%	0%
Team size	0%	0%	100%
Team velocity defined	100%	0%	0%
Test coverage analysis	0%	0%	100%

Testable	0%	100%	0%
Visionary and doer	0%	100%	0%

Para esse projeto, é esperado que o modelo sinalize que há muitas chances do projeto não atender a todas as expectativas. As principais causas são o envolvimento dos *clientes* em momentos inoportunos, falta de visão e definição de objetivos do projeto. As saídas do modelo são apresentadas na Tabela 5.6.

Tabela 5.6: Saídas do modelo para o Cenário 3

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	0%	100%
Code quality	0%	0%	100%
Daily Scrum quality	0%	17,7%	82,3%
Development team competence	0%	14,6%	85,4%
Development team teamwork skills	0%	50,0%	50,0%
Product backlog is properly ordered	0%	82,6%	17,4%
Product backlog items are properly detailed	53,2%	46,8%	0%
Product backlog quality	20,4%	75,3%	4,3%
Product backlog management	0%	91,2%	8,8%
Product increment quality	37,6%	60,9%	1,60%
Product owner overall quality	10,1%	87,6%	2,20%
Product owner personal characteristics	4,00%	96,0%	0%
Product vision quality	90,4%	9,60%	0%
Project success	11,4%	85,8%	2,80%
Release plan	72,0%	28,0%	0%
Software engineering techniques quality	0%	24,6%	76,4%
Sprint backlog quality	2,80%	89,1%	8,10%
Sprint Planning quality	87,3%	12,7%	0%
Sprint progress	0%	83,0%	17,0%
Sprint Review meeting achieving its goals	77,2%	22,8%	0%
Work validation quality	1,70%	85,2%	13,1%

Observando os dados da Tabela 5.6, pode-se concluir que o modelo atende às expectativas. Além de detectar que o projeto tende a falhar - 11,3% de chance de falhar totalmente e 85,8% de atender apenas algumas expectativas -, o envolvimento dos *clientes* em momentos

inoportunos, falta de visão e definição de objetivos do projeto são as principais responsáveis por reduzir as chances de sucesso do projeto. Além disto, o modelo aponta que as razões para o projeto ter chances de alcançar algumas expectativas do projeto é por ter uma equipe de *desenvolvedores* competente e ter, pelo menos, algum tipo de envolvimento dos *clientes*.

5.4 Cenário 4

Uma empresa com várias sedes no Brasil e madura em gestão *Ágil* indica uma equipe para desenvolver um novo sistema usando *Scrum*. Os membros dessa equipe trabalham em sedes diferentes da empresa, porém não há diferença de fuso horário. A empresa utiliza as melhores ferramentas para comunicação a longa distância e gerenciamento de projetos. O sistema resultante do projeto está bem definido e tanto os *desenvolvedores* como os *clientes* e *dono do produto* tem experiência e competência para trabalhar com o *Scrum*. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.7.

Tabela 5.7: Entradas no modelo para o Cenário 4

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	100%	0%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	0%	0%	100%
All participants present	0%	0%	100%
Answer the three basic questions	0%	0%	100%
Automated tests	0%	0%	100%
Available and qualified	0%	0%	100%
Broad and engaging goal	0%	0%	100%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	0%	0%	100%
Code integration frequency	0%	0%	100%
Code refactoring	0%	0%	100%
Collective ownership	0%	0%	100%
Communicator and negotiator	0%	0%	100%
Considers business value	0%	0%	100%
Considers non-functional requirements	0%	0%	100%

Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%
Continuous improvement	0%	0%	100%
Critical attributes to satisfy customer needs described	0%	0%	100%
Documentation	0%	0%	100%
Empowered and committed	0%	0%	100%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	0%	0%	100%
Leader and team player	0%	0%	100%
Members expertise	0%	0%	100%
Members motivation	0%	0%	100%
Members personality	0%	0%	100%
Monitor sprint progress	0%	0%	100%
Negotiable	0%	0%	100%
Pair programming	0%	100%	0%
Peer code review	0%	0%	100%
Product backlog items estimated	0%	0%	100%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	0%	100%
Short and concise	0%	0%	100%
Size estimation quality	0%	0%	100%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	0%	100%
Stakeholders feedback	0%	0%	100%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	0%	100%
Team physical distribution	100%	0%	0%
Team size	0%	0%	100%
Team velocity defined	0%	0%	100%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	0%	0%	100%

Para esse projeto, é esperado que o modelo sinalize que há muitas chances do projeto

obter sucesso. Apesar da equipe ser distribuída, as tecnologias de comunicação a longa distância e as ferramentas de gerenciamento de projetos que existem atualmente devem ajudar a equipe a diminuir o impacto da distância. As saídas do modelo são apresentadas na Tabela 5.8.

Tabela 5.8: Saídas do modelo para o Cenário 3

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	38,3%	61,7%
Code quality	0%	0%	100%
Daily Scrum quality	0%	1,50%	98,5%
Development team competence	0%	5,50%	94,5%
Development team teamwork skills	0%	15,0%	85,0%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	0%	100%
Product backlog quality	0%	2,10%	97,9%
Product backlog management	0%	0%	100%
Product increment quality	0%	3,50%	96,5%
Product owner overall quality	0%	2,10%	97,9%
Product owner personal characteristics	0%	0%	100%
Product vision quality	0%	0%	100%
Project success	0%	2,5%	97,5%
Release plan	0%	0%	100%
Software engineering techniques quality	0%	9,4%	90,6%
Sprint backlog quality	0%	0%	100%
Sprint Planning quality	0%	0%	100%
Sprint progress	0%	2,30%	97,7%
Sprint Review meeting achieving its goals	0%	0%	100%
Work validation quality	0%	2,60%	97,4%

Observando os dados da Tabela 5.8, pode-se concluir que o modelo atende às expectativas, pois indica que há 97,5% do projeto obter sucesso. Desde que a equipe utilize ferramentas apropriadas de comunicação e gerenciamento do projeto, o fato de esta equipe ser distribuída não deve afetar de forma crítica as chances de sucesso do projeto.

5.5 Cenário 5

Uma empresa nova com recursos humanos competentes indica uma equipe para desenvolver um novo sistema usando *Scrum*. Os membros dessa equipe trabalham em sedes diferentes da empresa, porém não há diferença de fuso horário. A empresa utiliza ferramentas precárias para comunicação a longa distância e gerenciamento de projetos por não querer aumentar os custos do projeto. O sistema resultante do projeto está bem definido e tanto os *desenvolvedores* como os *clientes* e *dono do produto* tem experiência e competência para trabalhar com o *Scrum*. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.9.

Tabela 5.9: Entradas no modelo para o Cenário 5

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	100%	0%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	0%	100%	0%
All participants present	100%	0%	0%
Answer the three basic questions	100%	0%	0%
Automated tests	0%	0%	100%
Available and qualified	0%	0%	100%
Broad and engaging goal	0%	0%	100%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	0%	0%	100%
Code integration frequency	0%	100%	0%
Code refactoring	0%	0%	100%
Collective ownership	0%	100%	0%
Communicator and negotiator	0%	0%	100%
Considers business value	0%	0%	100%
Considers non-functional requirements	0%	0%	100%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%
Continuous improvement	0%	0%	100%
Critical attributes to satisfy customer needs described	0%	0%	100%
Documentation	0%	0%	100%

Empowered and committed	0%	0%	100%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	0%	0%	100%
Leader and team player	0%	0%	100%
Members expertise	0%	0%	100%
Members motivation	0%	0%	100%
Members personality	0%	0%	100%
Monitor sprint progress	0%	0%	100%
Negotiable	0%	0%	100%
Pair programming	0%	100%	0%
Peer code review	0%	100%	0%
Product backlog items estimated	0%	100%	0%
Product backlog is emergent	0%	0%	100%
Progress tracking	100%	0%	0%
Short and concise	0%	0%	100%
Size estimation quality	0%	100%	0%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	100%	0%
Stakeholders feedback	0%	100%	0%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	100%	0%
Team physical distribution	100%	0%	0%
Team size	0%	0%	100%
Team velocity defined	0%	0%	100%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	0%	0%	100%

Para esse projeto, é esperado que o modelo sinalize que há as chances de obter sucesso são moderadas. A falta de ferramentas adequadas de comunicação a longa distância e as ferramentas de gerenciamento de projetos vão causar problemas na colaboração da equipe e análise do progresso do projeto. As saídas do modelo são apresentadas na Tabela 5.10.

Tabela 5.10: Saídas do modelo para o Cenário 5

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	93,5%	6,50%
Code quality	0%	0%	100%
Daily Scrum quality	70,8%	29,2%	0%
Development team competence	0%	37,9%	62,1%
Development team teamwork skills	0%	50,0%	50,0%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	0%	100%
Product backlog quality	0%	2,10%	97,9%
Product backlog management	0%	9,90%	90,1%
Product increment quality	0%	40,2%	59,8%
Product owner overall quality	0%	24,8%	75,2%
Product owner personal characteristics	0%	0%	100%
Product vision quality	0%	0%	100%
Project success	0%	50,8%	49,2%
Release plan	0%	0%	100%
Software engineering techniques quality	0%	9,4%	90,6%
Sprint backlog quality	0%	62,1%	37,1%
Sprint Planning quality	0%	6,10%	93,9%
Sprint progress	0%	73,1%	26,1%
Sprint Review meeting achieving its goals	0%	22,7%	77,3%
Work validation quality	1,80%	85,1%	13,1%

Observando os dados da Tabela 5.10, pode-se concluir que o modelo atende às expectativas, pois indica que há 49,2% do projeto obter sucesso. Mesmo a equipe sendo composta por recursos humanos competentes e experientes, o fato de o projeto ser composto por equipes distribuídas e não contar com as ferramentas adequadas para garantir colaboração e gerenciamento do projeto impacta negativamente, principalmente, a qualidade do incremento e a validação do mesmo.

5.6 Cenário 6

Os executivos e gerentes de uma empresa de desenvolvimento de software decidem migrar do método tradicional de gerenciamento de projetos para o *Ágil*. O *waterfall* era o

método utilizado para desenvolver software, mas dado que a empresa entrará em um novo nicho de mercado que utiliza novas tecnologias, decidiu utilizar a metodologia *Ágil*, mais especificamente, o *Scrum*. Desta forma, uma equipe foi nomeada para trabalhar no primeiro projeto da empresa usando o *Scrum*. Porém, treinamentos não foram fornecidos para os *desenvolvedores* e, consequentemente, eles continuaram trabalhando com a mesma cultura que estavam habituados. Ou seja, as equipes são formadas por especialistas que trabalham apenas nas suas áreas específicas e o ciclo de desenvolvimento é similar ao *waterfall*, mas em um espaço mais curto de tempo. Por outro lado, os *clientes* e o *dono do produto* envolvidos no projeto são competentes e comprometidos com a gestão *Ágil*. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.11.

Tabela 5.11: Entradas no modelo para o Cenário 6

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	100%	0%	0%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	0%	0%	100%
All participants present	0%	100%	0%
Answer the three basic questions	100%	0%	0%
Automated tests	0%	100%	0%
Available and qualified	0%	0%	100%
Broad and engaging goal	0%	0%	100%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	0%	0%	100%
Code integration frequency	100%	0%	0%
Code refactoring	0%	100%	0%
Collective ownership	100%	0%	0%
Communicator and negotiator	0%	0%	100%
Considers business value	0%	0%	100%
Considers non-functional requirements	0%	0%	100%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%
Continuous improvement	100%	0%	0%
Critical attributes to satisfy customer needs described	0%	0%	100%

Documentation	100%	0%	0%
Empowered and committed	0%	0%	100%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	0%	0%	100%
Leader and team player	0%	0%	100%
Members expertise	100%	0%	0%
Members motivation	0%	100%	0%
Members personality	100%	0%	0%
Monitor sprint progress	0%	100%	0%
Negotiable	0%	0%	100%
Pair programming	100%	0%	0%
Peer code review	100%	0%	0%
Product backlog items estimated	0%	0%	100%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	0%	100%
Short and concise	0%	0%	100%
Size estimation quality	100%	0%	0%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	0%	100%
Stakeholders feedback	0%	0%	100%
Static code analysis	0%	0%	100%
Task breakdown quality	100%	0%	0%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	100%	0%	0%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	0%	0%	100%

Para esse projeto, é esperado que o modelo sinalize que há mais chances do projeto obter sucesso do que falhar. Dado que os *clientes* e o *dono do produto* seguem as regras do *Scrum*, os problemas com a equipe de desenvolvimento serão detectados. Apesar disto, o modelo deve indicar que os *desenvolvedores* necessitam de treinamento em práticas *Ágeis* e *Scrum* para aumentar as chances de sucesso do projeto. As saídas do modelo são apresentadas na

Tabela 5.12.

Tabela 5.12: Saídas do modelo para o Cenário 6

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	81,1%	18,9%	0%
Code quality	0%	77,3%	22,7%
Daily Scrum quality	1,70%	98,3%	0%
Development team competence	69,2%	30,8%	0%
Development team teamwork skills	0%	100%	0%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	0%	100%
Product backlog quality	0%	2,90%	97,1%
Product backlog management	0%	0%	100%
Product increment quality	2,20%	87,7%	10,1%
Product owner overall quality	0%	2,10%	97,9%
Product owner personal characteristics	0%	0%	100%
Product vision quality	0%	0%	100%
Project success	0%	24,1%	75,9%
Release plan	0%	0%	100%
Software engineering techniques quality	55,9%	44,1%	0%
Sprint backlog quality	5,25%	94,7%	0%
Sprint Planning quality	0%	90,6%	9,40%
Sprint progress	0%	92,0%	8,00%
Sprint Review meeting achieving its goals	0%	0%	100%
Work validation quality	0%	2,60%	97,4%

Observando os dados da Tabela 5.12, pode-se concluir que o modelo atende às expectativas. Além de detectar que o projeto tem mais chances de sucesso do que de falhar - 75,9% de probabilidade de ter boas chances de sucesso -, a qualidade do incremento produzido durante as *sprint* é a principal responsável por reduzir as chances de sucesso do projeto. Além disto, o modelo aponta as práticas de engenharia de software, auto-gerenciamento, colaboração e planejamento iterativo como principais áreas a serem melhoradas.

5.7 Cenário 7

Uma empresa é contratada para desenvolver um sistema. A contratada fornece os *desenvolvedores* e *ScrumMaster*; e, a contratante fornece o *dono do produto* e um arquiteto de sistemas que deve compor a equipe de desenvolvimento. Pelo contrato firmado entre as duas empresas, o arquiteto da empresa contratante deve ter o poder final de decisões técnicas da equipe. Além disto, apenas o *dono do produto* representará a contratante nas reuniões do *Scrum*, participará apenas duas vezes por semana das *reuniões diárias* e apenas se reunirá bimestralmente com os *clientes* da contratante para discutir o projeto. Os *desenvolvedores* e o *dono do produto* são treinados e experientes em utilizar o *Scrum*. Por outro lado, apenas a empresa contratante tem o poder de tomar decisões sobre o projeto e o *dono do produto* tem que se adaptar a essas condições do contrato. O arquiteto é o melhor funcionário, em termos técnicos, da empresa contratante. Por outro lado, ele costuma centralizar as decisões técnicas e não tem facilidade de trabalhar em equipe. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.13.

Tabela 5.13: Entradas no modelo para o Cenário 7

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	0%	100%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	0%	0%	100%
All participants present	100%	0%	0%
Answer the three basic questions	0%	0%	100%
Automated tests	0%	0%	100%
Available and qualified	100%	0%	0%
Broad and engaging goal	0%	100%	0%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	0%	0%	100%
Code integration frequency	0%	0%	100%
Code refactoring	0%	0%	100%
Collective ownership	0%	100%	0%
Communicator and negotiator	0%	100%	0%
Considers business value	0%	0%	100%

Considers non-functional requirements	0%	0%	100%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%
Continuous improvement	0%	100%	0%
Critical attributes to satisfy customer needs described	0%	0%	100%
Documentation	0%	0%	100%
Empowered and committed	0%	100%	0%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	0%	0%	100%
Leader and team player	0%	100%	0%
Members expertise	0%	0%	100%
Members motivation	100%	0%	0%
Members personality	100%	0%	0%
Monitor sprint progress	0%	100%	0%
Negotiable	100%	0%	0%
Pair programming	0%	0%	100%
Peer code review	0%	0%	100%
Product backlog items estimated	0%	0%	100%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	0%	100%
Short and concise	0%	0%	100%
Size estimation quality	0%	0%	100%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	100%	0%	0%
Stakeholders feedback	100%	0%	0%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	0%	100%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	0%	0%	100%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	100%	0%	0%

Para esse projeto, é esperado que o modelo sinalize que há mais chances do projeto falhar do que obter sucesso. Além da falta de validação no final das *sprints* por parte dos *clientes* e a pouca disponibilidade do *dono do produto*, a presença do arquiteto na equipe de desenvolvimento vai reduzir a colaboração na equipe e prejudicar o seu auto-gerenciamento. As saídas do modelo são apresentadas na Tabela 5.14.

Tabela 5.14: Saídas do modelo para o Cenário 7

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	0%	100%
Code quality	0%	0%	100%
Daily Scrum quality	0%	98,4%	1,60%
Development team competence	0%	62,1%	37,9%
Development team teamwork skills	0%	100%	0%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	13,4%	86,6%
Product backlog quality	0%	6,80%	93,2%
Product backlog management	0%	1,60%	98,4%
Product increment quality	0%	32,1%	67,8%
Product owner overall quality	0%	56,8%	43,2%
Product owner personal characteristics	28,4%	71,6%	0%
Product vision quality	0%	50,0%	50,0%
Project success	1,90%	74,1%	24,0%
Release plan	0%	11,7%	88,3%
Software engineering techniques quality	0%	1,10%	98,9%
Sprint backlog quality	2,00%	59,2%	38,8%
Sprint Planning quality	0%	6,80%	93,2%
Sprint progress	0%	55,9%	44,1%
Sprint Review meeting achieving its goals	0%	77,2%	22,8%
Work validation quality	39,6%	60,4%	0%

Observando os dados da Tabela 5.14, pode-se concluir que o modelo atende às expectativas. Além de detectar que o projeto tem mais chances de falhar do que de obter sucesso - apenas 24,0% de chance de atender todas as expectativas -, a falta de disponibilidade do *dono do produto* e *clientes*, e o fato das decisões serem unilaterais serem as principais causas

disso. Além disto, o modelo aponta a validação do produto, o papel do *dono do produto* e colaboração entre os *desenvolvedores* como pontos a serem melhorados.

5.8 Cenário 8

Uma empresa contrata um consultor *Ágil* renomado para capacitar seus executivos, *desenvolvedores* e *ScrumMasters*. Depois da consultoria, uma equipe é alocada para trabalhar em um projeto cujo o objetivo não está bem definido. Além disto, a empresa decidiu não seguir algumas recomendações do consultor, e iniciou o projeto sem definir a data final e os atributos básicos da definição da visão do produto. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.15.

Tabela 5.15: Entradas no modelo para o Cenário 8

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	0%	100%
3 to 5 top features described	100%	0%	0%
Acceptance criteria check	0%	0%	100%
All participants present	0%	0%	100%
Answer the three basic questions	0%	0%	100%
Automated tests	0%	0%	100%
Available and qualified	0%	0%	100%
Broad and engaging goal	100%	0%	0%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	100%	0%	0%
Code integration frequency	0%	0%	100%
Code refactoring	0%	0%	100%
Collective ownership	0%	0%	100%
Communicator and negotiator	0%	0%	100%
Considers business value	0%	0%	100%
Considers non-functional requirements	0%	0%	100%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%
Continuous improvement	0%	0%	100%

Critical attributes to satisfy customer needs described	0%	100%	0%
Documentation	0%	0%	100%
Empowered and committed	0%	0%	100%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	100%	0%	0%
Leader and team player	0%	0%	100%
Members expertise	0%	0%	100%
Members motivation	0%	0%	100%
Members personality	0%	0%	100%
Monitor sprint progress	0%	0%	100%
Negotiable	0%	0%	100%
Pair programming	0%	0%	100%
Peer code review	0%	0%	100%
Product backlog items estimated	0%	0%	100%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	0%	100%
Short and concise	100%	0%	0%
Size estimation quality	0%	0%	100%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	0%	100%
Stakeholders feedback	0%	0%	100%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	0%	100%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	0%	0%	100%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	0%	0%	100%

Para esse projeto, é esperado que o modelo sinalize que probabilidade de sucesso é elevada. Apesar da falta de definição do produto final, seguir as regras, melhores práticas e princípios do *Scrum* devem levar o projeto a alcançar um produto que seja competitivo no mercado. Apesar disto, o modelo deve sinalizar que a definição da visão do produto e dos

objetivos do projeto são cruciais para aumentar a sua probabilidade de sucesso. As saídas do modelo são apresentadas na Tabela 5.16.

Tabela 5.16: Saídas do modelo para o Cenário 8

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	0%	100%
Code quality	0%	0%	100%
Daily Scrum quality	0%	0%	100%
Development team competence	0%	2,20%	97,8%
Development team teamwork skills	0%	5,10%	84,9%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	0%	100%
Product backlog quality	0%	68,9%	31,1%
Product backlog management	0%	0%	100%
Product increment quality	0%	0%	100%
Product owner overall quality	0%	26,8%	73,2%
Product owner personal characteristics	0%	0%	100%
Product vision quality	90,4%	9,60%	0%
Project success	0%	8,30%	91,7%
Release plan	97,2%	2,80%	0%
Software engineering techniques quality	0%	1,10%	98,9%
Sprint backlog quality	0%	0%	100%
Sprint Planning quality	0%	0%	100%
Sprint progress	0%	1,30%	98,7%
Sprint Review meeting achieving its goals	0%	0%	100%
Work validation quality	0%	2,60%	97,4%

Observando os dados da Tabela 5.16, pode-se concluir que o modelo atende às expectativas. Além de detectar que o projeto tem probabilidade elevada de sucesso - 91,7% -, o fato de todos os envolvidos seguirem as regras e melhores práticas do *Scrum* é a principal responsável por aumentar as chances de sucesso do projeto. Além disto, o modelo aponta que a definição do produto e a motivação da equipe são áreas que podem ser melhoradas. É importante notar que a baixa motivação da equipe pode alterar valores de outras entradas do modelo no futuro. Apesar da probabilidade de sucesso do projeto ter sido calculada em

91,7%, isso reflete o estado no qual o projeto foi avaliado. Futuramente, a falta de motivação da equipe, por exemplo, pode, indiretamente, influenciar negativamente a qualidade do código e da colaboração da equipe.

5.9 Cenário 9

Uma empresa contrata um consultor *Ágil* renomado para capacitar seus executivos, *desenvolvedores* e *ScrumMasters*. Depois da consultoria, os executivos e *desenvolvedores* trabalham colaborativamente para planejar um projeto seguindo as recomendações do consultor. Desta forma, as melhores práticas para gestão de projetos *Ágeis* foram colocadas em práticas, incluindo as definições do produto a ser construído durante o projeto. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.17.

Tabela 5.17: Entradas no modelo para o Cenário 9

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	0%	100%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	0%	0%	100%
All participants present	0%	0%	100%
Answer the three basic questions	0%	0%	100%
Automated tests	0%	0%	100%
Available and qualified	0%	0%	100%
Broad and engaging goal	0%	0%	100%
Broad and realistic goal	0%	0%	100%
Clear acceptance criteria	0%	0%	100%
Clear and stable	0%	0%	100%
Code integration frequency	0%	0%	100%
Code refactoring	0%	0%	100%
Collective ownership	0%	0%	100%
Communicator and negotiator	0%	0%	100%
Considers business value	0%	0%	100%
Considers non-functional requirements	0%	0%	100%
Considers risk	0%	0%	100%
Considers technical dependencies	0%	0%	100%

Continuous improvement	0%	0%	100%
Critical attributes to satisfy customer needs described	0%	0%	100%
Documentation	0%	0%	100%
Empowered and committed	0%	0%	100%
Estimated size is small	0%	0%	100%
Independent	0%	0%	100%
Launch date defined	0%	0%	100%
Leader and team player	0%	0%	100%
Members expertise	0%	0%	100%
Members motivation	0%	0%	100%
Members personality	0%	0%	100%
Monitor sprint progress	0%	0%	100%
Negotiable	0%	0%	100%
Pair programming	0%	0%	100%
Peer code review	0%	0%	100%
Product backlog items estimated	0%	0%	100%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	0%	100%
Short and concise	0%	0%	100%
Size estimation quality	0%	0%	100%
Sprint items and priority not stability	0%	0%	100%
Sprint goal check	0%	0%	100%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	0%	100%
Stakeholders feedback	0%	0%	100%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	0%	100%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	0%	0%	100%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	0%	0%	100%

Para esse projeto, é esperado que o modelo sinalize que há muitas chances do projeto obter sucesso. Além da equipe de desenvolvimento, o *dono do produto* e as outras par-

tes interessadas seguem as melhores práticas e regras do *Scrum*. As saídas do modelo são apresentadas na Tabela 5.18.

Tabela 5.18: Saídas do modelo para o Cenário 9

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	0%	100%
Code quality	0%	0%	100%
Daily Scrum quality	0%	0%	100%
Development team competence	0%	1,70%	98,3%
Development team teamwork skills	0%	2,90 %	97,1%
Product backlog is properly ordered	0%	0%	100%
Product backlog items are properly detailed	0%	0%	100%
Product backlog quality	0%	4,10%	95,9%
Product backlog management	0%	0%	100%
Product increment quality	0%	2,90%	97,1%
Product owner overall quality	0%	2,10%	97,9%
Product owner personal characteristics	0%	0%	100%
Product vision quality	0%	0%	100%
Project success	0%	2,40%	97,6%
Release plan	0%	0%	100%
Software engineering techniques quality	0%	1,10%	98,9%
Sprint backlog quality	0%	0%	100%
Sprint Planning quality	0%	0%	100%
Sprint progress	0%	1,10%	98,9%
Sprint Review meeting achieving its goals	0%	0%	100%
Work validation quality	0%	2,60%	97,4%

Observando os dados da Tabela 5.18, pode-se concluir que o modelo atende às expectativas. Dado que todas as partes interessadas do projeto seguem as melhores práticas e regras do *Scrum*, o mesmo está encaminhado para ter sucesso.

5.10 Cenário 10

Como é feito em muitas empresas, uma empresa promove um dos seus melhores *desenvolvedores*, sem nenhum treinamento específico, para gerente. Para um projeto cujo objetivo

é desenvolver um novo sistema, nomeou-se o recém-promovido gerente para ser o *dono do produto*. Apesar de ser um ótimo *desenvolvedor*, o recém-promovido gerente não tem experiência com gerenciamento de projetos e não tem as habilidades necessárias para guiar a equipe de desenvolvimento. Para o projeto, além do recém-promovido gerente, uma equipe experiente de *desenvolvedores* foi alocada, os *clientes* se comprometeram a seguir o *Scrum* e o produto foi definido usando as melhores práticas *Ágeis*. Os dados de entrada para o modelo desse projeto são apresentados na Tabela 5.19.

Tabela 5.19: Entradas no modelo para o Cenário 10

	Estados		
Fator	Bad	Moderate	Good
15 minutes limit	0%	0%	100%
3 to 5 top features described	0%	0%	100%
Acceptance criteria check	100%	0%	0%
All participants present	0%	0%	100%
Answer the three basic questions	0%	0%	100%
Automated tests	0%	0%	100%
Available and qualified	100%	0%	0%
Broad and engaging goal	0%	0%	100%
Broad and realistic goal	100%	0%	0%
Clear acceptance criteria	0%	100%	0%
Clear and stable	0%	0%	100%
Code integration frequency	0%	100%	0%
Code refactoring	0%	0%	100%
Collective ownership	0%	0%	100%
Communicator and negotiator	100%	0%	0%
Considers business value	0%	100%	0%
Considers non-functional requirements	0%	0%	0%
Considers risk	0%	100%	0%
Considers technical dependencies	0%	0%	100%
Continuous improvement	0%	0%	100%
Critical attributes to satisfy customer needs described	0%	0%	100%
Documentation	0%	0%	100%
Empowered and committed	0%	0%	100%
Estimated size is small	0%	100%	0%
Independent	100%	0%	0%

Launch date defined	0%	0%	100%
Leader and team player	100%	0%	0%
Members expertise	0%	0%	100%
Members motivation	0%	0%	100%
Members personality	0%	0%	100%
Monitor sprint progress	0%	100%	0%
Negotiable	100%	0%	0%
Pair programming	0%	0%	100%
Peer code review	0%	0%	100%
Product backlog items estimated	0%	100%	0%
Product backlog is emergent	0%	0%	100%
Progress tracking	0%	100%	0%
Short and concise	0%	0%	100%
Size estimation quality	0%	0%	100%
Sprint items and priority not stability	0%	100%	0%
Sprint goal check	100%	0%	0%
Sprint length	0%	0%	100%
Stakeholder feedback outside of the Sprint Review meeting	0%	100%	0%
Stakeholders feedback	0%	100%	0%
Static code analysis	0%	0%	100%
Task breakdown quality	0%	0%	100%
Team physical distribution	0%	0%	100%
Team size	0%	0%	100%
Team velocity defined	100%	0%	0%
Test coverage analysis	0%	0%	100%
Testable	0%	0%	100%
Visionary and doer	100%	0%	0%

Para esse projeto, é esperado que o modelo sinalize que há uma probabilidade considerável do projeto falhar. A falta de competência do *dono do produto* em gerenciar o *backlog do produto*, liderar a equipe e as *reuniões de revisão* devem ser apontados como os motivos para a possível falha do projeto. As saídas do modelo são apresentadas na Tabela 5.20.

Tabela 5.20: Saídas do modelo para o Cenário 10

	Estados		
Fator	Bad	Moderate	Good
Code inspection quality	0%	0%	100%

Code quality	0%	0%	100%
Daily Scrum quality	0%	0%	100%
Development team competence	0%	1,70%	98,3%
Development team teamwork skills	0%	5,60%	94,4%
Product backlog is properly ordered	0%	82,6%	17,4%
Product backlog items are properly detailed	0%	100%	0%
Product backlog quality	0%	68,9%	31,1%
Product backlog management	0%	97,9%	2,10%
Product increment quality	1,00%	61,7%	37,3%
Product owner overall quality	19,3%	76,9%	3,80%
Product owner personal characteristics	71,6%	28,4%	0%
Product vision quality	0%	0%	100%
Project success	5,90%	86,1%	8,00%
Release plan	0%	0%	100%
Software engineering techniques quality	0%	1,10%	98,9%
Sprint backlog quality	0%	37,9%	62,1%
Sprint Planning quality	3,80%	96,2%	0%
Sprint progress	23,9%	76,1%	0%
Sprint Review meeting achieving its goals	22,8%	77,2%	0%
Work validation quality	13,1%	85,1%	1,80%

Observando os dados da Tabela 5.20, pode-se concluir que o modelo atende às expectativas. Há poucas chances do projeto obter sucesso - apenas 8,00% -, e os principais pontos apontados são o trabalho do *dono do produto* e a validação do produto. Além disto, a personalidade do *dono do produto* e sua capacidade de gerenciar o escopo são as principais causas de seu trabalho não ser adequado.

5.11 Ameaças à Validade

Há dois tipos de ameaças relativas à construção da *rede Bayesiana*: interna e externa. As ameaças internas referem-se às suposições que foram feitas para construir as TPN. As suposições relativas aos parâmetros μ e σ^2 da distribuição Normal duplamente truncada utilizada para modelar as causas entre os nós podem estar incorretas. As ameaças externas referem-se às conclusões relativas aos nós e arcos que compõem o GAD. Apesar da pesquisa

literária e consultas a especialistas, os conjuntos de nós e arcos identificados podem não ser o suficiente para modelar um projeto *Scrum* de desenvolvimento de software. Para minimizar essas ameaças, definiu-se cenários para aplicar o modelo e demonstrar a sua validade. Futuramente, estudos de caso em projetos reais serão realizados para reforçar a validação e minimizar essas ameaças.

Há dois tipos de ameaças relativas ao estudo empírico: interna e externa. As ameaças internas referem-se ao controle de acesso ao questionário e à qualidade das perguntas e respostas do mesmo. Com relação ao controle de acesso, não há nada que impeça que alguém responda ao questionário mais de uma vez ou o publique em um meio inadequado. Com relação à qualidade das perguntas e respostas, perguntas ambíguas podem confundir os participantes e, consequentemente, coletar respostas prejudiciais ao estudo. Além disso, as respostas podem ser inadequadas se o respondente não responder com a dedicação adequada ou não tiver o conhecimento necessário. Para minimizar essas ameaças, o questionário apenas foi publicado em meios onde há pessoas com conhecimento e interesse no *Scrum*. Ao publicar o questionário, especificou-se que apenas pessoas com experiência em projetos de desenvolvimento de software em empresas devem participar. Além disto, trata-se de um questionário relativamente longo (gasta-se, aproximadamente, quinze minutos para respondê-lo), e foi assumido que apenas as pessoas interessadas no estudo investiram seu tempo para respondê-lo por completo.

As ameaças externas referem-se ao alcance da notificação sobre a existência da survey e a baixa taxa de respostas. Com certeza, nem todas as pessoas que têm experiência com *Scrum* serão notificados da survey. É fora da realidade tentar atingir todas essas pessoas. Desta forma, afirmar que o resultado do estudo representa a opinião de especialistas, que têm experiência com *Scrum* no geral, pode ser precipitado. Além disso, a baixa taxa de respostas pode invalidar os resultados do estudo. Isso pode significar que as pessoas que são mais ocupadas não se interessaram em usar o tempo para responder a survey, ou seja, a survey pode ter sido respondida apenas por pessoas menos ocupadas. Pode ser que essas pessoas menos ocupadas tenham menos conhecimento na área. Não há uma estimativa de quantas pessoas tiveram acesso à survey. Desta forma, não é possível calcular a taxa de respostas. Desta forma, supõem-se que o conjunto de participantes do estudo é composto apenas por pessoas interessadas no estudo e com conhecimento adequado.

Capítulo 6

Validação do Método

A validação do método foi realizada a partir de um estudo de caso realizado em dois projetos de uma empresa, que não será identificada para manter a confidencialidade, ou seja, há duas unidades de análise. Essa empresa usa *Scrum* há três anos na maioria dos projetos de desenvolvimento de software. Esse estudo durou quarenta e cinco dias (que é equivalente a três sprints). Cada projeto é composto por uma equipe de desenvolvimento. O escopo de um dos projetos é desenvolver um serviço na nuvem e um cliente web. Esse projeto é composto por seis desenvolvedores. O escopo do outro projeto é desenvolver clientes em iOS, Android e sistemas embarcados para o serviço web desenvolvido pela equipe do outro projeto. Esse projeto é composto por cinco desenvolvedores. O objetivo do estudo de caso é responder as questões de pesquisa:

1. O método é robusto para identificar problemas na aplicação do *Scrum*?
2. O custo para alterar o modelo de forma a melhorar a representação do *Scrum* para um projeto específico é aceitável?
3. O método é útil para auxiliar o *ScrumMaster* a guiar a equipe na busca por excelência?
4. O custo-benefício de usar o método é positivo?

As proposições do estudo são:

1. O método é robusto para identificar problemas na aplicação do *Scrum*

2. O custo para alterar o modelo de forma a melhorar a representação do *Scrum* para um projeto específico é aceitável

3. O método é útil para auxiliar o *ScrumMaster* a guiar a equipe na busca por excelência

4. O custo-benefício de usar o método é positivo

Para cada sprint, os seguintes passos foram executados:

1. *ScrumMaster* com a participação opcional de outros membros da equipe de projeto analisam o modelo para verificar se esse precisa ser alterado para refletir o projeto corretamente. Neste passo, as técnicas de engenharia e de gerenciamento do produto e projeto podem ser alteradas. Recomenda-se que as regras e princípios do Scrum não sejam alteradas, salvo casos especiais.

2. Se necessário, o modelo deve ser alterado. Para tal, inicialmente, os nós e arcos devem ser alterados para que as necessidades do projeto sejam satisfeitas. Depois que o grafo estiver coerente com as necessidades, as tabelas de probabilidades dos nós (TPN) devem ser definidas. Recomenda-se que essas sejam definidas utilizando expressões ponderadas, a partir da ordenação das influências via discussões entre os envolvidos, e o algoritmo para gerar TPN apresentado nesta dissertação seja utilizado.

3. *ScrumMaster* observa a equipe e alimenta o modelo com auxílio do *dono do produto* e de alguns membros da equipe de desenvolvimento.

4. Durante a reunião de retrospectiva, os resultados calculados pela ferramenta que implementa o modelo são apresentados para a equipe. Durante essa reunião, a equipe irá discutir os resultados, avaliar se os resultados representam a realidade do projeto e escolher ações corretivas e preventivas.

5. *ScrumMaster* guia execução das ações durante a execução da sprint.

Esse processo foi repetido três vezes (durante três sprints) para cada projeto. Ao final do estudo de caso, os *ScrumMasters* de seus respectivos projetos responderam as questões de pesquisa. Para responder as questões de pesquisa, para cada pergunta, os *ScrumMasters*

tiveram uma escala Likert com quatro opções para expressar sua opinião e espaço para justificar e evidenciar suas respostas. As proposições de estudo foram verificadas a partir das respostas dos *ScrumMasters*. Para analisar as respostas, triangulação para analisar as áreas de divergências e convergências foi realizada.

Com relação às ameaças à validade desse estudo de caso, destacam-se duas: de construção e externa. A ameaça de construção refere-se ao risco das opiniões dos *ScrumMasters* não serem suficientes para validar ou refutar as proposições de pesquisa. A validade externa refere-se à incapacidade de generalização. Dado que o estudo é realizado em apenas dois projetos de uma empresa, não se deve confiar que o estudo de caso valide o modelo para ser utilizado por qualquer empresa que utiliza *Scrum* em projetos de desenvolvimento de software.

6.1 Dados da Execução do Estudo de Caso

Para manter a confidencialidade do estudo, além da empresa participante do estudo de caso não ser identificada, alguns resultados do processo de utilização do método foram omitidos. Isto é resultado do acordo firmado entre o pesquisador e a empresa para poder realizar o estudo. Por outro lado, apesar das omissões, a verificação do resultado do estudo de caso não foi prejudicada. Além disso, os dois projetos observados são executados em paralelo e suas *sprints* são sincronizadas.

Inicialmente, o pesquisador se reuniu com o *ScrumMaster*, *dono do produto* e um líder da equipe de desenvolvimento de cada projeto para esclarecer os passos do estudo. Ao fim dos esclarecimentos, sem o auxílio do pesquisador e separadamente pra cada projeto, (i) o modelo foi analisado para verificar se esse precisava ser alterado para refletir o projeto corretamente, (ii) o modelo foi alterado quando necessário e (iii) o modelo foi alimentado com dados do projeto. Todos os passos citados foram realizados em uma reunião que ocorreu durante a execução da primeira *sprint* na qual o estudo de caso foi realizado. Cada reunião foi limitada em uma hora, e esse tempo foi o suficiente para tal.

Com o modelo de cada projeto alimentado, os resultados desse foram calculados e apresentados nas respectivas *reuniões de retrospectiva*, que não sofreram modificações nas suas durações. Durante as reuniões, os resultados dos modelos foram analisados para detectar

os problemas dos projetos com o intuito de definir ações corretivas e preventivas para aumentar a probabilidade de sucesso desses. Dessa forma, os resultados do modelo serviram como uma fonte a mais de informação para facilitar na identificação de oportunidades de melhorias, que é um dos objetivos das *reuniões de retrospectiva*. Com as informações colhidas pela análise do modelo, oportunidades de melhoria foram identificadas e uma lista com ações corretivas e preventivas foi definida. Em ambos os projetos, alguns pontos de ações já tinham sido identificados no passado, mas por diversos motivos, não tinham sido colocados em prática.

Depois de ter identificado as ações corretivas e preventivas, o próximo passo foi desenvolver um plano de execução das ações. Para tal, o modelo foi utilizado para avaliar o impacto no projeto de ter cada ação executada e, dessa forma, priorizar a lista de ações. Para avaliar o impacto no projeto de ter cada ação executada, as informações referentes a cada ação foram modificadas no modelo com o intuito de prever as consequências da execução dessas na probabilidade de sucesso do projeto. As ações foram priorizadas com relação à magnitude do impacto de suas respectivas execuções na probabilidade de sucesso do projeto. Após ter a lista de ações priorizada, os participantes das reuniões definiram quais ações seriam colocadas em prática durante a execução da próxima *sprint*.

Durante a segunda *sprint*, os *ScrumMasters* lideraram suas respectivas equipes para colocarem em prática os planos de ações definidos nas *reuniões de retrospectiva*. Essa atividade faz parte do dia-a-dia de um *ScrumMaster* independente de utilizar o método proposto nesse trabalho. Ao final da *sprint*, antes da *reunião de retrospectiva*, os *ScrumMasters* se reuniram com o *dono do produto* e um líder da equipe de desenvolvimento de seus respectivos projetos para analisar e alimentar novamente o modelo para refletir as mudanças que ocorreram desde o final da *sprint* anterior. Cada reunião foi limitada em quinze minutos, e esse tempo foi o suficiente para tal.

Durante as *reuniões de retrospectiva* da segunda *sprint*, os modelos atualizados, assim como o resultado da execução dos pontos de ação definidos ao final da *sprint* passada foi apresentado e discutido. Como resultado da análise do modelo e discussão, novos pontos de ações foram identificados. Em uma das reuniões, dentre os novos pontos de ações, alguns não se referiram aos resultados do modelo, mas de observações do dia-a-dia dos desenvolvedores. Os pontos de ações identificados que não se referiram aos resultados do modelo foram

relacionados ao gerenciamento de *bugs* e novas requisições. Os novos pontos de ações foram adicionados à lista de pontos de ações, priorizados e, ao final da reunião, as equipes escolheram alguns itens do topo da lista para executar durante a *sprint*.

Da mesma forma que na segunda *sprint*, os *ScrumMasters* lideraram suas respectivas equipes para colocarem em prática os planos de ações definidos na *reunião de retrospectiva* e, ao final da *sprint*, os modelos foram atualizados para refletirem as mudanças que ocorreram durante a mesma. A *reunião de retrospectiva* seguiu o mesmo formato da anterior: discussão do modelo atualizado e resultado da execução dos pontos de ação na *sprint*, identificação de oportunidades de melhorias, priorização da lista de pontos de ação, e escolha de pontos de ação a serem executados durante a *sprint*.

6.2 Resultados e Discussão do Estudo de Caso

Como dito anteriormente, ao final do estudo de caso, os *ScrumMasters* utilizaram uma escala Likert com quatro opções para expressar sua opinião e espaço para justificar e evidenciar suas respostas. As proposições de estudo foram verificadas a partir das respostas dos *ScrumMasters*. Na Figura 6.1 são apresentados os dados coletados.

Observando os dados coletados, pode-se concluir que as quatro proposições do estudo foram validadas, ou seja, o estudo de caso obteve sucesso. No caso da robustez do modelo para identificar problemas na aplicação do *Scrum*, ambos *ScrumMasters* afirmaram que a aplicação do método os ajudou a identificar pontos de ações que tinham passado despercebidos. Isso confirma que o método auxilia na detecção de problemas. Além disso, um dos *ScrumMasters* afirmou que a aplicação do método identificou pontos de ações que já tinham sido identificados pela equipe. Isso evidencia que os problemas identificados são válidos.

Os *ScrumMasters* afirmaram que não precisaram modificar muito o modelo para utilizá-lo nas suas respectivas equipes. Isso segue o objetivo dessa pesquisa de criar um modelo genérico que pudesse ser aplicado para qualquer projeto *Scrum* de desenvolvimento de software sem necessitar de muitas modificações. Além disso, segundo os *ScrumMasters*, o esforço necessário para realizar as modificações necessárias e alimentar o modelo é compensado pelos resultados obtidos pela utilização do método.

1. O método é robusto para identificar problemas na aplicação do Scrum?			
Sim	Precisa de algumas melhorias	Precisa de muitas melhorias	Não
2	0	0	0
Comentários			
Foi capaz de identificar pontos que já tinham chamado nossa atenção no passado e alguns que tinham passados despercebidos.			
Conseguimos identificar alguns problemas que tinham passados despercebidos e que eram relevantes.			
2. O custo para alterar o modelo de forma a melhorar a representação do Scrum para um projeto específico é aceitável?			
Sim	Precisa de algumas melhorias	Precisa de muitas melhorias	Não
2	0	0	0
Comentários			
Não precisamos modificar muito o modelo.			
No nosso caso, o modelo não precisou de muitas modificações.			
3. O método é útil para auxiliar o ScrumMaster a guiar a equipe na busca por excelência?			
Sim	Precisa de algumas melhorias	Precisa de muitas melhorias	Não
2	0	0	0
Comentários			
A análise e alimentação do modelo ajudam o ScrumMaster a ter uma visão geral do projeto e prestar atenção em pontos que podem passar despercebidos, principalmente para ScrumMasters inexperientes. Além disso, a utilização do modelo nas reuniões de retrospectiva ajudam na identificação de oportunidades de melhoria.			
É uma ferramenta adicional que acredito que pode auxiliar na identificação de problemas e pontos de melhoria. Acredito que o método pode ser útil para ScrumMasters.			
4. O custo-benefício de usar o método é positivo?			
Sim	Precisa de algumas melhorias	Precisa de muitas melhorias	Não
2	0	0	0
Comentários			
Os únicos custos adicionais referentes à aplicação do método referem-se ao tempo necessário para entender, modificar e editar o modelo. Dado o tempo que o método foi avaliado no projeto que trabalhei, verificou-se que esse tempo é compensado pelo auxílio que o modelo teve para facilitar a identificação de oportunidades de melhoria.			
O esforço adicional necessário pra utilizar o método resume-se a alguns minutos por sprint pra analisar e alimentar o modelo. No nosso caso, esse esforço foi compensado pelos pontos de ações que foram identificados nas reuniões de retrospectiva.			

Figura 6.1: Dados coletados no estudo de caso

Capítulo 7

Considerações Finais

Neste trabalho, propõe-se um método baseado em modelo probabilístico ara auxiliar na detecção de problemas na aplicação do *Scrum* em projetos de desenvolvimento de software. O método é composto de quatro etapas: análise do modelo, alimentação do modelo, análise das saídas do modelo e aplicação de ações corretivas e preventivas. Seu objetivo é que sua utilização exponha os problemas na aplicação do *Scrum* de forma a auxiliar o *Scrum-Master* a liderar a equipe na busca por excelência. O método limita-se apenas a projetos de desenvolvimento de software que utilizam *Scrum* composto por apenas uma equipe de desenvolvimento e apenas considera fatores relacionados à utilização do *Scrum* para monitorar a saúde do projeto e calcular a probabilidade de sucesso do mesmo. A aplicação do método requer a utilização de um modelo probabilístico que representa um projeto de desenvolvimento de software usando *Scrum* e técnicas e processos complementares necessários. Dado que diferentes projetos necessitam de diferentes técnicas e processos complementares para utilizar o *Scrum*, este trabalho apresenta um modelo genérico que requer pouco esforço para ser modificado para representar qualquer projeto de software que utiliza *Scrum*.

O modelo trata-se de uma *Rede Bayesiana*. Para construí-lo, uma lista de fatores chave em projetos *Scrum* de desenvolvimento de software e seus relacionamentos foram definidos a partir de uma pesquisa literária e refinada com um grupo de onze especialistas. Após refinar a lista de fatores chaves e seus relacionamentos, o GAD da *Rede Bayesiana* foi construído. Além disso, um questionário foi publicado para coletar a opinião de quarenta especialistas e quantificar a intensidade dos relacionamentos. Os dados coletados foram estatisticamente analisados e utilizados para construir as TPN.

A modelagem foi validada a partir da comparação dos resultados esperados e calculados da aplicação do modelo em dez cenários, que representam possíveis casos reais de projetos. Em todos os cenários, o modelo obteve sucesso. O método foi validado a partir de um estudo de caso em dois projetos de uma empresa. Para ambos os projetos, o método obteve sucesso.

7.1 Trabalhos em Andamento e Sugestões para Trabalhos Futuros

Atualmente, uma ferramenta específica para a utilização do modelo está sendo implementada. Com essa ferramenta, a realização de estudos de caso em diversas empresas será viável. Os estudos de caso são necessários para mitigar a ameaça à validade externa do método.

Bibliografia

[1] M. Abouelela and L. Benedicenti. Bayesian Network Based XP Process Modelling. *International Journal of Software Engineering and Applications*, vol. 1, no. 3, pp. 1-15, 2010.

[2] D.J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*, First Edition. Blue Hole Press. (2010)

[3] X. Bai , L. Huang, , H. Zhang, and S. Koolmanojwong. Hybrid modeling and simulation for trustworthy software process management: a stakeholder-oriented approach. *Journal of Software: Evolution and Process*, vol. 24, no. 7, pp. 721-740, 2012.

[4] J.E. Bartlett, II, J.E Kotrlik and C.C Higgins. Organizational Research: Determining Appropriate Sample Size in Survey Research. *Information Technology, Learning, and Performance Journal*, vol. 19, no. 1, pp. 43-50, 2001.

[5] K. Beck. *Extreme Programming Explained: Embrace Change*, First, Addison Wesley, 2000.

[6] I. Ben-Gal. Bayesian Networks. In F. Ruggeri, F. Falti and R. Kenett (Eds), *Encyclopedia of Statistics in Quality & Reliability*, Wiley & Sons, 2007.

[7] B. Boehm. Software Engineering Is a Value-Based Contact Sport. *IEEE Software*, vol. 19, no. 5, pp. 95-97, 2002.

[8] G. Büyüközkan and D. Ruan. Choquet integral based aggregation approach to software development risk assessment. *Journal of Information Sciences*, vol. 180, no. 3, pp. 441-451, 2010.

[9] M. Cohn. *Agile Estimating and Planning*, First Edition, Prentice Hall, 2005.

[10] M. Cohn. *Succeeding With Agile Software Development Using Scrum*, First Edition, Addison Wesley, 2009.

[11] S. Dong , and B. Hu. Multi-Agent Based Simulation of Team Effectiveness in Team's Task Process: A Member-Task Interaction Perspective. *International Journal of Simulation and Process Modelling*, vol. 4, no. 1, pp. 54-68, 2008.

[12] K. El Emam and A.G. Koru. A Replicated Survey of IT Software Project Failures. *IEEE Software*, September/October, vol. 25, no. 5, pp: 84-90, 2008.

[13] C. Fan and Y. Yu. BBN-based software project risk management. *Journal of Systems and Software*, vol. 73, no. 2, pp. 193-203, 2004.

[14] N.E. Fenton, M. Neil and J.S. Caballero. Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks. *Proc. of IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, Issue 10, pp: 1420-1432, 2007.

[15] N. Friedman, D. Geiger and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, no. 29, pp. 131-163, 1997.

[16] P. Hearty, N. Fenton, D. Marquez and M. Neil. Predicting project velocity in XP using a learning dynamic Bayesian network model. *IEEE Transactions on Software Engineering*, vol. 35, no. 1, pp. 124-137, 2009.

[17] M. Hollander and D. A. Wolfe. *Nonparametric Statistical Methods*, Second Edition, New York: John Wiley & Sons, 1999.

[18] D.X Houston, G.T Mackulak and J.S. Collofello. Stochastic simulation of risk factor potential effects for software development risk management. *Journal of Systems and Software*, vol. 59, no. 3, pp. 247-257, 2001.

[19] K. Jeet, N. Bhatia and R.S. Minhas. A bayesian network based approach for software defects prediction. *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 4, pp. 1-5, 2011.

[20] K. Jeet, N. Bhatia and R.S. Minhas. A model for estimating the impact of low productivity on the schedule of a software development project. *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 4, pp. 1-6, 2011.

[21] M.I. Kellner, P.H. Feiler, A. Finkelstein, T. Katayama, and L.J. Osterweil.Ispw-6 software process example. in *Proceedings of the First International Conference on the Software Process*, pp. 176-186. 1991.

[22] M.I. Kellner, R.J. Madachy, D.M. Raffo. Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, vol. 46, no. 2-3, pp. 91-105, 1999.

[23] K.G. Kouskouras and A.C. Georgiou. A discrete event simulation model in the case of managing a software project. *Journal of Operational Research*, vol. 181, no. 1, pp. 374-389, 2007.

[24] S. Kuppuswami, K. Vivekanandan, P. Ramaswamy and P. Rodrigues. The effects of individual XP practices on software development effort. *ACM SIGSOFT Softw. Eng. Notes*, vol. 28, no. 6, pp. 1-6, 2003.

[25] C. Ladas. *Scrumban: Essays on Kanban Systems for Lean Software Development*, First Edition, Modus Cooperandi Press, 2009.

[26] F. Mariotti. Os perigos por trás do ScrumBut *Revista Engenharia de Software*, no. 50, 2012.

[27] R.C. Martin. *Agile Software Development: Principles, Patterns, and Practices*, First Edition, Prentice Hall, 2002.

[28] M. Melis. *A Software Process Simulation Model of Extreme Programming*, PhD Thesis, Universita' di Cagliari, 2006.

[29] R. Mulcahy. *PMP Exam Prep*, Seventh Edition, RMC Publications, 2011.

[30] A. Nagy, M. Njima and I. Mkrtchyan. A Bayesian Based Method for Agile Software Development Release Planning and Project Health Monitoring. in *Proceedings of the 2010 International Conference on Intelligent Networking and Collaborative Systems*, Thessaloniki, Grécia. 2010.

[31] L.J. Osterweil. Jil and little-jil process programming languages. in *EWSPT 98: Proceedings of the 6th European Workshop on Software Process Technology*, Londres, UK. 1998.

[32] J. Pearl and S. Russel. Bayesian Networks. *Handbook of Brain Theory and Neural Networks*, M. Arbib, ed, MIT Press, pp.157-160, 2001.

[33] P.C. Pendharkar, G.H. Subramanian and J.A. Rodger. A Probabilistic Model for Predicting Software Development Effort. *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 615-624, 2005.

[34] M. Perkusich, A. Perkusich, and H. Almeida. Using Survey and Weighted Functions to Generate Node Probability Tables for *Bayesian* Networks. *Proc. of BRICS-CCI 2013*, 2013.

[35] M. Perkusich, H. Almeida, and A. Perkusich. A model to detect problems on scrum-based software development projects. *Proc. of the 28th Annual ACM Symposium on Applied Computing*, pp. 1037-1042, 2013.

[36] R. Pichler, *Agile Product Management with Scrum: Creating Products That Customers Love*, First Edition, Addison Wesley, 2010.

[37] Pinto, A. *PMSURVEY.ORG - 2012*. http://www.pmsurvey.org , 2012.

[38] K. Schwaber, *Scrum is Hard and Disruptive*. http://www.controlchaos.com/storage/scrum-articles/Scrum%20Is%20Hard%20and%20Disruptive.pdf , 2006.

[39] K. Schwaber and J. Sutherland. *The Scrum Guide*. http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf , 2011.

[40] Smart Software Inc., *Peer Code Review: An Agile Process*. http://smartbear.com/SmartBear/media/pdfs/WP-CC-Peer-Code-Review-An-Agile-Process.pdf , Acessado em: 18/4/2013.

[41] B. Spasic and B. Onggo. AGENT-BASED SIMULATION OF THE SOFTWARE DEVELOPMENT PROCESS: A CASE STUDY AT AVL. in *Proceedings of the 2012 Winter Simulation Conference*, Berlim, Alemanha. 2012.

[42] The Standish Group. *The CHAOS Manifesto*, 2012.

[43] Version One. *7th Annual State of Agile Development Survey Results*, 2013.

[44] L. Williams. Agile Software Development Methodologies and Practices. *Advances in Computers*, vol. 80, pp. 1-44, 2010.

[45] M. Wu and H. Yan. Simulation in Software Engineering with System Dynamics: A Case Study. *Journal of Software*, vol. 4, no. 10, pp. 1127-1135, 2009.

[46] H. Zhang, B. Kitchenham and D. Pfhal. Reflections on 10 years of software process simulation modeling: a systematic review. in *ICSP'08 Proceedings of the Software process, 2008 international conference on Making globally distributed software development a success story*, Leipzig, Alemanha. 2008.

Apêndice A

Lista de Estados Para Cada Nó da Rede Bayesiana

Tabela A.1: Estados identificados

	Estados		
Nó	Bad	Moderate	Good
15 minutes limit	Never or rarely	Sometimes	Most of the times or always
3 to 5 top features described	Not defined	Somewhat defined	Clearly defined

Acceptance criteria check	Rarely or never	Sometimes	Most of the times or always
All participants present	Rarely or never	Sometimes	Most of the times or always
Answer the 3 basic questions	Rarely or never	Sometimes	Most of the times or always
Automated tests	Very few or none	Some tests are automated	All or more important test cases
Available and qualified	No	Somewhat	Yes
Broad and engaging goal	No	Somewhat	Yes
Broad and realistic goal	No	Somewhat	Yes
Clear acceptance criteria	No	Some items but not all items that enter the sprint	All or top priority items
Clear and stable	No	Somewhat	Yes

Code inspection quality	Low	Medium	High
Code integration frequency	Rare	Somewhat frequent	Continuous or very frequent
Code quality	Low	Medium	High
Code refactoring	No	Sometimes	Is part of the development process
Collective ownership	Team members do not feel responsible	Most part of the team members feels responsible	Entire team feels responsible
Communicator and negotiator	No	Somewhat	Yes
Considers business value	No	Some items	Yes
Considers non-functional requirements when necessary	No	Some items	Yes
Considers risk	No	Some items	Yes

Considers technical dependencies	No	Some items	Yes
Continuous improvement	Never or rarely	Sometimes	Continuously
Critical attributes to satisfy customer needs described	No	Somewhat	Yes
Daily Scrum quality	Low	Medium	High
Development team competence	Low	Medium	High
Development team teamwork skills	Low	Medium	High
Documentation	Very few or too much documentation	Some important things are documented	Whenever necessary
Empowered and committed	No	Somewhat	Yes
Estimated size is small	No	Some items but not all items that enter the sprint	All or top priority items

Independent	No	Some items but not all items that enter the sprint	All or top priority items
Launch date defined	No	Yes and it is not a relatively short period	Yes and it is a relatively short period
Leader and team player	No	Somewhat	Yes
Members expertise	Bad	Moderate	Good
Members motivation	Bad for most of the members	Moderate	Good for most or all the members
Members personality	Bad for most of the members	Moderate	Good for most or all the members
Monitor sprint progress	Never or rarely	Sometimes	Most of the times or always
Negotiable	No	Somewhat	Yes
Pair programming	Never	Sometimes	Is part of the development process

Peer code review	Never	Sometimes	Is part of the development process
Product backlog is emergent	No	Somewhat	Yes
Product backlog items are detailed properly	No	Some items but not all items that enter the sprint	All or top priority items
Product backlog is estimated	No	Some items but not all items that enter the sprint	All or top priority items
Product backlog management	Bad	Moderate	Good
Product backlog is properly ordered	No	Some items but not all items that enter the release	All or top priority items
Product backlog quality	Low	Medium	High
Product increment quality	Bad	Moderate	Good
Product Owner personal characteristics	Bad	Moderate	Good

Product Owner overall work quality	Bad	Moderate	Good
Product vision quality	Low	Medium	High
Progress tracking	No	Somewhat	Yes
Project success	Bad	Moderate	Good
Release plan	Bad	Moderate	Good
Short and concise	No	Somewhat	Yes
Size estimation quality	Low	Medium	High
Software engineering techniques quality	Low	Medium	High
Sprint backlog quality	Low	Medium	High

Sprint items and priority stability	Frequently	Sometimes	Never or very rarely
Sprint goal check	Rarely or never	Sometimes	Most of the times or always
Sprint length	Longer than 4 weeks	Less than 4 weeks but could be improved	Perfect size (less than 4 weeks and fits the needs)
Sprint planning quality	Low	Medium	High
Sprint progress	Bad	Medium	Good
Sprint Review meeting achieves its goals	Never or rarely	Sometimes	All or most of the times
Stakeholder feedback outside of the Sprint Review meeting	Never or rarely	Sometimes but it could be improved	Whenever necessary
Stakeholders feedback	Never or rarely	Sometimes	All or most of the times
Static code analysis	Never	Sometimes	Is part of the development process

Task breakdown quality	Low	Medium	High
Team physical distribution	Team is very distributed	A member or two are located on a different place	All members are co-located
Team size	Less than 5 or more than 9	Between 5 and 9, but could be improved	Perfect size (between 5 and 9 and fits the needs)
Team velocity defined	No	Somewhat	Yes
Test coverage analysis	No	Somewhat	Is part of the development process
Testable	No	Some items but not all items that enter the sprint	All or top priority items
Visionary and doer	No	Somewhat	Yes
Work validation quality	Bad	Moderate	Good

Apêndice B

Questões da Survey

As perguntas e respectivas tabelas do questionário são apresentadas nesse capítulo. O questionário foi disponibilizado em inglês, dado que os participantes tinham diferentes línguas nativas. Por isso, os dados apresentados nas figuras que seguem estão em inglês.

What is the influence of the following factors on the factor "Development team teamwork skills"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Members motivation	○	○	○	○	○
Team size	○	○	○	○	○
Members expertise	○	○	○	○	○
Collective ownership	○	○	○	○	○
Team physical distribution	○	○	○	○	○
Members personality	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Development team teamwork skills	Development team skills that will make them capable to work as a team
Collective ownership	Entire team responsible for code, design and to contribute with new ideas
Members expertise	Development team members technical knowledge and experience
Members motivation	Development team members motivation to work in the company and product
Members personality	Development team members personal characteristics, such as: introverted, self-giving, sincere, respectful, etc.
Team physical distribution	Physical distribution (co-located or not)
Team size	Number of members in the development team

Figura B.1: Primeira pergunta do questionário

What is the influence of the following factors on the factor "Code inspection quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Pair programming	○	○	○	○	○
Static code analysis	○	○	○	○	○
Peer code review	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Code inspection quality	Refers to the Development team's expertise to inspect the code they develop. Inspect the code means examine the source code in order to find and fix mistakes overlooked in the initial development phase.
Pair programming	A technique in which two developers work together at one workstation
Peer code review	A technique in which team members inspects code implemented by another team member
Static code analysis	Team uses tools (such as Sonar) for code analysis (comments, duplication, complexity, coding standards, etc).

Figura B.2: Segunda pergunta do questionário

What is the influence of the following factors on the factor "Code quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Test coverage analysis	○	○	○	○	○
Documentation	○	○	○	○	○
Code refactoring	○	○	○	○	○
Automated tests	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Code quality	Refers to code´s atributes that maximizes the chances of project success, such as: bug-proneness, readibility, requirements compliance, efficiency, flexibility, etc.
Code refactoring	A technique for restructuring an existing body of code, altering its internal behavior without changing its external behavior.
Automated tests	A tecnique that uses software to control the execution of tests, compare of actual outcomes to predicted outcomes, set up of test preconditions, and other test control and test reporting functions.
Test coverage analysis	A software testing technique that analysis the area of code that is covered by test cases. This technique looks to create additional test cases to increase coverage whenever it is beneficial.
Documentation	Refers to comments on source code and any documentation regarding any technical work performed by the development team (test reports, design, research documents, etc)

Figura B.3: Terceira pergunta do questionário

What is the influence of the following factors on the factor "Software engineering techniques quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Code quality	○	○	○	○	○
Code integration frequency	○	○	○	○	○
Code inspection quality	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Software engineering techniques quality	Refers to software engineering techniques used by the development team that maximizes the chances of project success
Code inspection quality	Refers to the Development team's expertise to inspect the code they develop. Inspect the code means examine the source code in order to find and fix mistakes overlooked in the initial development phase.
Code integration frequency	Frequency that new or changed code is integrated with the existing code.
Code quality	Refers to code´s atributes that maximizes the chances of project success, such as: bug-proneness, readibility, requirements compliance, efficiency, flexibility, etc.

Figura B.4: Quarta pergunta do questionário

What is the influence of the following factors on the factor "Development team competence"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Software engineering techniques quality	○	○	○	○	○
Pursue for continuous improvement	○	○	○	○	○
Development team teamwork skills	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Development team competence	Team's capacity to produce high quality work within the project's restrictions and goals
Development team teamwork skills	Development team skills that will make them capable to work as a team
Pursue for continuous improvement	Refers to the development team's tools to pursue continuous improvement. Retrospective meetings that achieve its purpose are an example of tool.
Software engineering techniques quality	Refers to software engineering techniques used by the development team that maximizes the chances of project success

Figura B.5: Quinta pergunta do questionário

What is the influence of the following factors on the factor "Daily Scrum quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Monitor sprint progress	○	○	○	○	○
Fifteen minutes limit	○	○	○	○	○
Answer the three basic questions	○	○	○	○	○
All participants present	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Daily Scrum quality	Daily Scrum achieves its purpose to keep teams focused on their objectives and to help them avoid being thrown off track by less important concerns.
All participants present	ScrumMaster, Development team and Product Owner are present on the Daily Scrum
Answer the three basic questions	During the Daily Scrum, all the Development team members anwers the three questions: 1)What has been accomplished since last meeting? 2)What will be done before the next meeting? 3) What obstacles are in the way?
Fifteen minutes limit	Daily Scrum is time-boxed to fifteen minutes
Monitor sprint progress	Development team monitors sprint progress (for example, updates Sprint Burndown chart)

Figura B.6: Sexta pergunta do questionário

What is the influence of the following factors on the factor "Sprint progress"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Development team competence	○	○	○	○	○
Daily Scrum quality	○	○	○	○	○
Sprint items and priority not changed during sprint	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Sprint progress	Refers to the progress that the team achieves during the sprint
Sprint items and priority not changed during sprint	Sprint backlog items (not technical tasks!) remains unchanged during the sprint
Daily Scrum quality	Daily Scrum achieves its purpose to keep teams focused on their objectives and to help them avoid being thrown off track by less important concerns.
Development team competence	Team's capacity to produce high quality work within the project's restrictions and goals

Figura B.7: Sétima pergunta do questionário

What is the influence of the following factors on the factor "Sprint backlog quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Task breakdown quality	○	○	○	○	○
Size estimation quality	○	○	○	○	○
Product backlog items are detailed properly	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Sprint backlog quality	Refers to how well the product backlog items were allocated and planned to be delivered during the sprint in order to maximize the chance of achieving the sprint goal
Task breakdown quality	Refers to team's ability and techniques used to breakdown Product backlog items into technical tasks
Size estimation quality	Refers to team's ability and techniques used to estimate the size of Product backlog items and technical tasks
Product backlog items are detailed properly	Refers to how clear and concise the product backlog items represent the expected work in order to consider it 'done'

Figura B.8: Oitava pergunta do questionário

What is the influence of the following factors on the factor "Sprint planning quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Team velocity defined	○	○	○	○	○
Sprint backlog quality	○	○	○	○	○
Sprint length	○	○	○	○	○
Broad and realistic goal	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Sprint planning quality	After the Sprint planning meeting, the team should have the answers for these two questions: 1) What needs to be delivered in the increment resulting from the upcoming sprint? 2)How will the work needed to deliver the increment be achieved?
Sprint backlog quality	Refers to how well the product backlog items were allocated and planned to be delivered during the sprint in order to maximize the chance of achieving the sprint goal
Team velocity defined	Team velocity is measured given historical data (data from past sprints)
Broad and realistic goal	Team should be able to maneuver and still be valid if the team does not commit to all top product backlog items
Sprint length	Number of weeks/days that composes the sprint

Figura B.9: Nona pergunta do questionário

What is the influence of the following factors on the factor "Potentially shippable product increment quality (end of each sprint)"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Sprint progress	○	○	○	○	○
Sprint planning quality	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Potentially shippable product increment quality (end of each sprint)	Product increment delivered by the end of the sprint compliance with acceptance criterias
Sprint planning quality	After the Sprint planning meeting, the team should have the answers for these two questions: 1) What needs to be delivered in the increment resulting from the upcoming sprint? 2)How will the work needed to deliver the increment be achieved?
Sprint progress	Refers to the progress that the team achieves during the sprint

Figura B.10: Décima pergunta do questionário

What is the influence of the following factors on the factor "Sprint Review meeting achieves its goals"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Check if sprint goal was achieved	○	○	○	○	○
Stakeholders feedback	○	○	○	○	○
Check if acceptance criteria for sprint items were achieved	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Sprint Review meeting achieves its goals	Sprint Review meeting purpose is to give stakeholders the opportunity to check the product's current state and give feedback to ensure the project is following the correct path
Check if acceptance criteria for sprint items were achieved	During the Sprint Review meeting, the acceptance criteria for each product backlog item allocated for the sprint is inspected
Check if sprint goal was achieved	During the Sprint Review meeting, the PO checks if the sprint goal was achieved
Stakeholders feedback	During the Sprint Review meeting, the stakeholders checks the product's current state and gives input regarding the increment created during the sprint and the product envisioned

Figura B.11: Décima primeira pergunta do questionário

What is the influence of the following factors on the factor "Work validation quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Stakeholder feedback outside of the Sprint Review meeting	○	○	○	○	○
Sprint Review meeting achieves its goals	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Work validation quality	Refers to activities that checks if the correct product is being developed
Sprint Review meeting achieves its goals	Sprint Review meeting purpose is to give stakeholders the opportunity to check the product's current state and give feedback to ensure the project is following the correct path
Stakeholder feedback outside of the Sprint Review meeting	Refers to statekolders giving their input regarding the product being created outside of the Sprint Review meeting

Figura B.12: Décima segunda pergunta do questionário

What is the influence of the following factors on the factor "Product backlog item properly detailed"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Testable	○	○	○	○	○
Clear acceptance criteria	○	○	○	○	○
Estimated size is small	○	○	○	○	○
Negotiable	○	○	○	○	○
Independent	○	○	○	○	○
Considers non-functional requirements when necessary	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Product backlog items properly detailed	Refers to how clear and concise the product backlog items represent the expected work in order to consider it 'done'
Testable	Product backlog item is written so as to be testable.
Negotiable	Product backlog item is seen as contracts or requirements that must be implemented. They have short descriptions of functionality, the details of which are to be negotiated in a conversation between the customer and the development team.
Independent	No dependencies between product backlog items
Estimated size is small	Product backlog item is estimated by the development team as small
Considers non-functional requirements when necessary	Product backlog item considers non-functional requirements (performance, reliability, availability, portability, scalability, usability, maintainability, etc) whenever it is necessary
Clear acceptance criteria	Product backlog item has clear acceptance criteria that will guide the team during its implementation

Figura B.13: Décima terceira pergunta do questionário

What is the influence of the following factors on the factor "Product backlog properly ordered"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Considers technical dependencies	○	○	○	○	○
Considers business value	○	○	○	○	○
Considers risk	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Product backlog properly ordered	Refers to how well the Product Backlog items are ordered
Considers business value	Product backlog is prioritized considering business value for each item
Considers risk	Product backlog is prioritized considering the risks for each item
Considers technical dependencies	Product backlog is prioritized considering technical dependencies between the items

Figura B.14: Décima quarta pergunta do questionário

What is the influence of the following factors on the factor "Product backlog management"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Product backlog items estimated	○	○	○	○	○
Product backlog is emergent	○	○	○	○	○
Backlog properly ordered	○	○	○	○	○
Product backlog items properly detailed	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Product backlog management	Refers to the work performed by the Product Owner to manage the Product Backlog in order to fulfill its purpose
Product backlog items properly detailed	Refers to how clear and concise the product backlog items represent the expected work in order to consider it 'done'
Backlog properly ordered	Refers to how well the Product Backlog items are ordered
Product backlog items estimated	Product backlog items' size are estimated by the development team using estimation techniques (such as story points, ideal days or others)
Product backlog is emergent	Product backlog changes over time. Changes on priorities and items can occur during the project execution and are reflected on the product backlog.

Figura B.15: Décima quinta pergunta do questionário

What is the influence of the following factors on the factor "Product vision quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Critical attributes to satisfy customer needs described	○	○	○	○	○
Broad and engaging goal	○	○	○	○	○
Short and concise	○	○	○	○	○
Clear and stable	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Product vision quality	Refers to how well the product vision represents the product that will be worked on during the project
Short and concise	Product vision contains only information critical to the success of the product.
Clear and stable	Product vision has specific and enduring parameters, called "product pillars", to guide the team.
Critical attributes to satisfy customer needs described	Product vision describes the critical attributes required to satisfy customer needs
Broad and engaging goal	Product vision guides the development efforts but leaves enough room for creativity; a goal that engages and inspires people

Figura B.16: Décima sexta pergunta do questionário

What is the influence of the following factors on the factor "Release plan"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Product vision quality	○	○	○	○	○
3 to 5 top features described	○	○	○	○	○
Launch date defined	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	**Definition**
Release plan	A plan for the product's release that will be worked on during the project
Product vision quality	Refers to how well the product vision represents the product that will be worked on during the project
Launch date defined	A date for the product launch (or project deadline) is set on the Release plan
3 to 5 top features described	Most important product's features (3 to 5 top) are described on the Release plan

Figura B.17: Décima sétima pergunta do questionário

What is the influence of the following factors on the factor "Product backlog quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Release plan	○	○	○	○	○
Product backlog management	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	Definition
Product backlog quality	Refers to how well the Product backlog represents everything that might be needed in the product in a way that maximizes the chances of project success
Product backlog management	Refers to the work performed by the Product Owner to manage the Product Backlog in order to fulfill its purpose
Release plan	A plan for the product's release that will be worked on during the project

Figura B.18: Décima oitava pergunta do questionário

What is the influence of the following factors on the factor "Desired personal characteristics of the Product Owner"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Communicator and negotiator	○	○	○	○	○
Leader and team player	○	○	○	○	○
Available and qualified	○	○	○	○	○
Empowered and committed	○	○	○	○	○
Visionary and doer	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	Definition
Desired personal characteristics of the Product Owner	The desired personal traits for a person playing the role of the Product Owner
Available and qualified	PO has enough time and abilities to fulfill his duties
Communicator and negotiator	Product owner communicates with different parties and bridges the gap between business and technical people
Empowered and committed	Product owner has enough authority and the right level of management sponsorship to lead the development effort and to align stakeholders. He must have the ability to create a work environment that forsts creativity and innovation
Leader and team player	Product owner provides guidance and direction for everyone involved in the development effort and ensures that tough decisions are made. He encourages the team to participate in the decision-making process.
Visionary and doer	Product owner is a visionary who can envision the final product and communicate the vision. He sees the vision through to completion.

Figura B.19: Décima nona pergunta do questionário

What is the influence of the following factors on the factor "Product Owner overall work quality"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Progress tracking	○	○	○	○	○
Product backlog quality	○	○	○	○	○
Product Owner personal characteristics	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	Definition
Product Owner overall work quality	Product owner competence to fulfull his duties. The Product Owner is responsible for maximizing the value of the product and the work of Development team.
Product Owner personal characteristics	Refers to the personal characteristics of the Product owner
Product backlog quality	Refers to how well the Product backlog represents everything that might be needed in the product in a way that maximizes the chances of project success
Progress tracking	Product owner tracks the project progress (for example, Product owner uses techniques such as Release Burndown)

Figura B.20: Vigésima pergunta do questionário

What is the influence of the following factors on the factor "Project success"?

	Very High Influence	High Influence	Moderate Influence	Low Influence	Very Low Influence
Potentially shippable product increment quality (end of each sprint)	○	○	○	○	○
Product Owner overall work quality	○	○	○	○	○
Work validation quality	○	○	○	○	○

Please, check the table below if you have any doubts.

Glossary	
Factor	Definition
Project success	Project reaches goals and satisfies stakeholders
Product Owner overall work quality	Product owner competence to fulfull his duties. The Product Owner is responsible for maximizing the value of the product and the work of Development team.
Potentially shippable product increment quality (end of each sprint)	Product increment delivered by the end of the sprint compliance with acceptance criterias
Work validation quality	Refers to activities that checks if the correct product is being developed

Figura B.21: Vigésima primeira pergunta do questionário

Apêndice C

Dados Coletados com o Questionário

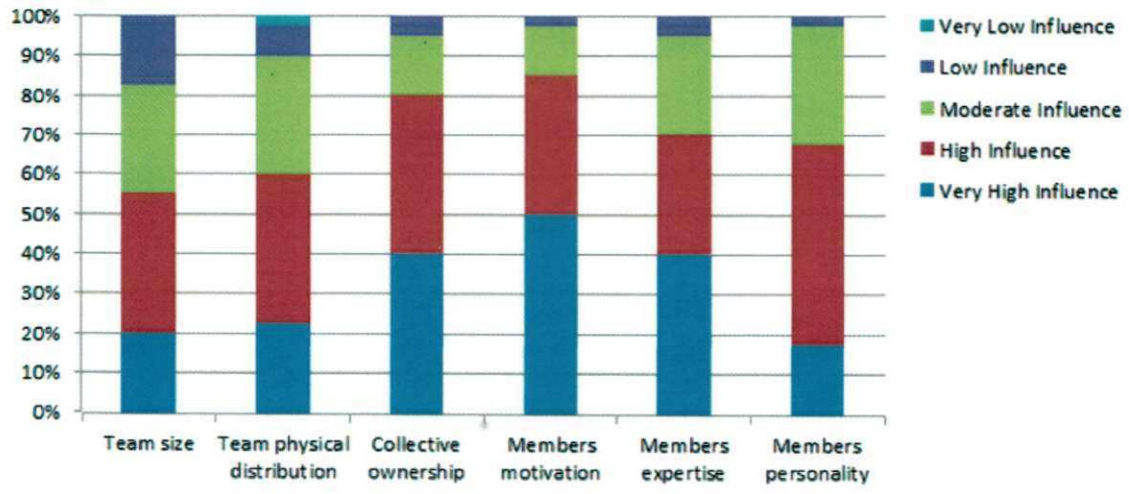

Figura C.1: Influências no fator *Development team teamwork skills*

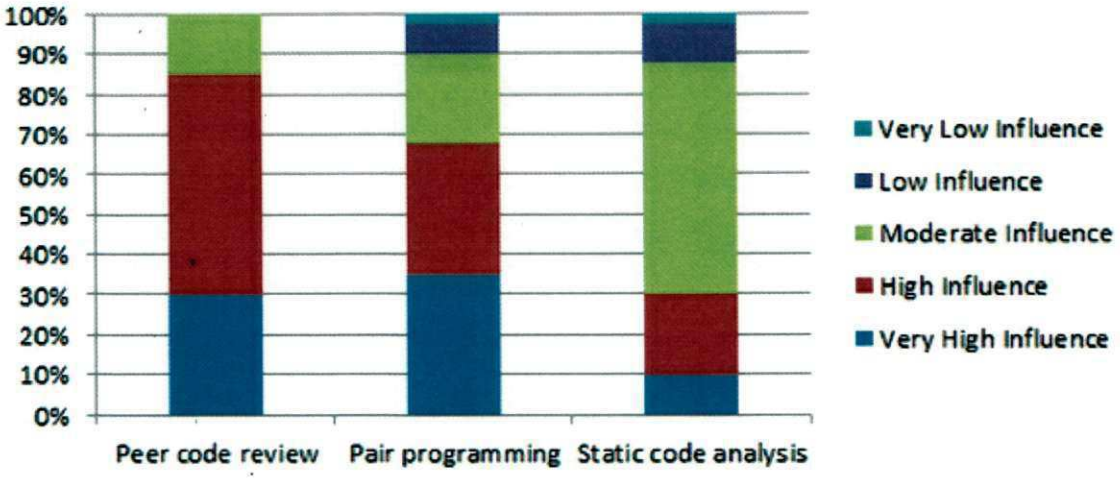

Figura C.2: Influências no fator *Code inspection quality*

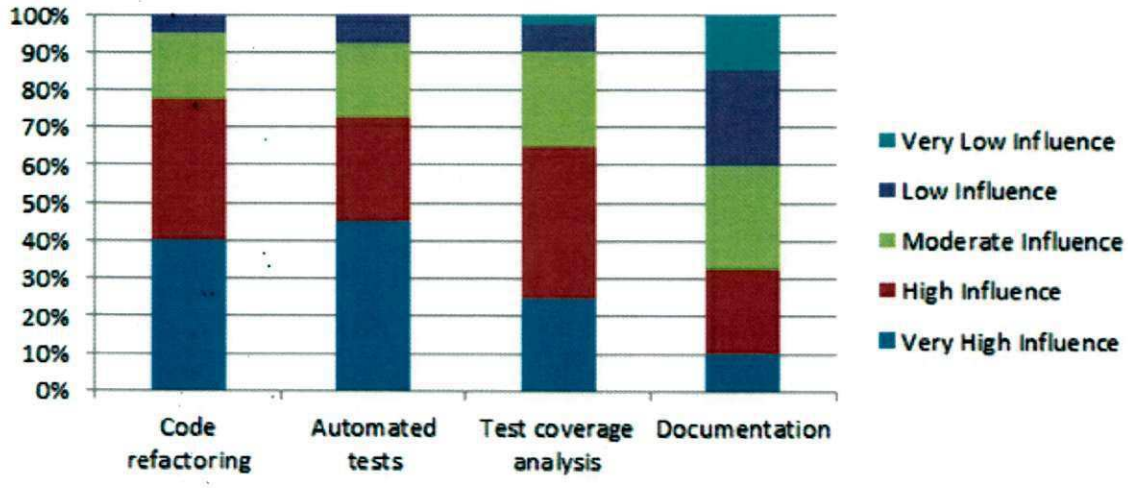

Figura C.3: Influências no fator *Code quality*

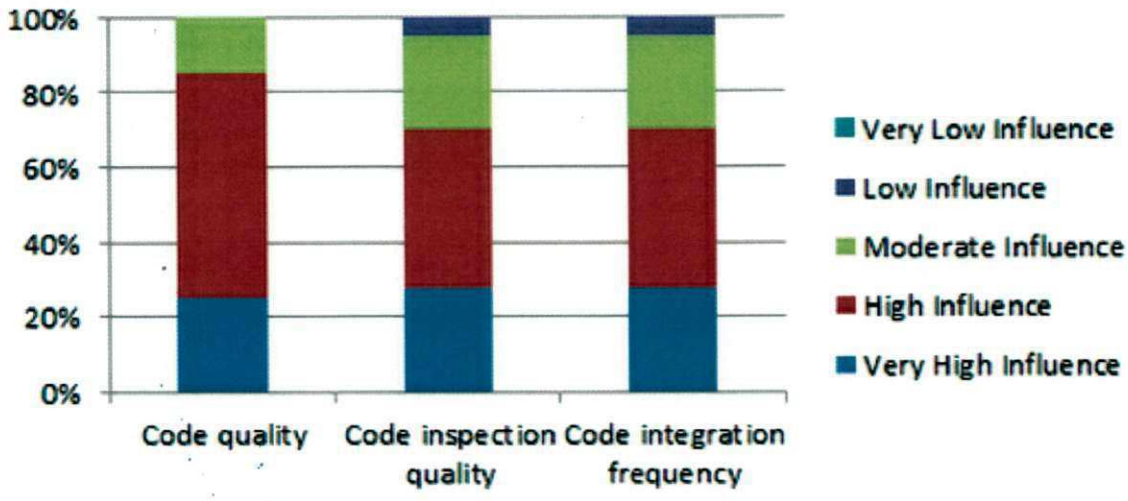

Figura C.4: Influências no fator *Software engineering techniques quality*

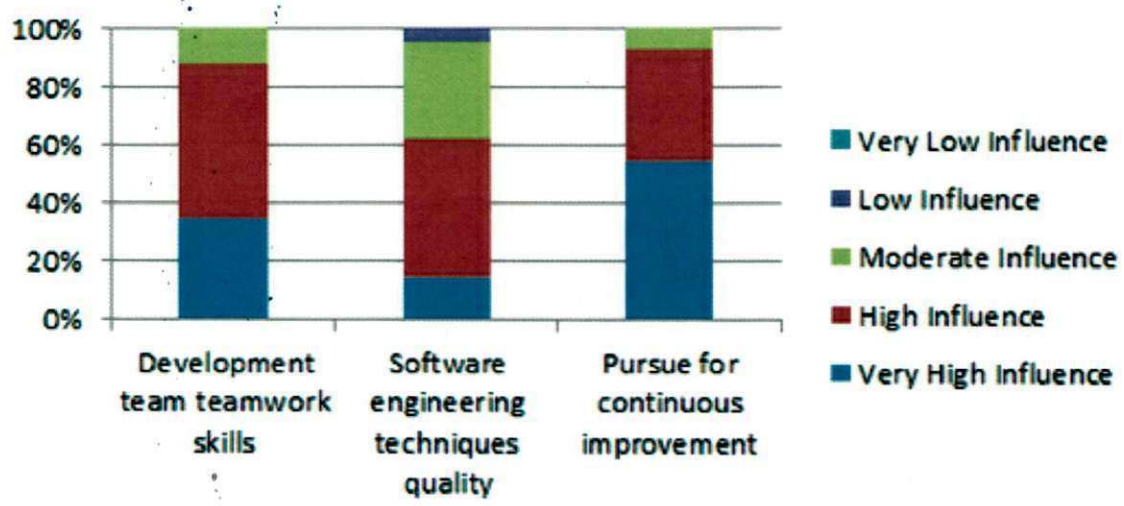

Figura C.5: Influências no fator *Development team competence*

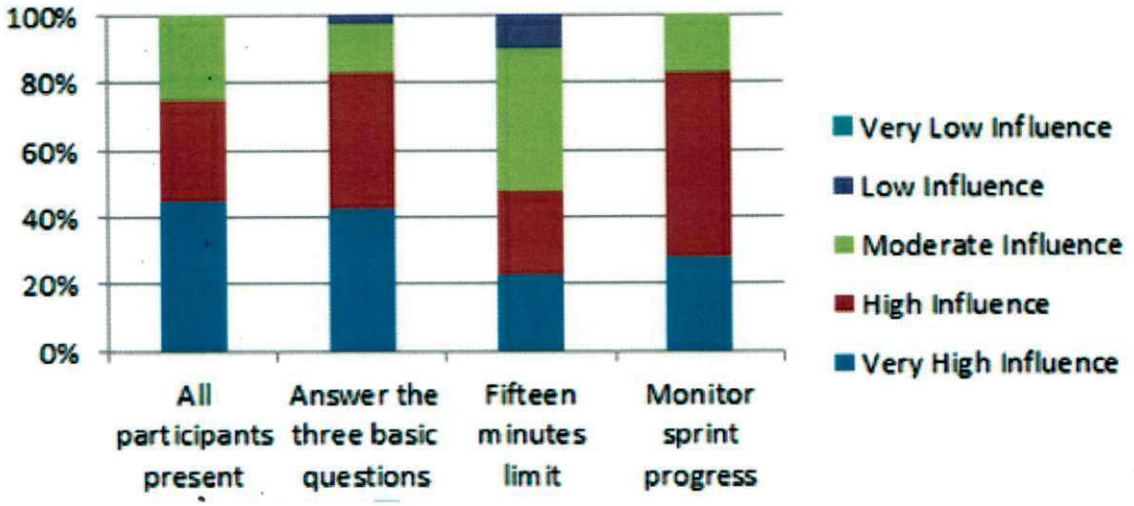

Figura C.6: Influências no fator *Daily Scrum quality*

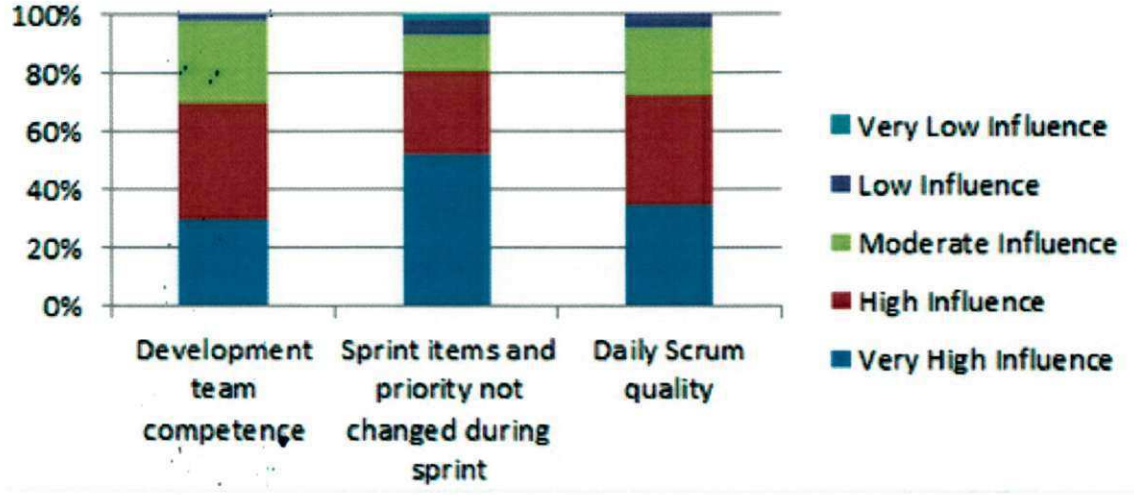

Figura C.7: Influências no fator *Sprint progress*

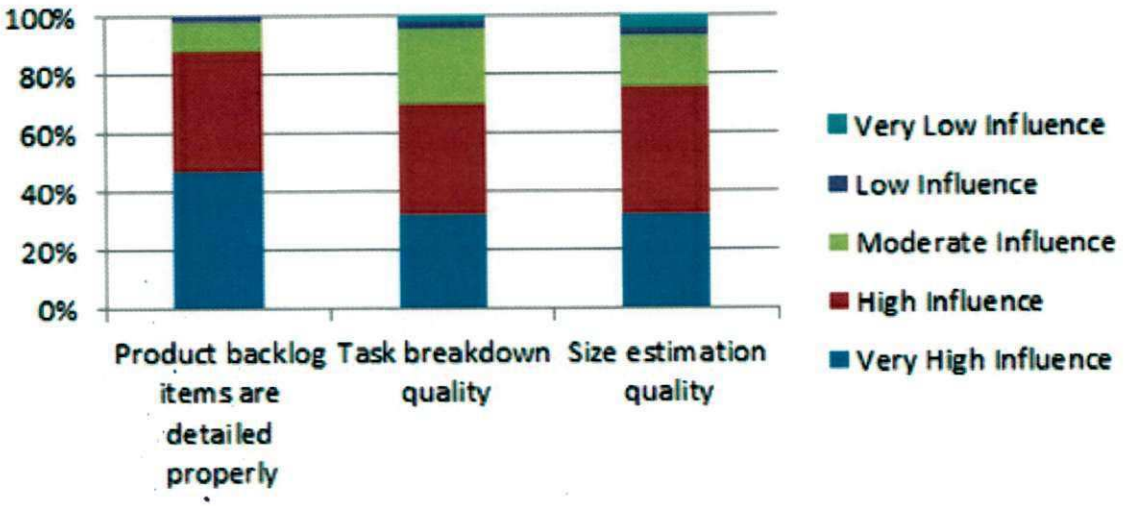

Figura C.8: Influências no fator *Sprint backlog quality*

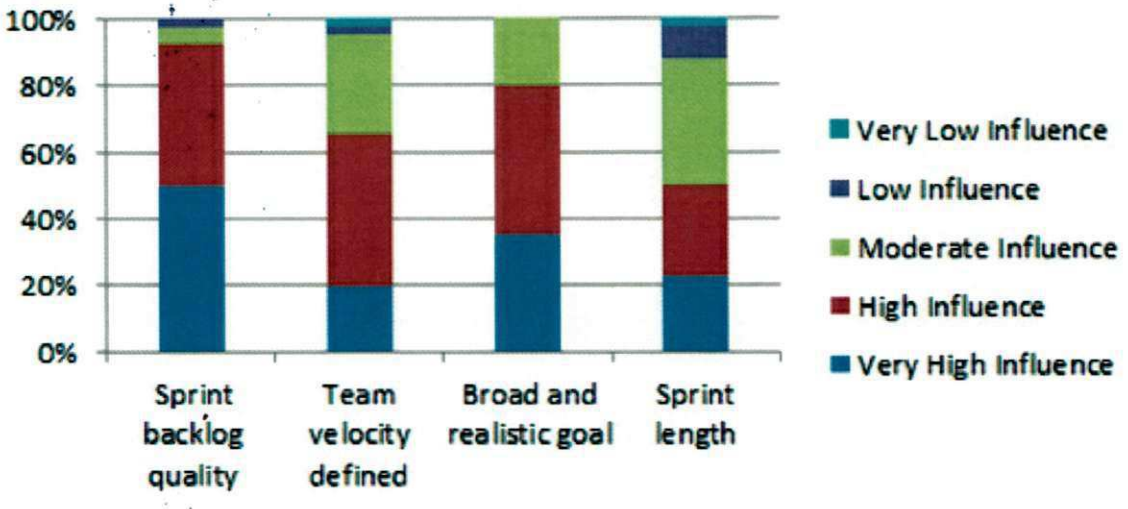

Figura C.9: Influências no fator *Sprint planning quality*

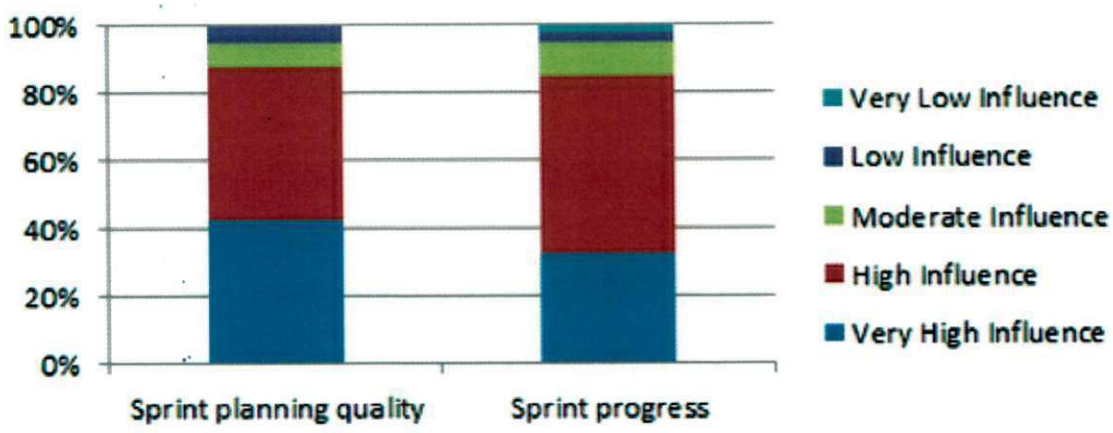

Figura C.10: Influências no fator *Potentially shippable product increment quality (end of each sprint)*

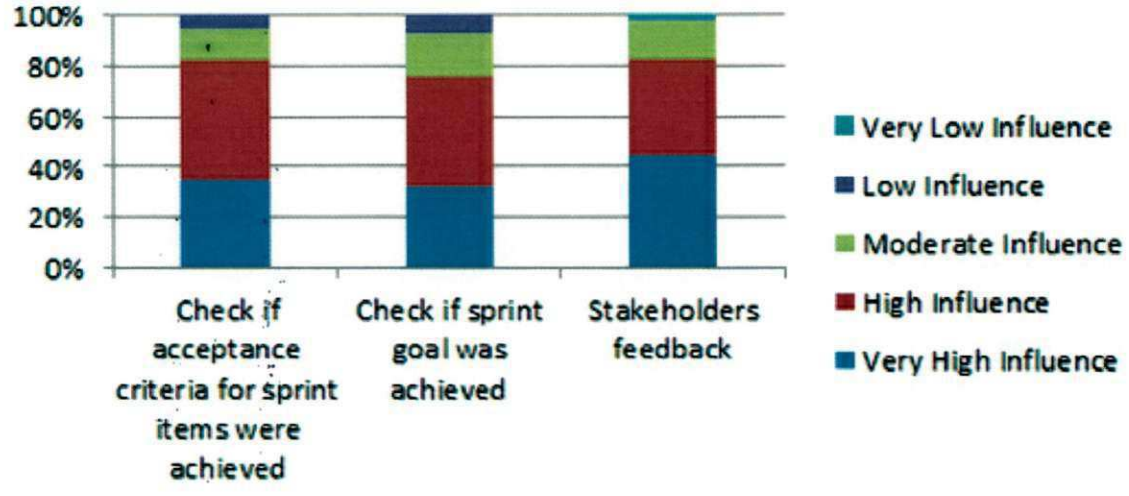

Figura C.11: Influências no fator *Sprint Review meeting achieving its goals*

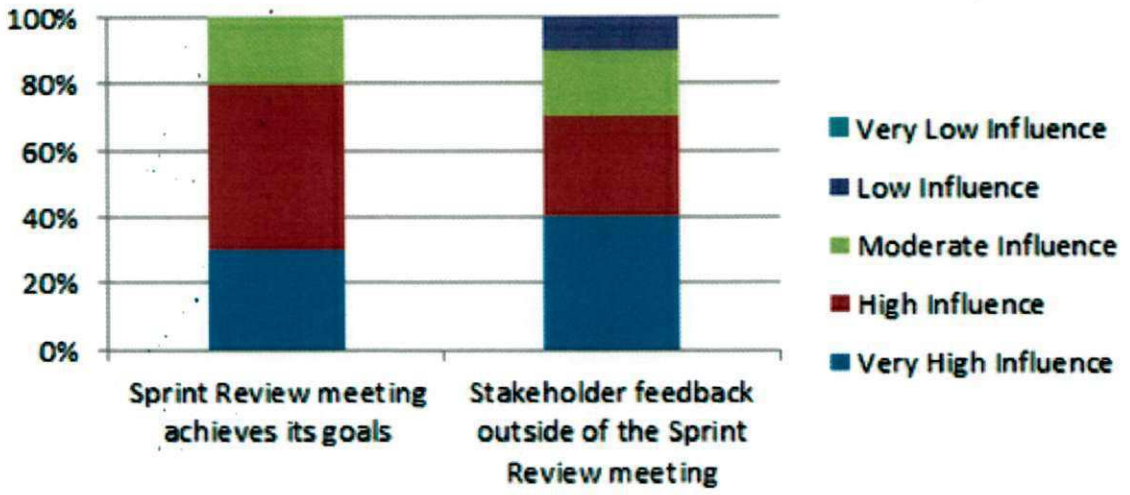

Figura C.12: Influências no fator *Work validation quality*

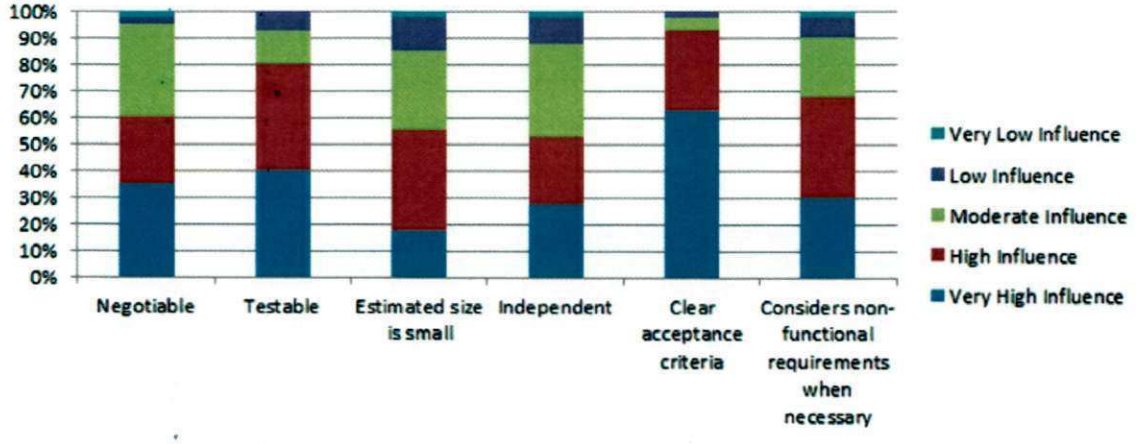

Figura C.13: Influências no fator *Product backlog item properly detailed*

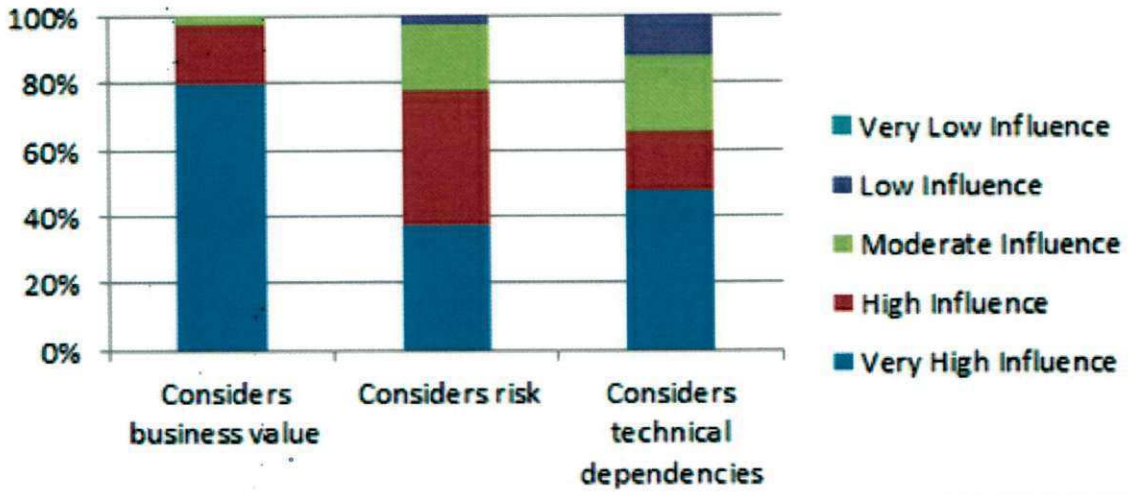

Figura C.14: Influências no fator *Product backlog properly ordered*

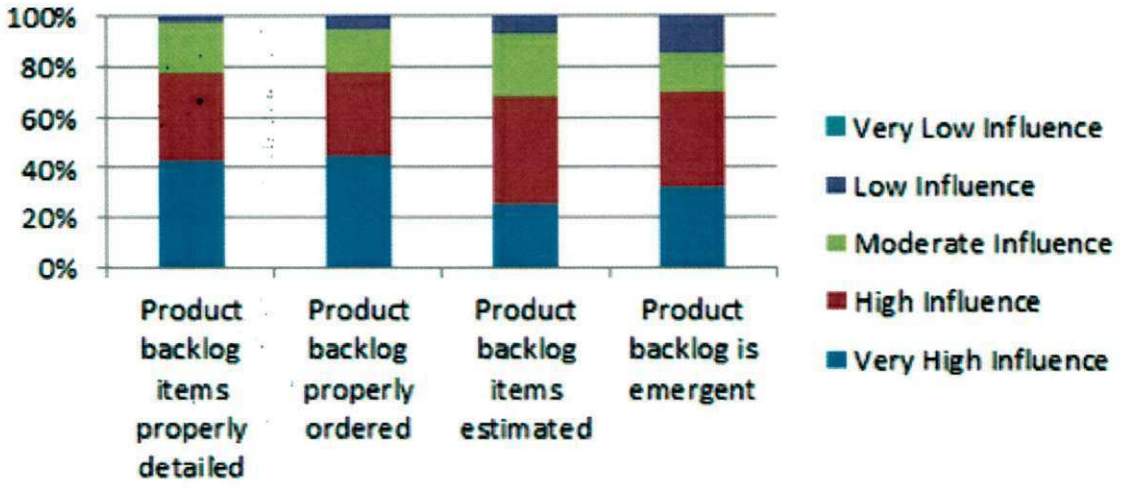

Figura C.15: Influências no fator *Product backlog management*

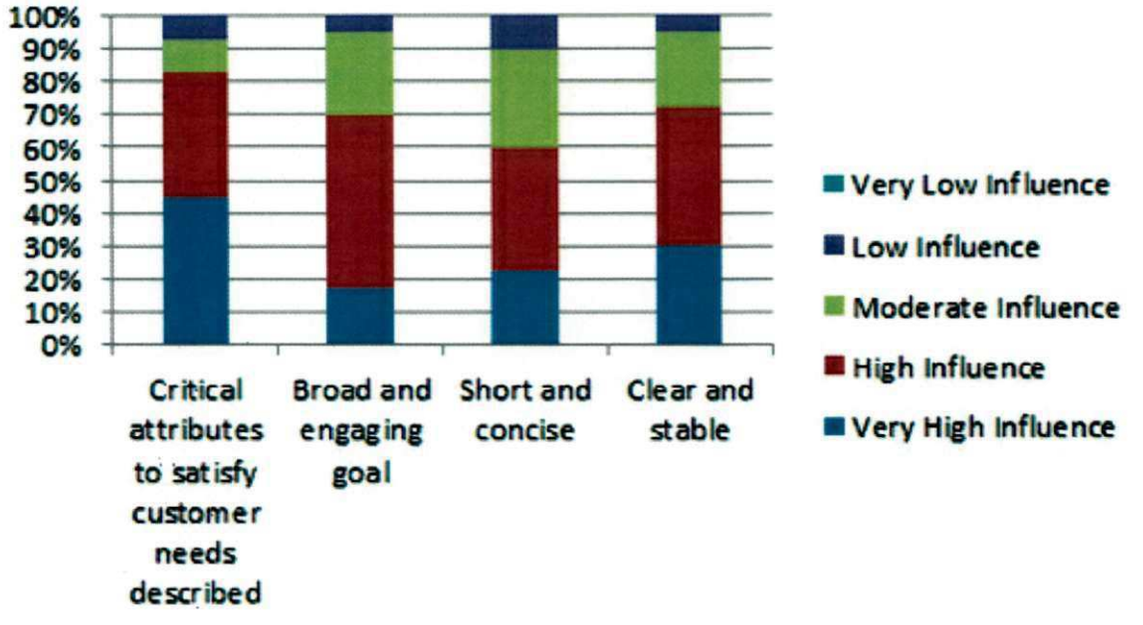

Figura C.16: Influências no fator *Product vision quality*

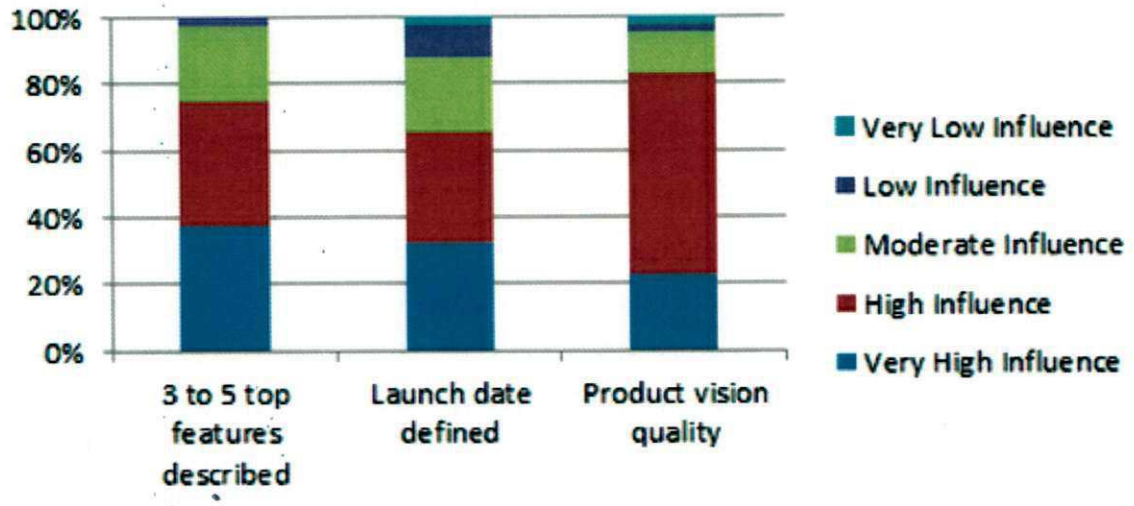

Figura C.17: Influências no fator *Release plan*

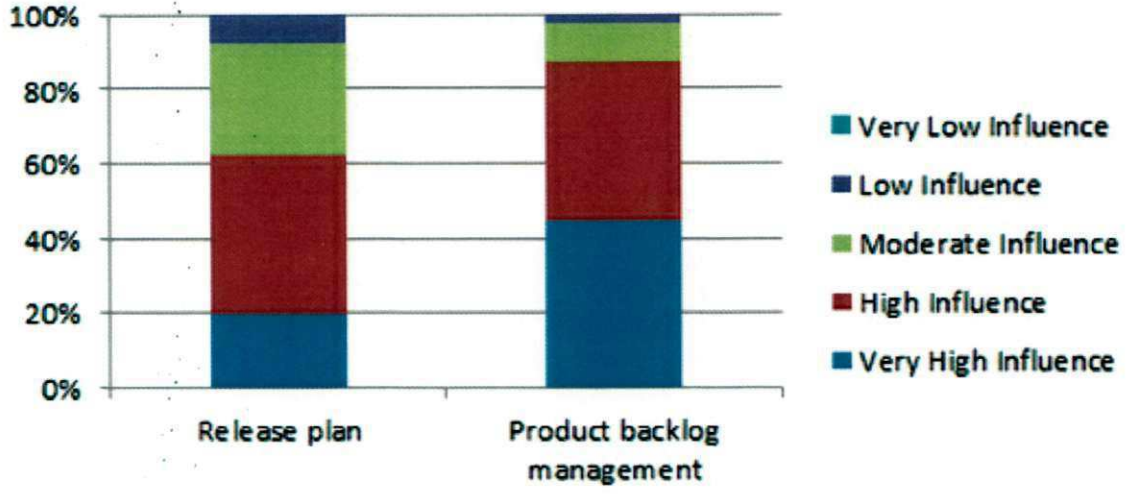

Figura C.18: Influências no fator *Product backlog quality*

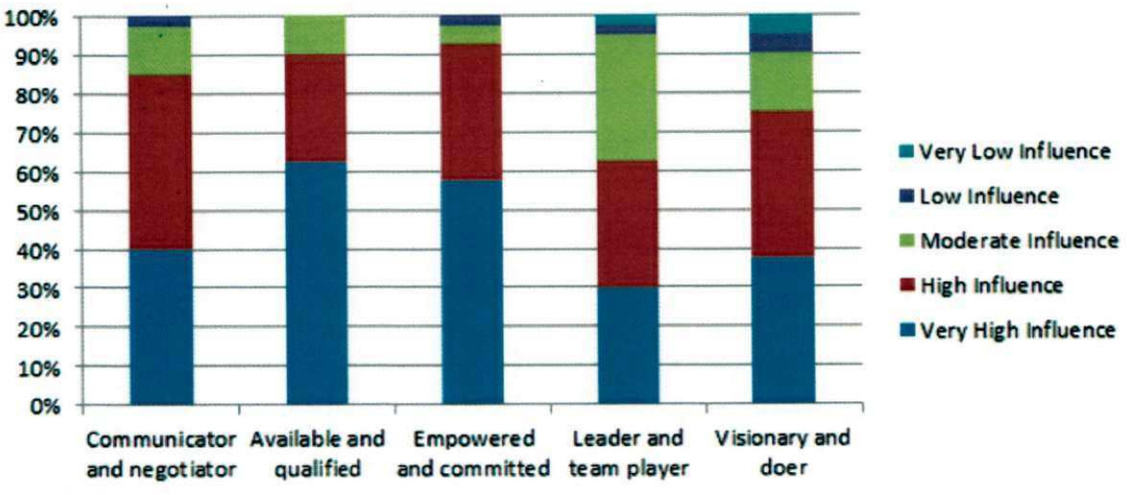

Figura C.19: Influências no fator *Desired personal characteristics of the Product Owner*

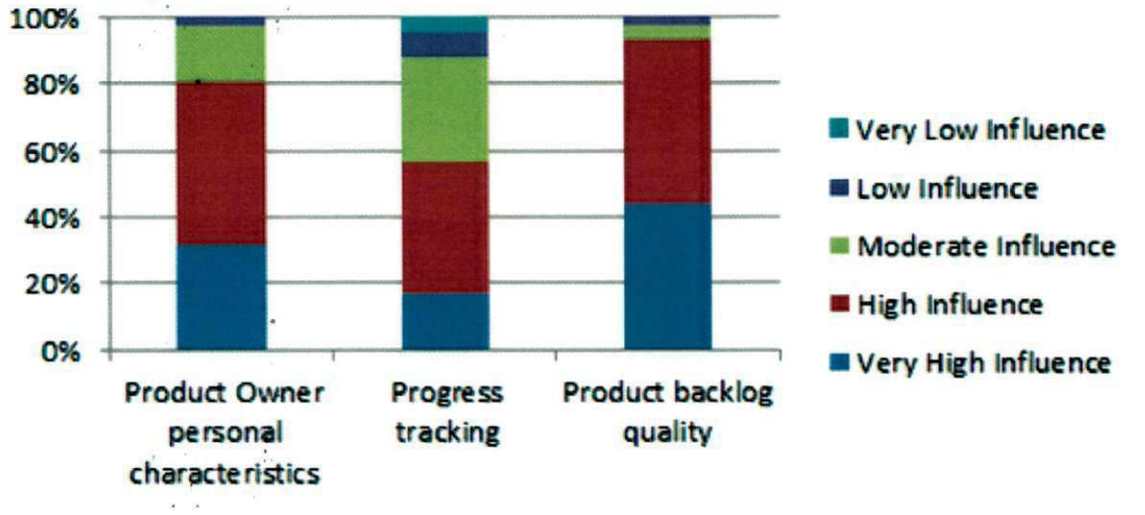

Figura C.20: Influências no fator *Product Owner overall work quality*

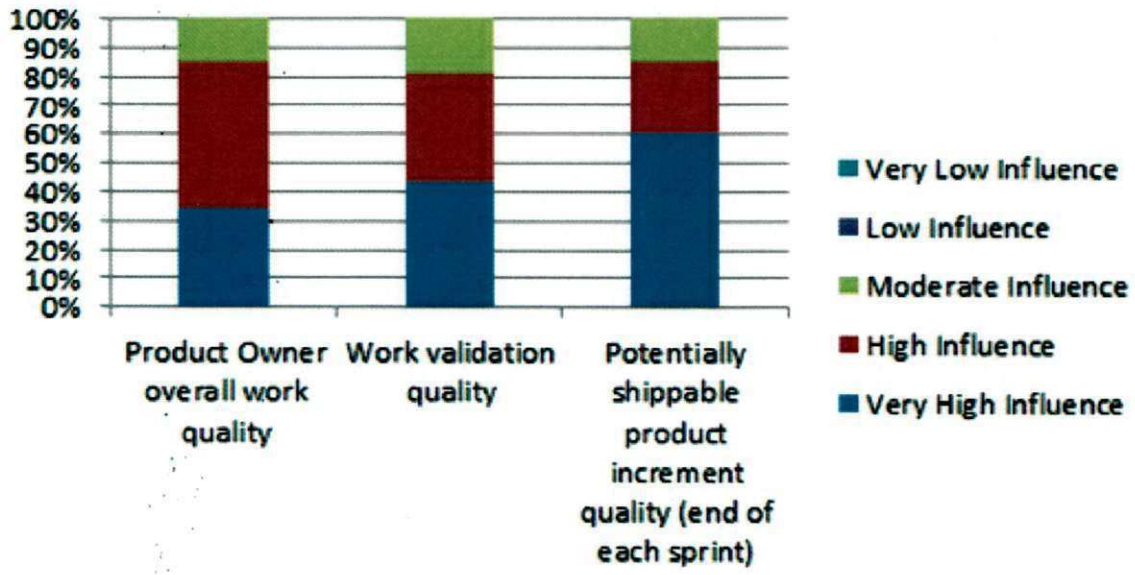

Figura C.21: Influências no fator *Project success*

Apêndice D

Resultados do Estudo Empírico

Neste capítulo, as hipóteses, testes de hipótese e conclusões do estudo empírico são apresentados. Para cada hipótese apresentada, onde tem *A - B*, lê-se *O rank médio do relacionamento entre o fator A e o fator B*. Todos os testes de hipótese foram executados usando o R [1].

- H1-0: *Team size - Development team teamwork skills = Team physical distribution - Development team teamwork skills = Collective ownership - Development team teamwork skills = Members motivation - Development team teamwork skills = Members expertise - Development team teamwork skills = Members personality - Development team teamwork skills*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (CO-DTTS, ME-DTTS, MM-DTTS, MP-DTTS,
                       TPD-DTTS, TS-DTTS)
         stratified by practitioner
```

[1] The R Project for Statistical Computing, *http://www.r-project.org/*

```
maxT = 3.8015, p-value = 0.001992

$PostHoc.Test

ME-DTTS - CO-DTTS  0.994375410
MM-DTTS - CO-DTTS  0.923092270
MP-DTTS - CO-DTTS  0.533679392
TPD-DTTS - CO-DTTS 0.141164969
TS-DTTS - CO-DTTS  0.054750232
MM-DTTS - ME-DTTS  0.646484346
MP-DTTS - ME-DTTS  0.858491734
TPD-DTTS - ME-DTTS 0.401547110
TS-DTTS - ME-DTTS  0.204585663
MP-DTTS - MM-DTTS  0.078905658
TPD-DTTS - MM-DTTS 0.008070088
TS-DTTS - MM-DTTS  0.001953002
TPD-DTTS - MP-DTTS 0.976008543
TS-DTTS - MP-DTTS  0.873259130
TS-DTTS - TPD-DTTS 0.999059446

Onde:
MM: Members motivation
TPD: Team physical distribution
MP: Members personality
TS: Team size
CO: Collective ownership
ME: Members expertise
DTTS: Development team teamwork skills
```

Conclusão: Apesar do resultado do teste de Friedman ter indicado que há diferença entre os ranks médios das amostras e o teste post-hoc ter indicado quais pares causaram o resultado do teste de Friedman, não pode-se afirmar que alguma das variáveis tem rank médio diferente de todas as outras com 90% de significância. Dessa forma, contrariar-se o teste de Friedman e afirma-se que não há diferença entre o rank médio das amostras. Ou seja, os fatores *Collective ownership*, *Team size*, *Team physical distribution*, *Members expertise*, *Members motivation* e *Members personality* influenciam *Development team teamwork skills* com a mesma intensidade.

- H2-0: *Peer code review - Code inspection quality = Pair programming - Code inspection quality = Static code analysis - Code inspection quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (PCR-CIQ, PP-CIQ, SCA-CIQ)
         stratified by practitioner
maxT = 4.7319, p-value = 6.064e-06

$PostHoc.Test

PP-CIQ - PCR-CIQ  0.2108112
SCA-CIQ - PCR-CIQ 6.029081e-06
SCA-CIQ - PP-CIQ  6.568024e-03

Onde:
PP: Pair programming
PCR: Peer Code Review
SCA: Static Code Analysis
```

```
CIQ: Code Inspection Quality
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que, como esperado ao analisar a Figura C.2, há diferença entre os ranks médios de *PP-CIQ* e *SCA-CIQ* e entre *PCR-CIQ* e *SCA-CIQ*. Além disso, o teste de post-hoc indica que não há diferença significativa entre *PCR-CIQ* e *PP-CIQ*. A partir da Figura D.1, pode-se afirmar com 90% de significância que os fatores *Pair programming* e *Peer code review* têm mais influência sobre *Code inspection quality* do que *Static code Analysis*.

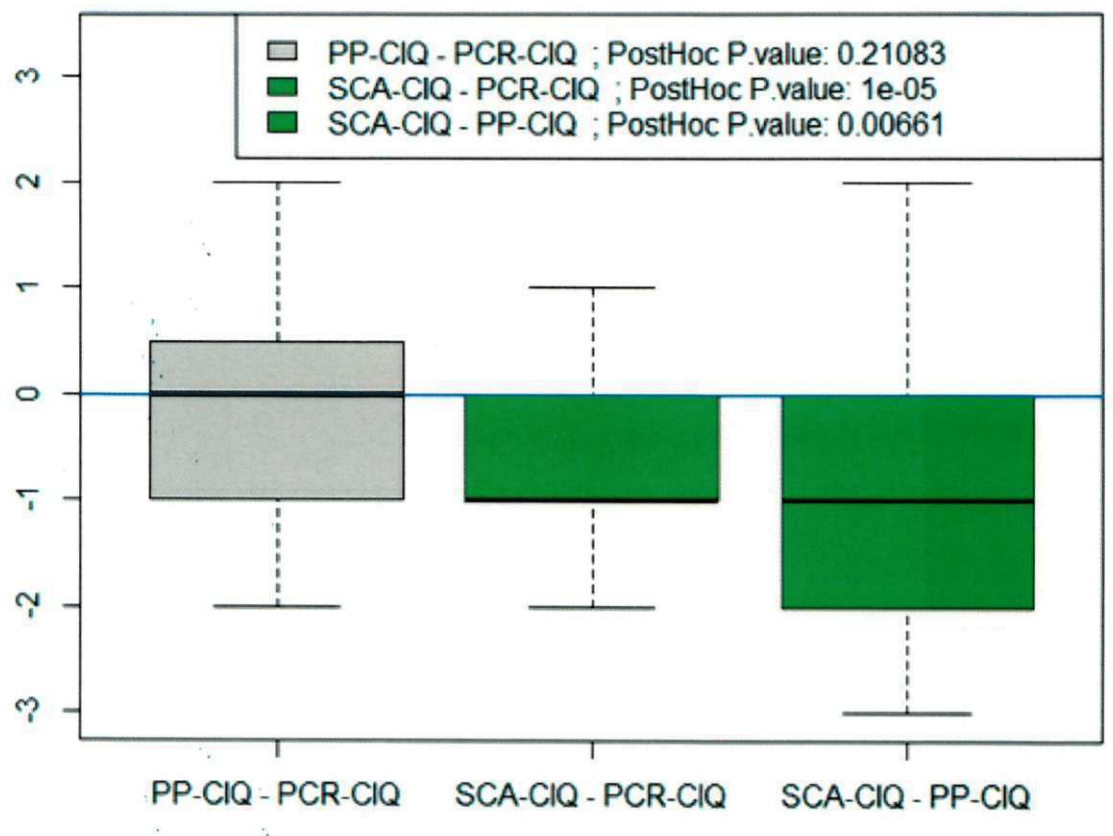

Figura D.1: Box plot das diferenças entre *PP-CIQ* e *PCR-CIQ*; *SCA-SIQ* e *PCR-SIQ*; e *SCA-SIQ* e *PP-CIQ*

- H3-0: *Code refactoring - Code quality = Automated tests - Code quality = Test coverage analysis - Code quality = Documentation - Code quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (AT-CQ, CR-CQ, D-CQ, TCA-CQ)
         stratified by practitioner
maxT = 4.5296, p-value = 3.514e-05

$PostHoc.Test

CR-CQ - AT-CQ  1.000000e+00
D-CQ - AT-CQ   4.239530e-05
TCA-CQ - AT-CQ 0.3569589e
D-CQ - CR-CQ   3.465601e-05
TCA-CQ - CR-CQ 0.3569581
TCA-CQ - D-CQ  0.01987526

Onde:
CR: Code refactoring
AT: Automated tests
TCA: Test coverage analysis
D: Documentation
CQ: Code quality
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que, como esperado ao analisar a Figura C.3, há diferença entre os ranks médios de *CR-CQ* e *D-CQ*, entre *AT-CQ* e *D-CQ*, e entre *TCA-CQ* e *D-CQ*. Além disso, o teste de post-hoc indica que não há diferença significativa entre os outros relacionamentos. A partir da Figura D.1, pode-se afirmar, com 90% de significância,

que os fatores *Test coverage analysis*, *Automated tests* e *Code refactoring* tem mais influência sobre *Code quality* do que *Documentation*.

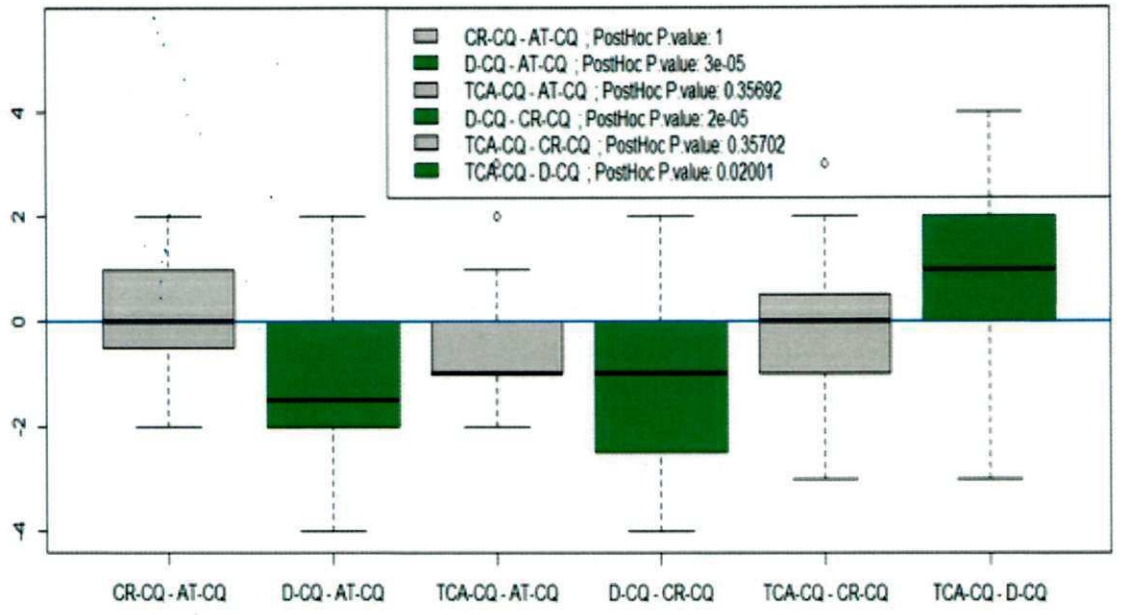

Figura D.2: Box plot das diferenças entre *CR-CQ* e *AT-CQ*; *D-CQ* e *AT-CQ*; *TCA-CQ* e *AT-CQ*;*D-CQ* e *CR-CQ*; *TCA-CQ* e *CR-CQ*; e *TCA-CQ* e *D-CQ*

- H4-0: *Code quality - Software engineering techniques quality = Code inspection quality - Software engineering techniques quality = Code integration frequency - Software engineering techniques quality*

```
$Friedman.Test

Asymptotic General Independence Test

data:  influenceLevel by
       relationship (CIF-SETQ, CIQ-SETQ, CQ-SETQ)
       stratified by practitioner
maxT = 1.2868, p-value = 0.4026
```

```
Onde:
CIF: Code integration frequency
CIQ: Code inspection quality
CQ: Code quality
SETQ: Software engineering techniques quality
```

Conclusão: O resultado do teste de Friedman indica que, com 90% de significância, não há diferença significativa entre os ranks médios das amostras. Dessa forma, pode-se afirmar, com 90% de significância, que os fatores *Code quality*, *Code integration frequency* e *Code inspection quality* influenciam *Software engineering techniques quality* com a mesma intensidade.

- H5-0: *Development team teamwork skills - Development team competence = Software engineering techniques quality - Development team competence = Pursue for continuous improvement - Development team competence*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (DTTS-DTC, PFCI-DTC, SETQ-DTC)
         stratified by practitioner
maxT = 5.1887, p-value = 6.635e-07

$PostHoc.Test

PFCI-DTC - DTTS-DTC 5.837272e-04
SETQ-DTC - DTTS-DTC 6.265966e-07
SETQ-DTC - PFCI-DTC 0.2993266
```

```
Onde:
PFCI: Pursue for continuous improvement
DTTS: Development team teamwork skills
SETQ: Software engineering techniques quality
DTC: Development team competence
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que há diferença entre os ranks médios de *PFCI-DTC* e *DTTS-DTC*, e entre *SETQ-DTC* e *DTTS-DTC*. Além disso, o teste de post-hoc indica que não há diferença significativa entre os ranks médios de *PFCI-DTC* e *SETQ-DTC*. A partir da Figura D.3, pode-se afirmar, com 90% de significância, que os fatores *Pursuit for continuous improvement* e *Software engineering techniques quality* tem mais influência sobre *Development team competence* do que *Development team teamwork skills*.

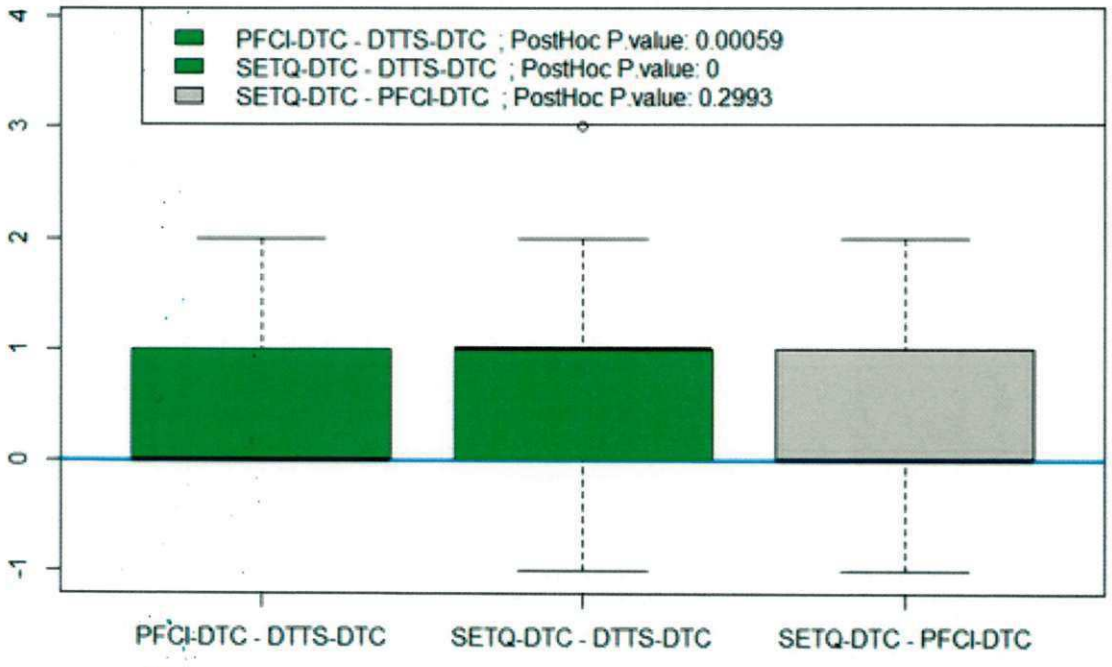

Figura D.3: Box plot das diferenças entre *PFCI-DTC* e *DTTS-DTC*; *SETQ-DTC* e *DTTS-DTC*; e *SETQ-DTC* e *FPCI-DTC*

- H6-0: *All participants present - Daily Scrum quality = Answer the three basic questions - Daily Scrum quality = Fifteen minutes limit - Daily Scrum quality = Monitor sprint progress - Daily Scrum quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (APP-DSQ, ATTBQ-DSQ, FML-DSQ, MSP-DSQ)
         stratified by practitioner
maxT = 3.4867, p-value = 0.002826

$PostHoc.Test

ATTBQ-DSQ - APP-DSQ 0.961108278
FML-DSQ - APP-DSQ   0.014588303
MSP-DSQ - APP-DSQ   0.919590278
FML-DSQ - ATTBQ-DSQ 0.002922137
MSP-DSQ - ATTBQ-DSQ 0.671274424
MSP-DSQ - FML-DSQ   0.085592418

Onde:
ATTBQ: Answer the three basic questions
FML: Fifteen minutes limit
MSP: Monitor sprint progress
APP: All participants present
DSQ: Daily Scrum quality
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que, como esperado ao analisar a Figura C.6, há diferença entre os ranks médios de *APP-DSQ* e *FML-DSQ*, entre *ATTBQ-DSQ* e *FML-DSQ*, e entre *MSP-DSQ* e *FML-DSQ*. Além disso, o teste de post-hoc indica que não há diferença significativa entre os outros relacionamentos. A partir da Figura D.4, pode-se afirmar, com 90% de significância, que os fatores *All participants present*, *Answer the three basic questions* e *Monitor sprint progress* tem mais influência sobre *Daily Scrum Quality* do que *Fifteen minutes limit*.

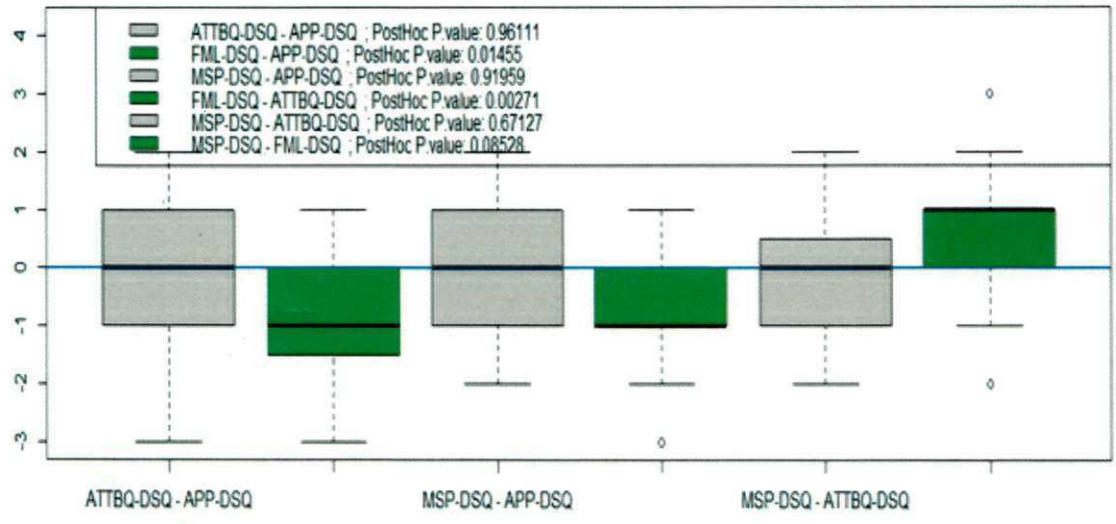

Figura D.4: Box plot das diferenças entre *ATTBQ-DSQ* e *APP-DSQ*; *FML-DSQ* e *APP-DSQ*; *MSP-DSQ* e *APP-DSQ*; *FML-DSQ* e *ATTBQ-DSQ*; *MSP-DSQ* e *ATTBQ-DSQ*; e *MSP-DSQ* e *FML-DSQ*

- H7-0: *Development team competence - Sprint progress = Sprint items and priority not changed during sprint - Sprint progress = Daily Scrum quality - Sprint progress*

```
$Friedman.Test

        Asymptotic General Independence Test
```

```
data:  influenceLevel by
         relationship (DSQ-SP, DTC-SP, SIAPSNCDS-SP)
         stratified by practitioner
maxT = 1.3074, p-value = 0.391

Onde:
DSQ: Daily Scrum quality
DTC: Development team competence
SIAPSNCDS: Sprint items and priority not changed during
           sprint
SP: Sprint progress
```

Conclusão: O resultado do teste de Friedman indica que, com 90% de significância, não há diferença significativa entre os ranks médios das amostras. Dessa forma, pode-se afirmar, com 90% de significância, que os fatores *Daily Scrum quality*, *Sprint items and priority not changed during Sprint* e *Development team competence* influenciam *Sprint progress* com a mesma intensidade.

- H8-0: *Product backlog items are detailed properly - Sprint backlog quality = Task breakdown quality - Sprint backlog quality = Size estimation quality - Sprint backlog quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (PBIADP-SBQ, SEQ-SBQ, TBQ-SBQ)
         stratified by practitioner
maxT = 2.1736, p-value = 0.07568
```

```
$PostHoc.Test

SEQ-SBQ - PBIADP-SBQ 0.07571234
TBQ-SBQ - PBIADP-SBQ 0.09018819
TBQ-SBQ - SEQ-SBQ    0.99690740

Onde:
SEQ: Size estimation quality
    TBQ: Task breakdown quality
    PBIADP: Product backlog items are detailed properly
    SBQ: Sprint backlog quality
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que há diferença entre os ranks médios de *SEQ-SBQ* e *PBIADP-SBQ* e entre *TBQ-SBQ* e *PBIADP-SBQ*. Além disso, o teste de post-hoc indica que não há diferença significativa entre *SEQ-SBQ* e *TBQ-SBQ*. A partir da Figura D.5, pode-se afirmar com 90% de significância que os fatores *Size estimation quality* e *Task breakdown quality* tem menos influência sobre *Sprint backlog quality* do que *Product backlog items are properly detailed.*

- H9-0: *Sprint backlog quality - Sprint planning quality = Team velocity defined - Sprint planning quality = Broad and realistic goal- Sprint planning quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (BARG-SPQ, SBQ-SPQ, SL-SPQ, TVD-SPQ)
```

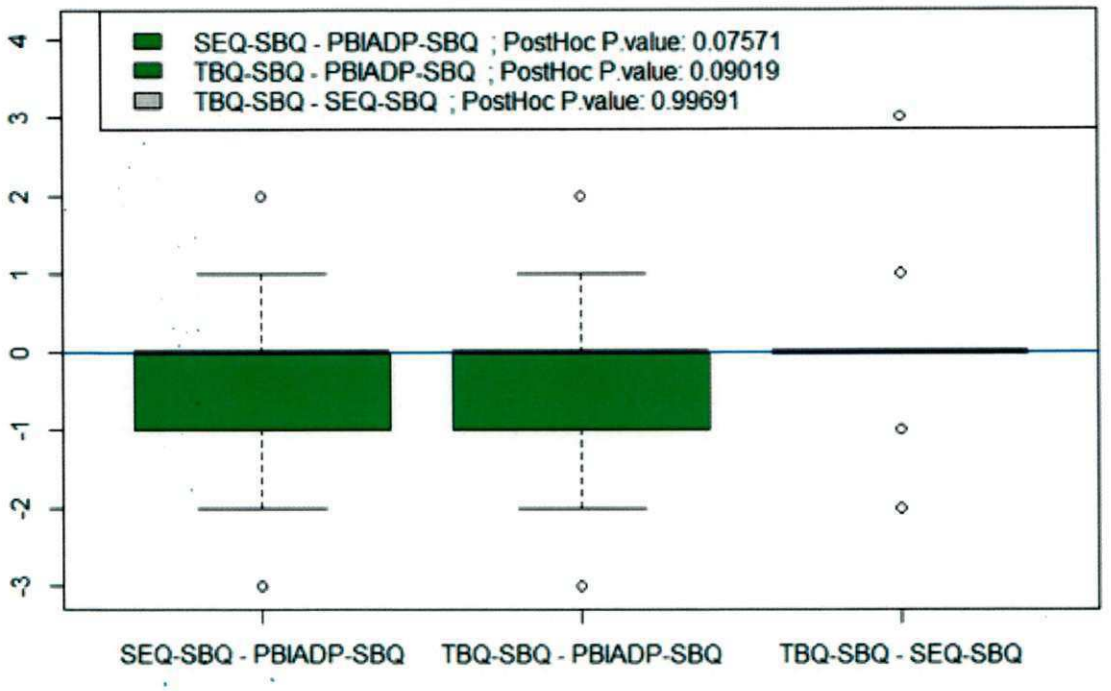

Figura D.5: Box plot das diferenças entre *SEQ-SBQ* e *PBIADP-SBQ*; *TBQ-SBQ* e *PBIADP-SBQ*; e *TBQ-SBQ* e *SEQ-SBQ*

```
        stratified by practitioner
maxT = 4.1122, p-value = 0.0001987

$PostHoc.Test

SBQ-SPQ - BARG-SPQ 0.7404521637
SL-SPQ - BARG-SPQ  0.0103935061
TVD-SPQ - BARG-SPQ 0.1956716192
SL-SPQ - SBQ-SPQ   0.0002120126
TVD-SPQ - SBQ-SPQ  0.0146231410
TVD-SPQ - SL-SPQ   0.6791053954

Onde:
SBQ: Sprint backlog quality
```

```
BARG: Broad and realistic goal
SL: Sprint length
TVD: Team velocity defined
SPQ: Sprint planning quality
```

Conclusão: Apesar do resultado do teste de Friedman ter indicado que há diferença entre os ranks médios das amostras e o teste post-hoc ter indicado quais pares causaram o resultado do teste de Friedman, não pode-se afirmar que alguma das variáveis tem rank médio diferente de todas as outras com 90% de significância. Dessa forma, contraria-se o teste de Friedman e afirma-se que não há diferença entre o rank médio das amostras. Ou seja, os fatores *Broad and realistic goal*, *Team velocity defined*, *Sprint backlog quality* e *Sprint length* influenciam Sprint planning quality com a mesma intensidade.

- H10-0: *Sprint planning quality - Potentially shippable product increment quality (end of each sprint) = Sprint progress - Potentially shippable product increment quality (end of each sprint)*

```
Wilcoxon signed rank test with continuity correction

data:  questao10$SP and questao10$SPQ
V = 40.5, p-value = 0.4509
alternative hypothesis: true location shift is not
                        equal to 0

Onde:
questao10$SP: SP-PSPIQ
questao10$SPQ: SPQ-PSPIQ
    SP: Sprint planning quality
SP: Sprint progress
    PSPIQ: Potentially shippable product increment quality
```

```
        (end of each sprint)
```

Conclusão: O resultado do teste de Wilcoxon indica que não há diferença significativa entre a distribuição das amostras. Dessa forma, podemos afirmar com 90% de significância que os fatores *Sprint progress* e *Sprint planning quality* influenciam o fator *Potentially shippable product increment quality (end of each sprint)* com a mesma intensidade.

- H11-0: *Check if acceptance criteria for sprint items were achieved - Sprint Review meeting achieves its goals = Check if sprint goal was achieved - Sprint Review meeting achieves its goals = stakeholders feedback - Sprint Review meeting achieves its goals*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (CIACFSIWA-SRMAIG, CISGWA-SRMAIG,
         SF-SRMAIG)
         stratified by practitioner
maxT = 1.3342, p-value = 0.3761

Onde:
CIACFSIWA: Check if acceptance criteria for sprint
           items were achieved
    SRMAIG: Sprint Review meeting achieves its goals
CISGWA: Check if sprint goal was achieved
    SF: \Textit{stakeholders} feedback
```

Conclusão: O resultado do teste de Friedman indica que, com 90% de significância, não há diferença significativa entre os ranks médios das amostras. Dessa forma, pode-se afirmar,

com 90% de significância, que os fatores *stakeholders feedback*, *Check if acceptance criteria for sprint items were achieved* e *Check if sprint goal was achieved* influenciam *Sprint Review meeting achieves its goals* com a mesma intensidade.

- H12-0: *Sprint Review meeting achieves its goals - Work validation quality = Stakeholder feedback outside of the Sprint Review meeting - Work validation quality*

```
Wilcoxon signed rank test with continuity correction

data:  questao12$SRMAIG and questao12$SFOOTSRM
V = 197, p-value = 0.5644
alternative hypothesis: true location shift is not
                        equal to 0

Onde:
questao12$SRMAIG: SRMAIG-WVQ
questao12$SFOOTSRM: SFOOTSRM-WVQ
    SRMAIG: Sprint Review meeting achieves its goals
SFOOTSRM: Stakeholder feedback outside of the Sprint
          Review meeting
    WVQ: Work validation quality
```

Conclusão: O resultado do teste de Wilcoxon indica que não há diferença significativa entre a distribuição das amostras. Dessa forma, podemos afirmar com 90% de significância que os fatores *Sprint Review meeting achieves its goals* e *Stakeholder feedback outside of the Sprint Review meeting* influenciam o fator *Work validation quality* com a mesma intensidade.

- H13-0: *Negotiable - Product backlog item properly detailed = Testable - Product backlog item properly detailed = Estimated size is small - Product backlog item properly detailed = Independent - Product backlog item properly detailed = Clear acceptance criteria - Product backlog item properly detailed = Considers non-functional requirements when necessary - Product backlog item properly detailed*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
        relationship (CAC-PBIPD, CNFRWN-PBIPB, ESIS-PBIPD,
         I-PBIPD, N-PBIPD, T-PBIPD)
         stratified by practitioner
maxT = 5.2904, p-value = 2.085e-06

$PostHoc.Test

CNFRWN-PBIPB - CAC-PBIPD  6.072561e-03
ESIS-PBIPD - CAC-PBIPD    1.152901e-06
I-PBIPD - CAC-PBIPD       7.716114e-05
N-PBIPD - CAC-PBIPD       3.692082e-03
T-PBIPD - CAC-PBIPD       0.4499492
ESIS-PBIPD - CNFRWN-PBIPB 0.4736166
I-PBIPD - CNFRWN-PBIPB    0.8978709
N-PBIPD - CNFRWN-PBIPB    0.9999908
T-PBIPD - CNFRWN-PBIPB    .5461598
I-PBIPD - ESIS-PBIPD      0.9783354
N-PBIPD - ESIS-PBIPD      0.5705540
T-PBIPD - ESIS-PBIPD      7.099311e-03
N-PBIPD - I-PBIPD         0.9435625
T-PBIPD - I-PBIPD         6.835781e-02
T-PBIPD - N-PBIPD         .04498918

Onde:
CNFRWN: Considers non-functional requirements when necessary
```

```
CAC: Clear acceptance criteria
ESIS: Estimated size is small
I: Independent
N: Negotiable
T: Testable
PBIPB: Product backlog item properly detailed
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que há diferença entre os ranks médios de *CAC-PBIPD* e todos os outros, entre *T-PBIPD* e *I-PBIPD*, e entre *T-PBIPD* e *ESIS-PBIPD*. Além disso, o teste de post-hoc indica que não há diferença significativa entre os outros relacionamentos. Dado os resultados do teste de post-hoc, pode-se concluir que *CAC-PBIPD* tem rank médio diferente das outras amostras. Por outro lado, não podemos concluir que há diferença entre os ranks médios das outras amostras. Dado isso e a partir da Figura D.6, pode-se afirmar, com 90% de significância, que os fatores *Negotiable*, *Independent*, *Considers non-functional requirements when necessary*, *Estimated size is small* e *Testable* tem menos influência sobre *Product backlog item properly detailed* do que *Clear acceptance criteria.*

- H14-0: *Considers business value - Product backlog properly ordered = Considers risk - Product backlog properly ordered = Considers technical dependencies - Product backlog properly ordered*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (CBV-PBPO, CR-PBPO, CTD-PBPO)
         stratified by practitioner
```

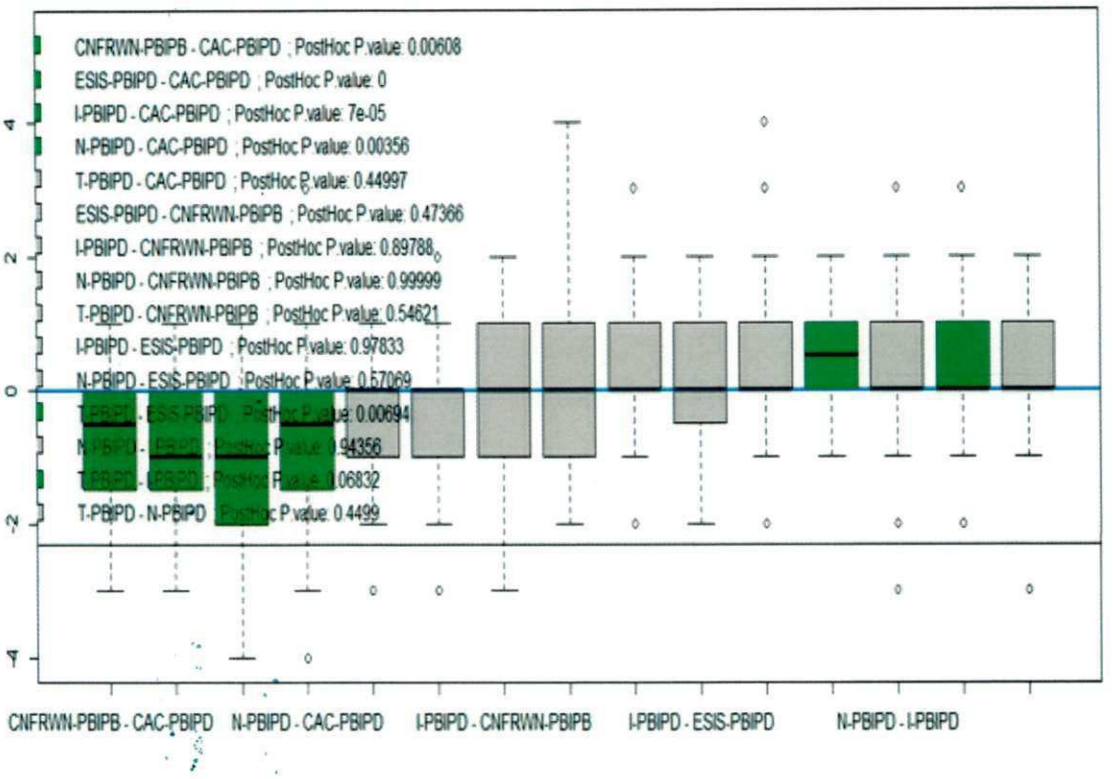

Figura D.6: Box plot das diferenças entre *CNFRWN-PBIPB* e *CAC-PBIPB*; *ESIS-PBIPB* e *CAC-PBIPB*; *I-PBIPB* e *CAC-PBIPB*; *N-PBIPB* e *CAC-PBIPB*; *T-PBIPB* e *CAC-PBIPB*; *ESIS-PBIPB* e *CNFRWN-PBIPB*; *I-PBIPB* e *CNFRWN-PBIPB*; *N-PBIPB* e *CNFRWN-PBIPB*; *T-PBIPB* e *CNFRWN-PBIPB*; *I-PBIPB* e *ESIS-PBIPB*; *N-PBIPB* e *ESIS-PBIPB*; *T-PBIPB* e *ESIS-PBIPB*; *N-PBIPB* e *I-PBIPB*; *T-PBIPB* e *I-PBIPB*; e *T-PBIPB* e *N-SBQ*

```
maxT = 3.6999, p-value = 0.0006018

$PostHoc.Test

CR-PBPO - CBV-PBPO  0.000641889
CTD-PBPO - CBV-PBPO 0.001441067
CTD-PBPO - CR-PBPO  0.974226359

Onde:
CR: Considers risk
```

```
CTD: Considers technical dependencies
CBV: Considers business value
PBPO: Product backlog properly ordered
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que, como esperado ao analisar a Figura ref:d14, há diferença entre os ranks médios de *CR-PBPO* e *CBV-PBPO* e entre *CTD-PBPO* e *CBV-PBPO*. Além disso, o teste de post-hoc indica que não há diferença significativa entre *CR-PBPO* e *CTD-PBPO*. A partir da Figura D.7, pode-se afirmar com 90% de significância que os fatores *Considers risk* e *Considers technical dependencies* tem menos influência sobre *Product backlog properly ordered* do que *Considers business value*.

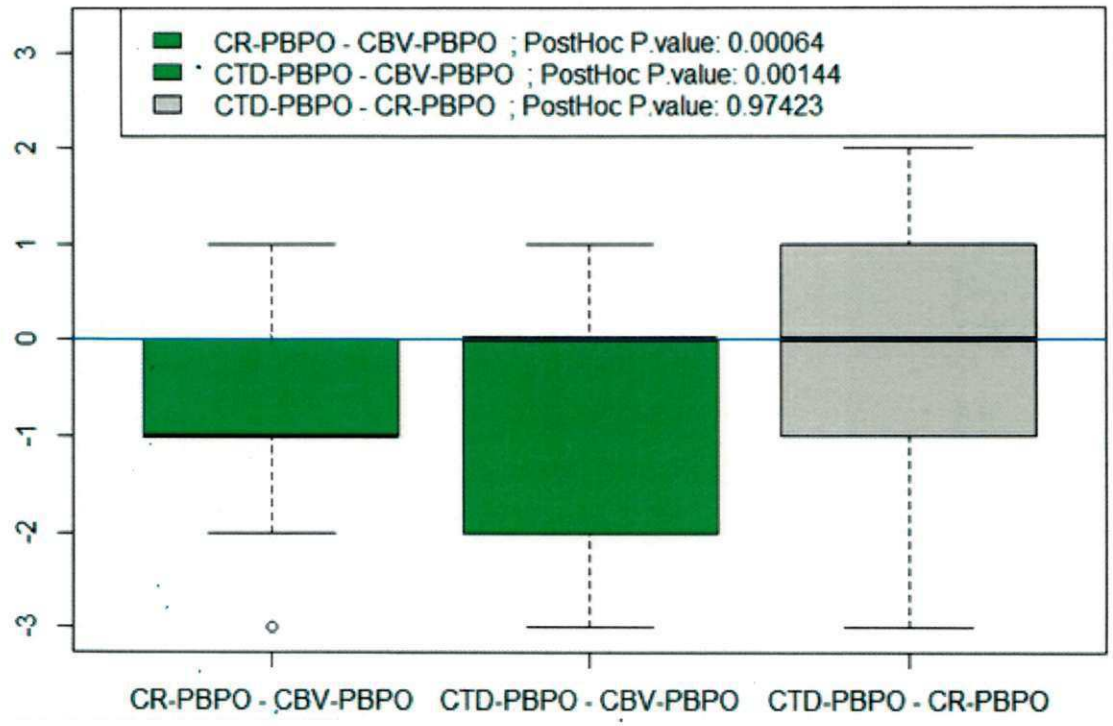

Figura D.7: Box plot das diferenças entre *CR-PBPO* e *CBV-PBPO*; *CTD-PBPO* e *CBV-PBPO*; e *CTD-PBPO* e *CR-PBPO*

- H15-0: *Product backlog items properly detailed - Product backlog management = Product backlog properly ordered - Product backlog management = Product backlog*

items estimated - Product backlog management = Product backlog is emergent - Product backlog management

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (PBIE-PBM, PBIEM-PBM, PBIPD-PBM,
         PBPO-PBM)
         stratified by practitioner
maxT = 2.1068, p-value = 0.1509

Onde:
PBIE: Product backlog items estimated
PBIEM: Product backlog is emergent
PBIPD: Product backlog items properly detailed
PBPO: Product backlog properly ordered
    PBM: Product backlog management
```

Conclusão: O resultado do teste de Friedman indica que, com 90% de significância, não há diferença significativa entre os ranks médios das amostras. Dessa forma, pode-se afirmar, com 90% de significância, que os fatores *Product backlog is estimated*, *Product backlog is emergent*, *Product backlog properly ordered* e *Product backlog items properly detailed* influenciam *Product backlog management* com a mesma intensidade.

- H16-0: *Critical attributes to satisfy customer needs described - Product vision quality = Broad and engaging goal - Product vision quality = Short and concise - Product vision quality = Clear and stable - Product vision quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (BAEG-PVQ, CAS-PVQ, CATSCND-PVQ,
         SAC-PVQ)
         stratified by practitioner
maxT = 2.9858, p-value = 0.01491

$PostHoc.Test

CAS-PVQ - BAEG-PVQ     0.79057407
CATSCND-PVQ - BAEG-PVQ 0.03760008
SAC-PVQ - BAEG-PVQ     0.98978609
CATSCND-PVQ - CAS-PVQ  0.29774175
SAC-PVQ - CAS-PVQ      0.60417433
SAC-PVQ - CATSCND-PVQ  0.01501345

Onde:
CAS: Short and concise
BAEG: Broad and engaging goal
CATSCND: Critical attributes to satisfy customer needs
         described
SAC: Short and concise
PVQ: Product vision quality
```

Conclusão: Apesar do resultado do teste de Friedman ter indicado que há diferença entre os ranks médios das amostras e o teste post-hoc ter indicado quais pares causaram o resultado do teste de Friedman, não pode-se afirmar que alguma das variáveis tem rank médio diferente

de todas as outras com 90% de significância. Dessa forma, contraria-se o teste de Friedman e afirma-se que não há diferença entre o rank médio das amostras. Ou seja, os fatores *Clear and stable*, *Critical attributes to satisfy customer needs described*, *Broad and engaging goal*, e *Short and concise* influenciam *Product vision quality* com a mesma intensidade.

- H17-0: *3 to 5 top features described - Release plan = Launch date defined - Release plan = Product vision quality - Release plan*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (LDD-RP, PVQ-RP, TFD-RP)
         stratified by practitioner
maxT = 1.2421, p-value = 0.4283

Onde:
LDD: Launch date defined
PVQ: Product vision quality
TFD: 3 to 5 top features described
RP: Release plan
```

Conclusão: O resultado do teste de Friedman indica que, com 90% de significância, não há diferença significativa entre os ranks médios das amostras. Dessa forma, pode-se afirmar, com 90% de significância, que os fatores *Launch date defined*, *Product vision quality* e *3 to 5 top features described* influenciam *Release plan* com a mesma intensidade.

- H18-0: *Release plan - Product backlog quality = Product backlog management- Product backlog quality*

```
Wilcoxon signed rank test with continuity correction

data:  questao18$RL and questao18$PBM
V = 26, p-value = 0.001202
alternative hypothesis: true location shift is not
                        equal to 0

--------------------

      Wilcoxon signed rank test with continuity correction

data:  questao18$RL and questao18$PBM
V = 26, p-value = 0.9995
alternative hypothesis: true location shift is greater
                        than 0

Onde:

questao18$RL: RL-PBQ
questao18$PBM: PBM-PBQ
RL - Release plan
PBM: Product backlog management
    PBQ: Product backlog quality
```

Conclusão: O resultado do teste de Wilcoxon indica que há diferença significativa entre a distribuição das amostras. Analisando o a Figura C.18, aparentemente, o fator *Product backlog management* tem mais influência em *Product backlog quality* que *Release plan*. Daí, a hipótese alternativa do teste de Wilcoxon foi definida como *RL-PBQ* é maior que *PBM-PBQ*. Dado o p-value desse teste, não pode-se rejeitar a hipótese nula. Dessa forma, pode-se afirmar com 90% de significância que o fator *Product backlog management* influência

Product backlog quality com mais intensidade do que o fator *Release plan*.

- H19-0: *Communicator and negotiator - Desired personal characteristics of the Product Owner = Available and qualified - Desired personal characteristics of the Product Owner = Empowered and committed - Desired personal characteristics of the Product Owner = Leader and team player - Desired personal characteristics of the Product Owner = Visionary and doer - Desired personal characteristics of the Product Owner*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (AAQ-POPC, CAN-POPC, EAC-POPC,
         LATP-POPC, VAD-POPC)
         stratified by practitioner
maxT = 3.7363, p-value = 0.001747

$PostHoc.Test

CAN-POPC - AAQ-POPC  0.297460690
EAC-POPC - AAQ-POPC  0.999998976
LATP-POPC - AAQ-POPC 0.002079969
VAD-POPC - AAQ-POPC  0.026432386
EAC-POPC - CAN-POPC  0.274015930
LATP-POPC - CAN-POPC 0.401520016
VAD-POPC - CAN-POPC  0.848684246
LATP-POPC - EAC-POPC 0.001715282
VAD-POPC - EAC-POPC  0.022573082
VAD-POPC - LATP-POPC 0.947606704
```

```
Onde:
LATP: Leader and team player
AAQ: Available and qualified
VAD: Visionary and doer
CAN: Communicator and negotiator
EAC: Empowered and committed
POPC: Desired personal characteristics of the Product Owner
```

Conclusão: Apesar do resultado do teste de Friedman ter indicado que há diferença entre os ranks médios das amostras e o teste post-hoc ter indicado quais pares causaram o resultado do teste de Friedman, não pode-se afirmar que alguma das variáveis tem rank médio diferente de todas as outras com 90% de significância. Por exemplo, *LATP-POPC* é diferente de *EAC-POPC* e *AAQ-POPC*. Por outro lado, *LATP-POPC* é igual a *CAN-POPC*, e *CAN-POPC* é igual a *EAC-POPC* e *AAQ-POPC*. Dessa forma, contraria-se o teste de Friedman e afirma-se que não há diferença entre o rank médio das amostras. Ou seja, os fatores *Communicator and negotiator*, *Available and qualified*, *Empowered and committed*, *Leader and team player* e *Visionary and doer* influenciam *Desired personal characteristics of the Product Owner* com a mesma intensidade.

- H20-0: *Product Owner personal characteristics - Product Owner overall work quality = Progress tracking - Product Owner overall work quality = Product backlog quality - Product Owner overall work quality*

```
$Friedman.Test

        Asymptotic General Independence Test

data:  influenceLevel by
         relationship (PBQ-POOWQ, POPC-POOWQ, PT-POOWQ)
         stratified by practitioner
```

```
maxT = 3.6585, p-value = 0.0007627

$PostHoc.Test

POPC-POOWQ - PBQ-POOWQ 0.4950611174
PT-POOWQ - PBQ-POOWQ   0.0007463225
PT-POOWQ - POPC-POOWQ  0.0309466940

Onde:
POPC: Product Owner personal characteristics
PBQ: Product backlog quality
PT: Progress tracking
POOWQ: Product Owner overall work quality
```

Conclusão: O resultado do teste de Friedman indica que há diferença entre os ranks médios das amostras e o teste post-hoc indica que, como esperado ao analisar a Figura C.20, há diferença entre os ranks médios de *PBQ-POOWQ* e *PT-POOWQ* e entre *POPC-POOWQ* e *PT-POOWQ*. Além disso, o teste de post-hoc indica que não há diferença significativa entre *PBQ-POOWQ* e *POPC-POOWQ*. A partir da Figura D.8, pode-se afirmar com 90% de significância que os fatores *Product backlog quality* e *Product Owner personal characteristics* tem mais influência sobre *Product Owner overall work quality* do que *Progress tracking*.

- H21-0: *Product Owner overall work quality - Project success = Work validation quality - Project success = Potentially shippable product increment quality (end of each sprint) - Project success*

```
$Friedman.Test

        Asymptotic General Independence Test
```

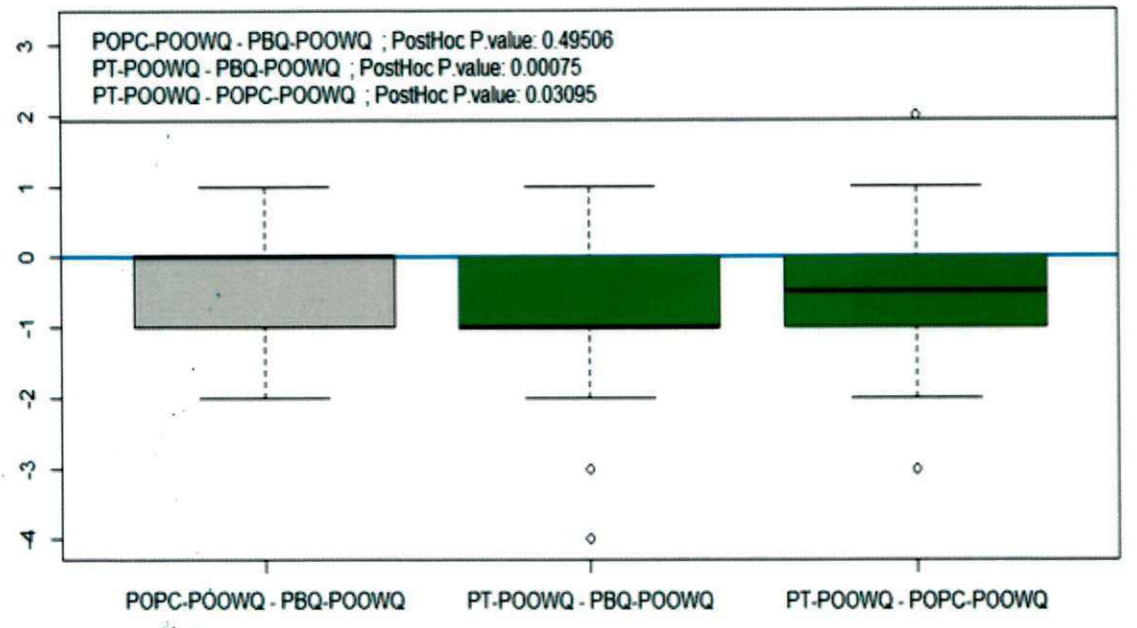

Figura D.8: Box plot das diferenças entre *POPC-POOWQ* e *PBQ-POOWQ*; *PT-POOWQ* e *PBQ-POOWQ*; e *PT-POOWQ* e *POPC-POOWQ*

```
data:  influenceLevel by
         relationship (POOPWQ-PS, PSPIQ-PS, WVQ-PS)
         stratified by practitioner
maxT = 1.8091, p-value = 0.1666


Onde:
POOPWQ: Product Owner overall work quality
PSPIQ: Potentially shippable product increment quality (end
       of each sprint)
WVQ: Work validation quality
PS: Project success
```

Conclusão: O resultado do teste de Friedman indica que, com 90% de significância, não há diferença significativa entre os ranks médios das amostras. Dessa forma, pode-se afirmar, com 90% de significância, que os fatores *Product Owner overall work quality*, *Work*

validation quality e *Potentially shippable product increment quality (end of each sprint)* influenciam *Project success* com a mesma intensidade.