Método para Avaliar o Impacto Potencial das Mudanças nos Requisitos de Software nos Projetos baseados em Metodologias Ágeis

Ângelo Amaral

Novembro / 2023

Dissertação de Mestrado em Ciência da Computação

Método para Avaliar o Impacto Potencial das Mudanças nos Requisitos de Software nos Projetos baseados em Metodologias Ágeis

Esse documento corresponde à Dissertação apresentada à Banca Examinadora no curso de Mestrado em Ciência da Computação da UNIFACCAMP – Centro Universitário Campo Limpo Paulista.

Campo Limpo Paulista, 24 de novembro de 2023.

Ângelo Amaral

Prof. Dr. Ferrucio de Franco Rosa (Orientador)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

Ficha catalográfica elaborada pela

Biblioteca Central da Unifaccamp

A512m

Amaral, Ângelo

Método para avaliar o impacto potencial das mudanças nos requisitos de *software* nos projetos baseados em metodologias ágeis / Ângelo Amaral. Campo Limpo Paulista, SP: Unifaccamp, 2023.
62 p.: il.

Orientador: Prof. Dr. Ferrucio de Franco Rosa

Dissertação (Programa de Mestrado Acadêmico em Ciência da Computação) – Centro Universitário Campo Limpo Paulista – Unifaccamp.

1. Escopo de projeto. 2. Solicitação de mudança. 3. Requisito de *software*. 4. Análise de impacto. 5. Estimativa de esforço. 6. Metodologias ágeis. I. Rosa, Ferrucio de Franco. II. Centro Universitário Campo Limpo Paulista. III. Título.

CDD – 005.1

AGRADECIMENTOS

À minha esposa Carolina e à minha filha Lívia, cuja motivação incondicional me permitiu acreditar que este era um sonho possível. Sem seu apoio, as horas de estudo e pesquisa, as madrugadas trabalhando e todo o esforço para concretizar este projeto não fariam sentido algum.

Ao professor e amigo Ferrucio de Franco Rosa, que acreditou neste trabalho e na minha capacidade quando eu mesmo tive dúvidas. Sem a sua mão guiando meus passos, eu não chegaria até aqui.

Resumo

Uma das principais questões na análise de impacto das mudanças de escopo em projetos de desenvolvimento de software é a necessidade de identificar outros requisitos potencialmente afetados pela mudança, assim como determinar a complexidade da mudança. Esses desafios são potencializados em projetos baseados em metodologias ágeis, caracterizados por uma comunicação ampla com o cliente, o que aumenta a oportunidade de solicitações de mudança durante o ciclo de vida de desenvolvimento de software. Propõe-se um novo método em que as metodologias ágeis podem se beneficiar da rastreabilidade de requisitos e da estimativa de esforço em mudanças de escopo, bem como um protótipo de software para sua implementação. Ao aplicar o método a um projeto de desenvolvimento de software, os resultados são sintetizados em indicadores que demonstram como as mudanças de escopo realizadas em sucessivos ciclos de desenvolvimento impactam outros requisitos de software e como a complexidade da mudança pode ser usada para determinar sua estimativa de esforço. As principais contribuições do presente trabalho são: (i) um método capaz de fornecer uma taxa de impacto potencial para mudanças de escopo em projetos baseados em metodologias ágeis e uma lista de requisitos potencialmente impactados pela mudança; (ii) uma métrica para classificar os requisitos de software com base em seu impacto potencial, apoiando a estimativa de esforço e análise de impacto; (iii) uma faixa de valores para determinar o risco de aceitar uma solicitação de mudança de escopo; e (iv) um protótipo de software que implementa o método. Uma discussão sobre os resultados da aplicação da abordagem ao desenvolvimento de software ágil é apresentada. A proposta de solução é destinada a equipes de desenvolvimento de software no contexto de projetos baseados em metodologias ágeis.

Palavras-chave: Escopo de Projeto, Solicitação de Mudança, Requisito de Software, Análise de Impacto, Estimativa de Esforço, Metodologias Ágeis.

Abstract

When analyzing the impact and effort resulting from scope changes in software development projects, one of the main issues is the need to identify other requirements potentially affected by the change, as well as determine the complexity of the change. These challenges are amplified in projects based on agile methodologies, characterized by extensive communication with the client, which increases the opportunity for change requests during the software development lifecycle. A new method is proposed in which agile methodologies can benefit from requirement traceability and effort estimation in scope changes, along with a software prototype for its implementation. When applying the method to a software development project, the results are synthesized into indicators that demonstrate how scope changes made in successive development cycles impact other software requirements and how the complexity of the change can be used to determine its effort estimation. The main contributions of this work are: (i) a method to provide a potential impact rate on scope changes in projects based on agile methodologies and a list of requirements potentially impacted by the change; (ii) a metric for classifying software requirements based on their potential impact, supporting effort estimation and impact analysis; (iii) a range of potential impact rate values to determine the risk of accepting a scope change request; and (iv) a software prototype capable of implementing the presented method. A discussion of the results arising from the application of the approach into an agile software development scenario is presented. This solution is intended to be useful for software development teams in the context of projects based on agile methodologies.

Keywords: *Project Scope, Change Request, Software Requirement, Impact Analysis, Effort Estimate, Agile.*

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS

CVDS	Ciclo de vida de desenvolvimento de software
XP	*Extreme Programming*
TDD	*Test-driven design*
FDD	*Feature-driven design*
DSDM	*Dynamic System Development Method*
HTML	*Hypertext Markup Language*
PHP	*Hypertext Processor*
SQL	*Structured Query Language*
B2B	*Business to Business*
B2C	*Business to Consumer*
MAPIR-CSR	*Method for Assessing the Potencial Impact Rate of Changes in Software Requirements*

1 INTRODUÇÃO

Ao projetar aplicações e serviços, uma questão-chave é a avaliação do impacto das mudanças de escopo nos requisitos de software. Isso se torna crítico ao adotar metodologias ágeis de desenvolvimento de software, cujo foco na transparência e na comunicação entre a equipe de desenvolvimento e o cliente pode resultar em uma documentação menos detalhada.

De acordo com Sommerville (2011), os requisitos podem ser organizados em uma hierarquia composta por Requisitos do Usuário, que descrevem as expectativas de alto nível do usuário final, e Requisitos do Sistema, que refletem aspectos-chave da implementação. Os requisitos do sistema podem ser derivados em requisitos de software, classificados como Funcionais e Não-Funcionais, referindo-se a implementações relacionadas às funcionalidades do software e às restrições do ambiente, respectivamente.

Essa abordagem se aplica ao ciclo de vida de desenvolvimento de software (CVDS), em uma visão preditiva tradicional (Díaz et al., 2013; Rubasinghe et al., 2018), enquanto projetos baseados em metodologias ágeis (*e.g.*, *Framework Scrum*) utilizam um modelo baseado em uma lista de requisitos de alto nível, denominada *backlog do produto*, que é decomposta iterativamente durante o CVDS (Raj et al., 2015; Schwaber & Sutherland, 2020; Y. Wang et al., 2017).

O *backlog do produto* descreve os requisitos de usuário em alto nível (Schwaber & Sutherland, 2020), se assemelhando aos requisitos de sistema de Sommerville (2011), enquanto o conjunto de "*Backlogs das Sprints*" descreve os itens que devem ser implementados pela equipe de desenvolvimento para entregar o software, correspondendo ao conceito de requisitos de software de Sommerville. Além disso, o *Backlog* de uma *Sprint* geralmente contém tarefas que podem ser associadas tanto a requisitos Funcionais quanto Não-Funcionais.

Com base nesta estrutura, faltam métodos nos quais Analistas de Requisitos e outros profissionais que atuam como "*Product Owners*" em projetos baseados em metodologias ágeis possam se beneficiar da rastreabilidade de requisitos para apoiar a análise de impacto e a estimativa de esforço em mudanças de escopo (Rubasinghe et al., 2018).

1.1 Questão de Pesquisa, Objetivos e Contribuições

A partir da problemática exposta, aborda-se a seguinte **questão de pesquisa**: "*Como avaliar o impacto de mudanças de requisitos em projetos de software geridos com uso de metodologias ágeis?*".

O objetivo geral e os objetivos específicos são apresentados a seguir.

Objetivo Geral: Propor um método voltado a avaliar o impacto potencial de mudanças em requisitos de software em projetos baseados em metodologias ágeis.

Objetivos Específicos:

(i) Propor um método que possibilite a avaliação do impacto potencial de mudanças de escopo de produto em projetos baseados em metodologias ágeis, provendo uma lista de requisitos potencialmente impactados por uma mudança específica;

(ii) Definir uma métrica para classificação de requisitos de software com base em seu impacto potencial que possibilite estimar esforço e analisar impacto;

(iii) Identificar faixas de valores que podem determinar o risco de aceitar uma determinada solicitação de mudança de escopo em determinado requisito de software; e

(iv) Desenvolver um protótipo de software para verificar a aplicabilidade do método.

As principais **contribuições** do presente trabalho são: (i) um método capaz de fornecer uma taxa de impacto potencial para mudanças de escopo em projetos baseados em metodologias ágeis e uma lista de requisitos potencialmente impactados pela mudança; (ii) uma métrica para classificar os requisitos de software com base em seu impacto potencial, apoiando a estimativa de esforço e análise de impacto; (iii) uma faixa de valores para determinar o risco de aceitar uma solicitação de mudança de escopo; e (iv) um protótipo de software que implementa o método. Uma discussão sobre os resultados da aplicação da abordagem ao desenvolvimento de software ágil foi conduzida. A proposta

de solução é destinada a equipes de desenvolvimento de software no contexto de projetos baseados em metodologias ágeis.

1.2 Estrutura da Dissertação

O restante deste trabalho está dividido da seguinte forma:

- O **Capítulo 2** apresenta o referencial teórico, com a introdução de conceitos essenciais à discussão dos temas abordados nos capítulos subsequentes.
- O **Capítulo 3** apresenta uma revisão sistemática da literatura a respeito da análise de impacto e estimativa de esforço das mudanças de escopo em projetos baseados em metodologias ágeis, buscando identificar o estado da arte adotado como base para a proposição do método, além de mapear trabalhos relacionados.
- O **Capítulo 4** apresenta o MAPIR-CSR (*Method for Assessing the Potencial Impact Rate of Changes in Software Requirements*), seu processo de aplicação e o protótipo de software para sua implementação.
- O **Capítulo 5** apresenta um cenário de aplicação do método em um projeto de desenvolvimento de software baseado em metodologias ágeis, assim como uma discussão sobre os resultados obtidos.
- O **Capítulo 6** apresenta as conclusões, incluindo limitações, propostas de trabalhos futuros e resultados da pesquisa.

2 REFERENCIAL TEÓRICO E METODOLÓGICO

Neste capítulo, apresentam-se conceitos importantes para um melhor entendimento do trabalho, considerando o contexto em que são abordados nas seções subsequentes.

Conceitos do domínio desenvolvimento de software baseado em metodologias ágeis são oriundos de (Schwaber & Sutherland, 2020). Conceitos relacionados aos tipos de requisitos em um projeto de software e seu relacionamento são oriundos de (Sommerville, 2011). Referências bibliográficas adicionais, oriundas da revisão de literatura, foram selecionadas para compor o referencial teórico.

Segundo Pressman & Maxim (2021), a engenharia de software tem seu foco na construção de sistemas de software eficientes e confiáveis, tendo como fator crítico de sucesso a definição, compreensão e gestão dos requisitos, que traduzem as necessidades e expectativas com relação ao software, sendo tanto o ponto de partida do processo de desenvolvimento como o principal elemento para validação de suas entregas.

Dentro deste contexto, os requisitos são identificados de forma iterativa (Wiegers, 2009), com o mapeamento inicial das necessidades indicadas pelo cliente, seu detalhamento técnico para estruturação do sistema e a implementações de código e restrições de ambiente resultantes, representando respectivamente os requisitos de usuário, requisitos de sistema e requisitos de software.

Os Requisitos de Usuário descrevem as expectativas de alto nível do usuário final, indicando os problemas ou necessidades que devem ser atendidos por meio da implementação de um sistema (Sommerville, 2011). Requisitos de Usuário são utilizados para registrar as funcionalidades desejadas do sistema sob a ótica do usuário, muitas vezes sem incorporar um detalhamento técnico, sendo o ponto de partida para a definição da finalidade do sistema. Segundo Heath (2020), os Requisitos de Usuário devem ser mapeados de forma clara e concisa, garantindo que o software produzido esteja alinhado às expectativas do cliente.

Requisitos de Usuário são comumente expressos em linguagem natural. Como exemplos, pode-se citar: "*Criar um e-Commerce com site de vendas e controle de inventário*"; "*O sistema deve permitir a realização de compras online de forma segura a*

partir de um celular". Este tipo de requisito pode originar múltiplos requisitos de sistema, que vão pautar a especificação técnica das implementações necessárias para atender a cada uma destas necessidades.

Requisitos de Sistema refletem aspectos-chave da implementação e traduzem a forma como será abordada a solução de cada problema apresentado. Requisitos de Sistema não descrevem elementos de software, como telas e relatórios, mas sim estratégias de implementação, como o mapeamento das funcionalidades que darão origem a estes elementos (Sommerville, 2011). Desta forma, este tipo de requisito surge na etapa seguinte ao levantamento de requisitos de usuário no processo de elicitação de requisitos (Pressman & Maxim, 2021), sendo derivados dos Requisitos de Usuário.

Os Requisitos de Sistema representam a tradução das necessidades apresentadas pelo cliente em especificações técnicas que pautam o desenvolvimento do software, detalhando as funcionalidades, limitações e comportamentos que o sistema deve reproduzir. Requisitos de Sistema direcionam as decisões relacionadas à arquitetura do software, desempenhando um papel crítico na definição do escopo do projeto (Campfield et al., 2013).

Como exemplo, considerando um sistema de comércio eletrônico, pode ser citado como um Requisito de Sistema o seguinte: "*O controle de inventário deve manter registro dos produtos com saldo de estoque e permitir pesquisas*". No exemplo, detalha-se como o controle de estoque deve ser realizado, servindo de ponte entre a necessidade apresentada inicialmente e as implementações que serão realizadas pela equipe de desenvolvimento.

Os Requisitos de Software são derivações dos Requisitos de Sistema, podendo ser classificados como "*funcionais*" e "*não-funcionais*" (Wiegers & Beatty, 2013). Requisitos de Software se referem a implementações relacionadas às funcionalidades do software e às restrições do ambiente. Segundo Wiegers (2009), neste nível de requisitos são descritos efetivamente os elementos que devem ser construídos pelos programadores, *e.g.*, as telas do sistema (requisitos de software funcionais) e a criptografia do banco de dados (requisitos de software não-funcionais).

Requisitos de Software são mais específicos e detalhados do que Requisitos de Sistema, sendo direcionados à equipe de desenvolvimento. Este tipo de requisito visa a

determinar o que efetivamente deve ser construído no sistema, descrevendo em detalhes como o software deve ser projetado e implementado. Requisitos de Sistema incluem especificações técnicas, requisitos de performance, interfaces com outros sistemas e aspectos relacionados à segurança da informação (Pressman & Maxim, 2021).

No exemplo do sistema de comércio eletrônico, um Requisito de Software poderia partir da redação "*Tela de controle de estoque com geração de relatórios*" e detalhar o processo de exportação do relatório, a interface que deve ser criada para interação com o usuário e seu processo de controle de transações.

Desta forma, nota-se que ocorre um relacionamento entre requisitos, representando a derivação de requisitos de Usuário em requisitos de Sistema e de requisitos de Sistema em requisitos de Software, permitindo estabelecer rastreabilidade entre eles (Sommerville, 2011). Segundo Heath (2020), requisitos de Usuário, Sistema e Software estão interconectados, resultando em um fluxo de informações que pauta o processo de desenvolvimento de software.

Para compreensão deste relacionamento, pode-se considerar um modelo no qual os Requisitos de Usuário estão no nível mais alto, seguidos por múltiplos Requisitos de Sistema e, finalmente, por múltiplos Requisitos de Software. Alterações em um determinado nível de requisito podem afetar os níveis seguintes, tornando a gestão de requisitos uma tarefa crítica (Taromirad & Paige, 2012). O método proposto leva em consideração a hierarquia entre os requisitos na análise de impacto potencial de mudanças nos requisitos.

Quanto mais frequentes forem as mudanças solicitadas pelo cliente nos requisitos de um projeto, maior será seu grau de incerteza e, portanto, de risco (Project Management Institute, 2017). Isto impacta diretamente na gestão dos requisitos de sistema e dos requisitos de software, resultando em problemas para as abordagens preditivas de desenvolvimento de software, que tratam a especificação de requisitos como uma etapa inicial no ciclo de vida do desenvolvimento de software (Pressman & Maxim, 2021; Sommerville, 2011).

Considerando a necessidade de resposta a um cenário de mudanças constantes nos requisitos, este trabalho tem seu foco nas metodologias ágeis para desenvolvimento de software, baseadas em um paradigma onde, de acordo com Beck et al. (2001), assume-se

que: i) capacidade de resposta às mudanças deve ser mais valorizada que seguir um plano previamente estabelecido, ii) um software plenamente operativo deve ser mais valorizado que uma documentação extensa, iii) a colaboração do cliente deve ser mais valorizada que a negociação de um contrato e iv) a interação entre indivíduos deve ser mais valorizada que processos e ferramentas.

Segundo Sletholt et al. (2011), Hamed & Abushama (2013) e Schön et al. (2017), as metodologias ágeis podem abordadas com o apoio do *Framework Scrum* (Schwaber & Sutherland, 2020), considerando-se a correlação entre suas práticas de análise e desenvolvimento de software, bem como o modelo de representação e relacionamento de requisitos apresentado. Outras metodologias ágeis, tais como *Kanban*, *XP*, *TDD*, *FDD* e *DSDM* estão fora do escopo deste trabalho.

Para facilitar o entendimento do *Framework Scrum*, faz-se necessário apresentar seus conceitos principais. *Sprints* são ciclos de desenvolvimento sucessivos e recorrentes, com duração constante, em que determinados requisitos de software são implementados de acordo com sua prioridade, para compor uma entrega incremental de software ao cliente em intervalos regulares (Schwaber & Sutherland, 2020), compreendendo atividades de planejamento, implementação e testes em cada ciclo.

Segundo Schön et al. (2017), *Sprints* são uma resposta à natureza dinâmica dos requisitos de software e suas recorrentes mudanças. Em contrapartida à dificuldade em prever todos os requisitos no início do projeto, os times de desenvolvimento se adaptam ao *feedback* do cliente a cada ciclo de desenvolvimento, tornando o processo mais ágil e responsivo.

O *Backlog do Produto* reflete um conjunto de requisitos de sistema descritos em alto nível, que serão decompostos e implementados em *Sprints* subsequentes (Schwaber & Sutherland, 2020), representando um artefato central no gerenciamento ágil de requisitos (Heath, 2020). O *Backlog do Produto* é representado por uma lista contendo todos os requisitos de sistema, priorizada com base no valor agregado por cada requisito para o cliente, a partir das informações providas por ele e revisadas periodicamente, o que torna este um artefato bastante dinâmico (Barton & Campbell, 2007).

Segundo Ayed et al. (2014), além de servir como um guia para a equipe de desenvolvimento, o *Backlog do Produto* fornece visibilidade para o cliente sobre as

próximas entregas do projeto, sendo frequentemente escrito em linguagem natural, com o objetivo de facilitar seu entendimento por todas as partes interessadas.

O *Backlog da Sprint* representa um conjunto de Requisitos de Software, derivados do *Backlog do Produto* e selecionados para implementação em uma *Sprint* específica (Schwaber & Sutherland, 2020). Segundo Bass (2014), uma vez definido, o *Backlog da Sprint* não pode ser alterado durante a *Sprint*, garantindo estabilidade e foco na entrega à equipe de desenvolvimento. Este artefato tem por objetivo manter a equipe concentrada no que é mais importante para o cliente em curtos períodos, permitindo que seja mantido o compromisso de entregar o que foi planejado (Heath, 2020), enquanto as solicitações de mudança recebidas durante a execução da *Sprint* corrente são avaliadas quanto ao seu impacto e complexidade e endereçadas para as *Sprints* subsequentes.

Cada item presente no *Backlog da Sprint* é classificado de acordo com uma pontuação medida em *Story Points*, que representam a estimativa da complexidade do requisito de software em relação aos demais requisitos presentes no *backlog*, cuja precisão é balizada por estimativas anteriores bem-sucedidas no mesmo projeto (Schwaber & Sutherland, 2020). *Story Points* são insumos para o método proposto.

A responsabilidade pela manutenção do *Backlog do Produto* é do *Product Owner*, que interage constantemente com o cliente para mapeamento, priorização e decomposição dos requisitos de usuário, sistema e software, bem como para elaboração do *Backlog da Sprint* junto ao time de desenvolvimento e para definição de valores de *Story Points* estimados pelos desenvolvedores para cada requisito de software (Schwaber & Sutherland, 2020). O *Product Owner* é visto como o representante do cliente e dos usuários no processo (Jakobsen & Sutherland, 2009) e deve inclusive tomar decisões a respeito da aceitação ou negociação das solicitações de mudanças nos requisitos.

Cabe também ao *Product Owner* a responsabilidade de realizar junto ao time de desenvolvimento o *Refinamento dos Backlogs*, uma vez que os *backlogs de Produto* e *Sprint* podem ser submetidos a processos de decomposição e detalhamento de requisitos (Schwaber & Sutherland, 2020). A realização do refinamento dos requisitos no momento de sua criação é uma premissa deste trabalho, pois a estimativa inicial de *Story Points* para cada requisito de software é um dos insumos do método proposto.

Durante o refinamento, os itens podem ser divididos em tarefas menores, às quais são adicionados detalhes técnicos e critérios de aceitação para a implementação que será realizada, preparando os itens para serem desenvolvidos com mais eficiência e qualidade (Kaur et al., 2021).

Sempre que o time de desenvolvimento recebe uma solicitação do cliente referente à mudança de um requisito de software já implementado anteriormente, é realizada uma análise de impacto para entender como esta solicitação afetará o sistema como um todo, envolvendo a estimativa de esforço, ou seja, o cálculo da complexidade das tarefas necessárias para a realização do trabalho.

A execução da análise de impacto se torna fundamental para a compreensão das implicações de uma mudança, além de preparar a equipe para potenciais dificuldades técnicas resultantes. Por sua vez, a estimativa de esforço apoia o planejamento realista do trabalho da *Sprint*, estabelecendo expectativas claras a respeito do escopo.

No contexto deste trabalho, o termo *Análise de Impacto* se refere ao objetivo de identificar, por meio de alguma métrica, quão crítica em termos de esforço poderia ser uma mudança em determinado requisito de usuário, sistema ou software. Por exemplo, uma mudança em um determinado requisito de usuário poderia resultar na criação ou modificação de outros requisitos de sistema ou requisitos de software, impactando custo, prazo e qualidade. Por sua vez, o termo *Estimativa de Esforço* se refere ao grau de complexidade para que o time de desenvolvimento implemente a mudança em questão.

Cabe mencionar que aspectos relacionados à validação e teste de software não integram o escopo desta dissertação, uma vez que o foco do trabalho está na análise de impacto potencial das solicitações de mudança antes de seu aceite e implementação. Neste contexto, Vasic et al. (2017) abordam o conceito de teste de regressão no desenvolvimento de software, e sua dependência da implementação concluída, sem relação com o uso da rastreabilidade entre requisitos, motivo pelo qual esta também não é considerada.

3 REVISÃO DE LITERATURA E TRABALHOS RELACIONADOS

Neste capítulo, apresenta-se a revisão sistemática de literatura, o processo de identificação e seleção dos artigos analisados e a discussão tanto dos artigos analisados quanto dos artigos considerados relacionados a este trabalho.

O objetivo principal desta revisão foi identificar o estado da arte no que se refere às práticas para análise de impacto e estimativa de esforço nas solicitações de mudança em requisitos de software, focados em projetos baseados em metodologias ágeis, de forma a pautar a abordagem proposta para a definição de um método que possa amparar tais atividades.

3.1 Metodologia da Revisão

Foram analisados 33 artigos nesta revisão sistemática da literatura sobre análise de impacto na gestão de mudanças em projetos de metodologias ágeis. A abordagem utilizada na revisão da literatura foi baseada nas diretrizes propostas por Kitchenham (2004) e a pesquisa foi orientada pela pergunta "*Como avaliar o impacto das mudanças de escopo em projetos baseados em metodologias ágeis?*".

Após consultas às bibliotecas ACM Digital Library e IEEE Xplore (as bibliotecas Scopus e Springer Link não foram utilizadas pela redundância de retorno com os resultados obtidos nas bibliotecas anteriores), seguiu-se uma análise exploratória classificando os artigos em 6 objetivos: i) utilização de um modelo de relacionamento existente (Ru) para organizar requisitos, ii) proposição de um novo modelo de relacionamento (Rp) para organizar requisitos, iii) apresentação de um modelo (M) para gestão dos requisitos, iv) apresentação de uma ferramenta de software (F) para gerenciar os requisitos, v) suporte à estimativa de esforço da mudança (EE) e vi) suporte à análise de impacto de mudança (AI).

Além destes objetivos, foram considerados 4 domínios de aplicação durante a análise: i) revisão do CVDS, ii) cobertura de metodologias ágeis, iii) cobertura de metodologias preditivas e iv) presença de uma abordagem baseada em algoritmo para controle das mudanças em requisitos de projetos baseados em metodologias ágeis.

A consulta utilizada para a busca nas bibliotecas buscou artigos publicados entre 2010 e 2022, sendo baseada, com adaptações de sintaxe, na seguinte *string* de busca:

(Impact AND (Analysis OR Evaluation OR Assessment) AND (Change OR Changing) AND (Agile OR SCRUM) AND Project).

A consulta resultou em 44 artigos, dos quais 29 retornaram de IEEE Xplore e 15 de ACM Digital Library. Dois artigos foram encontrados em ambas as bibliotecas, pois foram publicados em uma conferência internacional conjunta IEEE-ACM. Após remover a redundância, a lista preliminar foi composta por 42 artigos.

Como Critérios de Inclusão, foram considerados todos os artigos retornados, mais 2 trabalhos adicionais (Rubasinghe et al., 2018; C. Wang et al., 2018), citados por outros artigos retornados (processo de revisão por *snowballing*), totalizando 44 trabalhos avaliados. Como Critérios de Exclusão, durante a análise, 11 artigos foram identificados como não aderentes ao foco do trabalho e foram desconsiderados.

Dos 33 artigos resultantes, 24 foram classificados como “trabalhos analisados”, apresentados na Subseção 3.2.1, e 9 foram classificados como “trabalhos relacionados”, apresentados na Subseção 3.2.2. Os trabalhos analisados e relacionados combinam novas abordagens e o estado da arte da pesquisa atual, com o conceito de uma abordagem sistêmica, estabelecendo aspectos-chave para a análise de impacto das mudanças em requisitos de software.

A síntese desses trabalhos pode ser encontrada nas próximas subseções, que apresentam também a sua classificação. Considerando a revisão da literatura, identifica-se que a adoção de um modelo de relacionamento de requisitos para suportar a rastreabilidade na análise de impacto é uma estratégia promissora.

Adicionalmente à revisão de literatura realizada, foram consultados também outros trabalhos (Amaral & Nicoletti, 2021; Amaral & Rosa, 2023b; Asl & Kama, 2013; Barton & Campbell, 2007; Bass, 2014; Beck et al., 2001; Campfield et al., 2013; Cleland-Huang, 2013; Garcia, 2015; Girma et al., 2019; Hannay et al., 2019; Heath, 2020; Jakobsen & Sutherland, 2009; Kama & Azli, 2012; Kitchenham, 2004; Popli & Chauahn, 2015; Pressman & Maxim, 2021; Schön et al., 2017; Schwaber & Sutherland, 2020; Sjoberg et al., 2010; Sommerville, 2011; Wiegers, 2009; Wiegers & Beatty, 2013;

Zahraoui & Janati Idrissi, 2015) como referência metodológica, integrando a relação de obras citadas ao longo desta dissertação.

3.2 Resultados da Revisão de Literatura

Esta seção apresenta uma análise sintética dos 33 estudos selecionados, classificados como i) trabalhos analisados (24 artigos sintetizados na Subseção 2.2.1) e ii) trabalhos relacionados (9 artigos sintetizados na Subseção 2.2.2).

3.2.1 Discussão sobre os trabalhos analisados

Sletholt et al. (2011) apresentam uma revisão da literatura sobre análise de impacto das mudanças de requisitos em projetos baseados em metodologias ágeis. O estudo concentra-se estritamente na análise de impacto, enquanto o presente trabalho tem o foco na análise de impacto e estimativa de esforço para mudanças de escopo, apresentando uma atualização do mapeamento sistemático, classificação das referências e propõe um método para apoiar a tomada de decisão.

Basri, Kama, Haneem & Ismail (2016) e Basri, Kama, Sarkan, Adili & Haneem (2016) descrevem a implementação de modelos para estimar o esforço necessário para implementar solicitações de mudança e apoiar a análise de impacto em projetos de desenvolvimento de software, aplicando-os em metodologias ágeis e preditivas. Estes artigos abordam a adoção de uma ferramenta proposta previamente (Asl & Kama, 2013; Kama & Azli, 2012) para análise de impacto de mudanças em projetos de metodologias ágeis, sem o uso de uma estrutura de relacionamento entre os requisitos.

Kim et al, (2010) e Wang et al. (2018) propõem métodos e ferramentas para análise de impacto das mudanças, partindo da análise do código-fonte para estimar o impacto das mudanças de escopo. Nenhum dos artigos considera explicitamente projetos de metodologias ágeis e uma diferença significativa entre eles é o fato de que Wang et al. adicionaram o aspecto de estimativa de esforço ao seu método.

Krzanik et al. (2010) apresentam um método para análise de impacto em projetos baseados em metodologias ágeis e estima o esforço a partir de uma perspectiva de alto nível dos processos de gerenciamento de projetos.

Silva et al. (2018) descrevem um modelo baseado em testes para gerenciar o impacto de mudanças de escopo em projetos ágeis, sem utilizar um modelo de relacionamento de requisitos para apoiar a rastreabilidade. Cao et al. (2010), Vasic et al. (2017), Tufano et al. (2019) e Habib et al. (2021) apresentam processos para abordar as mudanças em projetos ágeis, porém não estabelecem associação entre os requisitos.

Jonkers et al. (2021) propõem um modelo para lidar com as mudanças em projetos ágeis que, entretanto, carece de uma abordagem para estimativa de esforço, além de também não propor levar em consideração os relacionamentos entre requisitos.

Raj et al. (2015), Duarte et al. (2011), Abou Khalil (2019), Hamed et al. (2013) e Ochoa et al. (2021) apresentam visões detalhadas do ciclo de vida de desenvolvimento de software, sem focar especificamente na análise de impacto de mudanças nos requisitos.

Kaur et al. (2021), Malhotra et al. (2016), Manisha et al. (2021) e Murphy et al. (2013) contribuem para a discussão sobre a análise de impacto de solicitações de mudança, porém sem oferecerem contribuições significativas.

Rajlich (2010), Rajlich (2013) e Włodarski et al. (Włodarski et al., 2020) concentram-se no ensino de engenharia de software, abordando os tópicos deste trabalho em um nível básico.

A Tabela 1 sintetiza os trabalhos analisados, além de classificá-los em função dos objetivos e domínios de aplicação propostos na revisão de literatura.

Tabela 1: *Síntese dos Trabalhos Analisados*

Autores	Objetivos						Domínios			
	Ru	**Rp**	**M**	**F**	**EE**	**AI**	**1**	**2**	**3**	**4**
(Abou Khalil, 2019)			X				X			
(Basri, Kama, Haneem, et al., 2016)			X		X	X	X	X	X	X
(Basri, Kama, Sarkan, et al., 2016)			X		X	X	X	X	X	X
(Cao et al., 2010)			X			X	X	X		
(Duarte et al., 2011)			X					X		
(Habib & Romli, 2021)			X			X	X	X		
(Hamed & Abushama, 2013)					X	X	X			
(Jonkers & Eftekhari Shahroudi, 2021)			X	X		X	X	X		
(Kaur et al., 2021)						X	X	X		
(Kim et al., 2010)			X	X		X			X	X
(Krzanik et al., 2010)				X	X	X		X		
(Malhotra & Chug, 2016)						X	X	X		
(Manisha et al., 2021)						X	X	X		
(Murphy & Williams, 2013)						X	X	X	X	
(Ochoa et al., 2021)					X	X	X	X		
(Raj et al., 2015)			X					X		
(Rajlich, 2010)						X	X	X	X	
(Rajlich, 2013)						X	X	X	X	
(Silva et al., 2018)			X			X	X	X		
(Sletholt et al., 2011)	X		X		X	X		X		
(Tufano et al., 2019)			X			X	X	X		
(Vasic et al., 2017)			X			X	X	X		
(C. Wang et al., 2018)			X	X	X	X			X	X
(Włodarski et al., 2020)						X	X	X	X	

Objetivos: Relacionamento entre Requisitos – uso de (Ru); Relacionamento entre Requisitos – proposta de (Rp); Modelo (M); Ferramenta (F); Estimativa de Esforço (EE); Análise de Impacto (AI). **Domínios:** (1) Ciclo de Vida do Desenvolvimento de Software; (2) Metodologias Ágeis; (3) Metodologias Preditivas; (4) Baseado em Algoritmo.

3.2.2 Discussão sobre os trabalhos relacionados

Os artigos que apresentaram contribuições à análise de impacto e estimativa de esforço em solicitações de mudança foram considerados trabalhos relacionados, conforme detalhado nos parágrafos a seguir.

Díaz et al. (2013) propõem uma abordagem para estabelecer relações entre requisitos de software, visando sua aplicação nos projetos baseados em metodologias ágeis, concentrando-se na análise do impacto das solicitações de mudança e excluindo o aspecto de estimativa de esforço para a implementação da mudança propriamente dita.

Wang et al. (2017) apresentam o conceito de *Épico*, que pode ser decomposto em uma ou mais Histórias, sendo a base para o *Backlog do Produto* e o *Backlog da Sprint* (Schwaber & Sutherland, 2020). Eles também apresentam o uso de um modelo de relação de requisitos, que remete à classificação de requisitos de Sommerville (2011), com foco na documentação de requisitos de metodologias ágeis.

Rubasinghe et al. (2018) apresentam um modelo de rastreabilidade de requisitos aderente à abordagem apresentada por Wang et al. (2017), adicionando uma perspectiva focada em metodologias ágeis ao modelo proposto, por meio do conceito da decomposição iterativa do *backlog*.

Ahmad et al. (2021) consideram a análise de impacto de mudanças e a estimativa de esforço como fatores de sucesso para projetos ágeis, apresentando uma revisão em alto nível destas abordagens. Gary et al. (2015) apresenta uma abordagem semelhante em alto nível em relação ao foco de Ahmad et al. (2021).

Zapotecas-Martinez et al. (2020) e Benedicenti et al. (2016) propõem estruturas de relacionamento para decompor requisitos em metodologias ágeis, se aproximando da abordagem de Díaz et al. (2013). Adicionalmente, Taromirad & Paige (2012) apresentam um método para decomposição de requisitos sem focar na análise de impacto de mudanças.

Utz (2019) define um meta-modelo para processos de negócio, capaz de suportar a análise de impacto das solicitações de mudança e sua estimativa de esforço em um trabalho baseado em abordagens preditivas de desenvolvimento de software, embora não exclua explicitamente a aplicação a projetos baseados em metodologias ágeis.

A Tabela 2 sintetiza os trabalhos relacionados, além de classificá-los em função dos objetivos e domínios de aplicação propostos na revisão de literatura.

Tabela 2: *Síntese dos Trabalhos Relacionados*

Autores	Objetivos						Domínios			
	Ru	**Rp**	**M**	**F**	**EE**	**AI**	**1**	**2**	**3**	**4**
(Ahmad et al., 2021)	X				X	X	X	X		
(Benedicenti et al., 2016)	X					X	X	X		
(Díaz et al., 2013)		X	X			X	X	X		
(Gary & Xavier, 2015)	X				X	X	X	X		
(Rubasinghe et al., 2018)	X		X	X		X	X		X	X
(Taromirad & Paige, 2012)		X	X					X		
(Utz, 2019)			X				X	X	X	
(Y. Wang et al., 2017)	X		X				X	X		
(Zapotecas-Martinez et al., 2020)	X							X		
Esta Dissertação	**X**		**X**	**X**	**X**	**X**	**X**	**X**		**X**

Objetivos: Relacionamento entre Requisitos – uso de (Ru); Relacionamento entre Requisitos – proposta de (Rp); Modelo (M); Ferramenta (F); Estimativa de Esforço (EE); Análise de Impacto (AI). **Domínios:** (1) Ciclo de Vida do Desenvolvimento de Software; (2) Metodologias Ágeis; (3) Metodologias Preditivas; (4) Baseado em Algoritmo.

Conforme apresentado na Tabela 2, em comparação aos demais trabalhos relacionados, este trabalho não propõe um novo modelo para relacionamento entre requisitos, por fazer uso do relacionamento proposto por Sommerville (2011) e não cobre aspectos específicos de projetos baseados em metodologias preditivas de desenvolvimento de software, por ser focado nas solicitações de mudança em requisitos de projetos baseados em metodologias ágeis.

Ao longo desta dissertação, são apresentados tanto um modelo quanto uma ferramenta para análise de impacto e estimativa de esforço das mudanças com base em um algoritmo discutido nas próximas seções, o que oferece uma cobertura mais ampla do tema do que qualquer dos trabalhos relacionados.

4 METHOD FOR ASSESSING THE POTENTIAL IMPACT RATE OF CHANGES IN SOFTWARE REQUIREMENTS (MAPIR-CSR)

Propõe-se MAPIR-CSR (*Method for Assessing the Potential Impact Rate of Changes in Software Requirements*) (Amaral & Rosa, 2023b) com o objetivo principal de apoiar a análise de impacto e a estimativa de esforço das solicitações de mudanças em requisitos de software nos projetos baseados em metodologias ágeis.

Na Figura 1 apresenta-se o MAPIR-CSR e suas i) Entradas - Requisitos de Sistema (α), Requisitos de Software (β) e *Story Points* (γ); além de suas ii) Saídas - Taxa de Impacto Potencial da Mudança (ε) e Lista de Requisitos Potencialmente Afetados (δ).

Figura 1: *MAPIR-CSR - Entradas e Saídas. (Amaral & Rosa, 2023b). Reproduzido com permissão da Springer Nature.*

4.1 Processo de Aplicação do MAPIR-CSR

Esta seção apresenta o processo de aplicação do MAPIR-CSR, que especifica os passos necessários para aplicar este método na prática. A utilização do processo é destinada a analistas de requisitos e outros profissionais que atuam como *Product Owners* em projetos baseados em metodologias ágeis, que necessitem conduzir a análise de impacto e a estimativa de esforço em solicitações de mudanças de escopo.

Este método considera um modelo de relacionamento de requisitos baseado na abordagem de Sommerville (2011) com o intuito de mapear casos em que a alteração de um requisito software possa impactar outros requisitos de software do projeto, detalhado no Apêndice 1 com base no trabalho publicado por Amaral & Rosa (2023b) e ilustrado na Figura 2, com requisitos de Usuário, Sistema e Software.

Os requisitos de software (β) são utilizados para análise de impacto das solicitações de mudança e sua conexão com os requisitos de sistema (α) oferece suporte à rastreabilidade, identificando quais outros requisitos de software (δ) são potencialmente afetados por uma mudança de escopo.

O terceiro parâmetro de entrada é a medida de complexidade, chamada de *Story Points* (γ), para cada requisito de software, possibilitando a estimativa de esforço por meio de uma taxa de impacto potencial da mudança (ε). Um *Story Point* é uma estimativa relativa da complexidade da atividade em comparação com outras atividades do projeto (Schwaber & Sutherland, 2020; Zahraoui & Janati Idrissi, 2015), cuja precisão é balizada por estimativas anteriores bem-sucedidas no mesmo projeto.

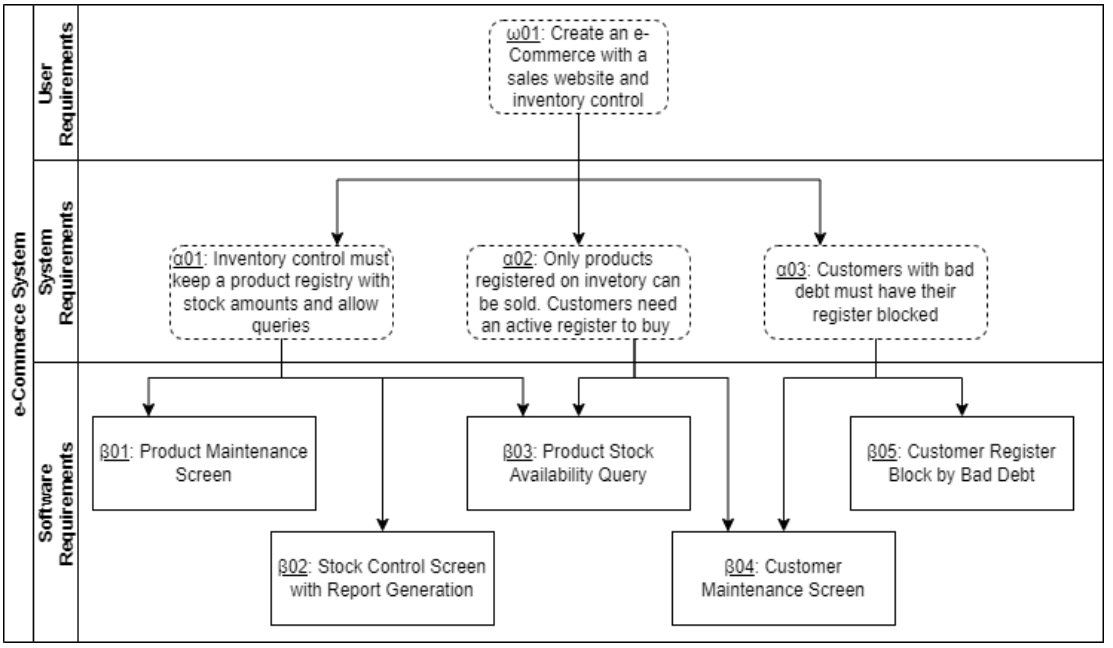

Figura 2: *Modelo de Relacionamento de Requisitos para um Sistema de e-Commerce Hipotético (Amaral & Rosa, 2023b). Reproduzido com permissão da Springer Nature.*

Considerando o uso do *Framework Scrum* (Schwaber & Sutherland, 2020), as solicitações do cliente resultam no *backlog do produto*, que é um conjunto de requisitos de alto nível, necessários para entregar o sistema. Os itens do *backlog do produto* são decompostos e implementados em sprints subsequentes (Amaral & Nicoletti, 2021).

Os requisitos de uma sprint específica compõem o *backlog da sprint*. Wang et al. (2017) propõem uma associação entre os Requisitos de Software (β) e o *Backlog da Sprint*, enquanto os Requisitos do Sistema (α) podem ser relacionados ao *Backlog do Produto*, conforme mostrado na Figura 3, representando os requisitos da Figura 2.

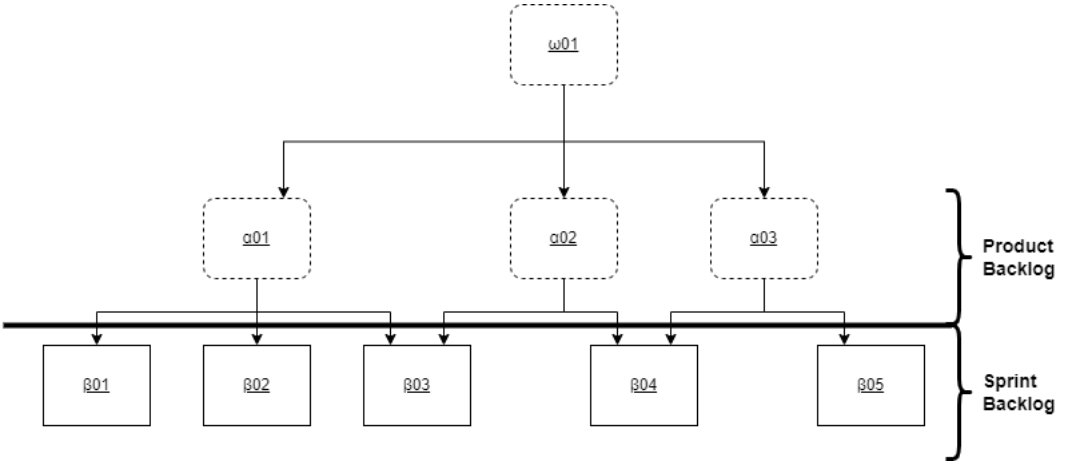

Figura 3: *Backlog do Produto, Backlog da Sprint, e o modelo tradicional de relacionamento entre requisitos (Amaral & Rosa, 2023b). Reproduzido com permissão da Springer Nature.*

No Scrum, a complexidade dos requisitos não deve ser estimada com base na quantidade de horas necessárias para sua implementação, mas sim por meio de uma estimativa relativa em comparação a todos os demais requisitos presentes no *backlog da sprint* (Schwaber & Sutherland, 2020). Esta complexidade relativa é medida em *Story Points* (Popli & Chauahn, 2015; Zahraoui & Janati Idrissi, 2015), utilizados pelas equipes de desenvolvimento para classificar o esforço de implementação dos itens do *backlog da sprint*, sendo atribuído o maior valor aos requisitos mais complexos e o menor valor aos menos complexos.

Os valores de complexidade (*Story Points*) são usualmente baseados na Escala de Fibonacci (Zahraoui & Janati Idrissi, 2015). No MAPIR-CSR, considera-se {*1, 2, 3, 5, 8, 13, 21*} como o conjunto de valores que podem ser atribuídos. O processo consiste em selecionar um valor intermediário (5 ou 8) como valor de referência para uma "tarefa normal", que não apresenta fatores de incerteza e não é considerada trivial pela equipe de desenvolvimento.

O próximo passo é selecionar um requisito de software que atenda a esta definição e atribuir a ele o valor selecionado. O método adota a abordagem proposta por Hannay et al. (2019), estabelecendo o valor *5* em *Story Points* para a "*tarefa normal*" (requisito de software β04) e reserva o valor de *21 Story Points* para tarefas definidas por Wang et al (2017) como "*Épicas*", ou seja, tarefas com alto nível de incerteza, que devem ser decompostas antes que o time de desenvolvimento possa prosseguir com sua implementação.

A partir destas definições, no exemplo em andamento, cada outro requisito de software é comparado ao β04, definido como a "*tarefa normal*", sendo atribuídos valores menores de *Story Points* se considerados menos complexos do que ele e valores maiores se considerados mais complexos. Os valores de todos os requisitos são então equilibrados para representar a comparação entre si (Jakobsen & Sutherland, 2009), resultando nos valores apresentados na Figura 4 e listados na matriz de rastreabilidade de requisitos, no Apêndice I (Tabela 8).

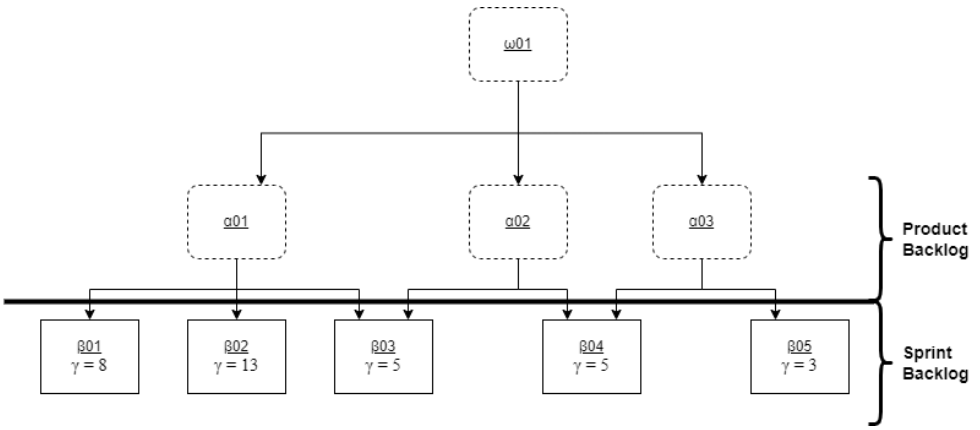

Figura 4: *Story Points (γ) atribuídos a cada Requisito de Software (Amaral & Rosa, 2023b). Reproduzido com permissão da Springer Nature.*

Conforme mostrado na Figura 4, o MAPIR-CSR parte da premissa de que todos os requisitos de software recebam uma pontuação inicial de *Story Points* quando são identificados, mesmo que venham a ser detalhados e revisados durante o planejamento de *sprints* futuras, com base no modelo das cerimônias de refinamento do *backlog do produto*, apresentadas no *Scrum Guide* (Schwaber & Sutherland, 2020), mesmo embora

esta prática não seja uma constante em projetos baseados em metodologias ágeis (Cao et al., 2010; Girma et al., 2019; Popli & Chauahn, 2015).

Uma vez que a relação de requisitos esteja definida e a complexidade estimada, rastrear a solicitação de mudança de um requisito de software (β) de volta ao requisito do sistema (α) que o originou permite identificar os outros requisitos de software também derivados dele e potencialmente impactados pela mudança.

A taxa de impacto potencial da mudança (ε) é definida como a soma dos *story points* de todos os requisitos de software potencialmente impactados (∑γ), dividida pelo número total de requisitos de software no projeto (n): $\varepsilon = (\sum\gamma) / n$.

Essa fórmula implica que o impacto potencial de uma mudança não seja o simples produto do número total de outros requisitos que podem ser impactados por ela, pois a taxa de impacto é reduzida proporcionalmente ao número total de requisitos de software no projeto, assumindo que um projeto com um conjunto menor de requisitos tende a adotar uma granularidade maior em cada requisito (Kim et al., 2010), enquanto um conjunto maior de requisitos permite uma granularidade menor, restringindo o impacto de cada mudança isolada (Asl & Kama, 2013).

Um dos resultados da aplicação do MAPIR-CSR é classificar todos os itens do *backlog da sprint* pelo seu impacto potencial em caso de mudança, conforme ilustrado na Tabela 3. Essa abordagem pode direcionar a discussão com o cliente para esclarecer o impacto de uma mudança, além de apoiar a estimativa do esforço necessário para sua implementação.

Tabela 3: *Requisitos de Software ordenados por sua Taxa de Impacto Potencial*

Requisitos de Software (β)	**Esforço em Story Points (γ)**	**Requisitos Potencialmente Impactados (δ)**	**Taxa de Impacto Potencial (ε)**
β03	5	{β01, β02, β03, β04}	6,2
β02	13	{β01, β02, β03}	5,2
β01	8	{β01, β02, β03}	5,2
β04	5	{β03, β04, β05}	2,6
β05	3	{β04, β05}	1,6

Na Tabela 3 são listados os requisitos do exemplo em curso, classificados por sua Taxa de Impacto Potencial (ε). Como exemplos, na primeira linha (β03), *(8+13+5+5) / 5 = 6,2* e na segunda linha (β02), *(8+13+5) / 5 = 5,2*.

Conforme mostrado na Tabela 3, apesar de β02 ser considerado o requisito mais complexo do sistema por ter o maior valor de *Story Points* (γ), ele não teve a maior Taxa de Impacto Potencial (ε), que pertence a β03. Outro resultado do método é a lista de Requisitos Potencialmente Impactados (δ) de cada requisito, com o objetivo de apoiar a análise exploratória de impacto.

O Capítulo 5 apresenta um cenário de aplicação do MAPIR-CSR, contendo a decomposição do *backlog do produto* e a relação de requisitos, bem como a avaliação de mudanças em diferentes sprints, com base nos requisitos descritos acima.

4.2 Protótipo de Software que Implementa o Processo de Aplicação do MAPIR-CSR

Foi desenvolvido um protótipo que implementa o processo de aplicação proposto. O objetivo principal do protótipo é automatizar o cálculo da taxa de impacto potencial de cada requisito de software, além gerar a lista de requisitos potencialmente impactados por mudanças de escopo.

O algoritmo (pseudocódigo) para implementação do MAPIR-CSR é apresentado no Apêndice 2 e consiste na lógica para i) recuperação dos Requisitos de Sistema relacionados ao Requisitos de Software que é objeto da mudança, ii) construção de uma lista com todos os Requisitos de Software relacionados a estes Requisitos de Sistema (lista de requisitos potencialmente afetados pela mudança) e iii) o acúmulo de seus *Story Points*, posteriormente dividido pela quantidade total de Requisitos de Software existentes no projeto, resultando na taxa de impacto potencial da mudança.

A partir deste algoritmo foi realizada a implementação de uma aplicação *Web*, com interface *HTML*, codificado na linguagem *PHP* e fazendo uso de um banco de dados relacional *MySQL*, cuja estrutura de tabelas é apresentada na Figura 5.

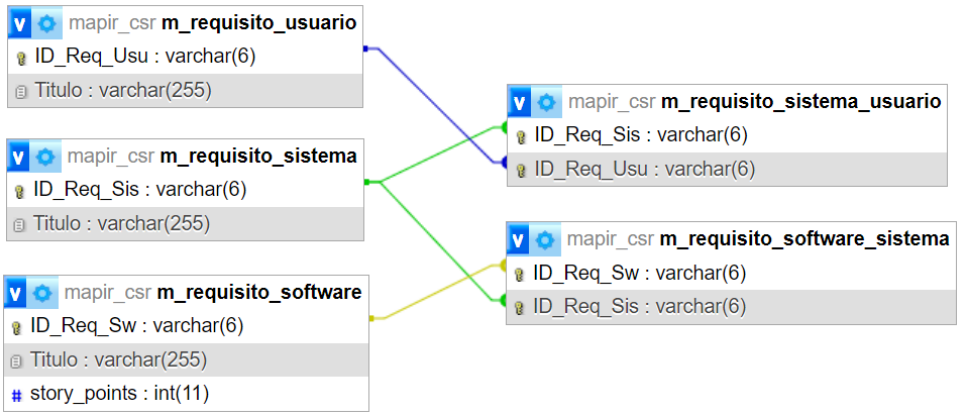

Figura 5: *Esquema de tabelas (MySQL) para implementação do MAPIR-CSR*

No protótipo desenvolvido, é possível realizar o cadastro e a manutenção dos requisitos de usuário, sistema e software de um projeto, bem como estabelecer relações entre eles e informar a quantidade de *Story Points* atribuída a cada requisito de software, conforme apresentado nas Figura 6 e 7.

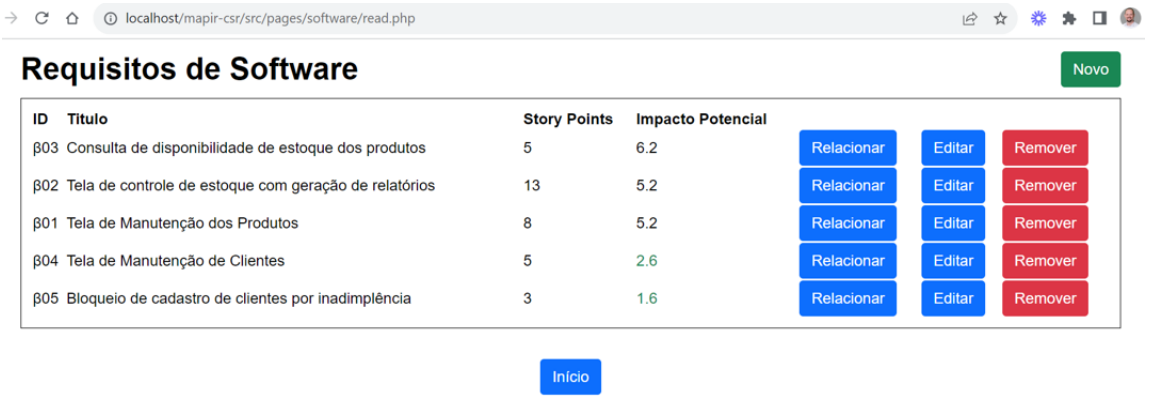

Figura 6: *Interface Principal do Protótipo Desenvolvido*

Além de listar os Requisito de Software cadastrados no sistema, a interface principal do sistema apresenta também a Taxa de Impacto Potencial calculada para cada requisito, adotando um código de cores no texto desta coluna visando refletir a estratégia para aceitação de mudanças conforme discutido na Seção 5.4.2 e ilustrado também na Figura 8.

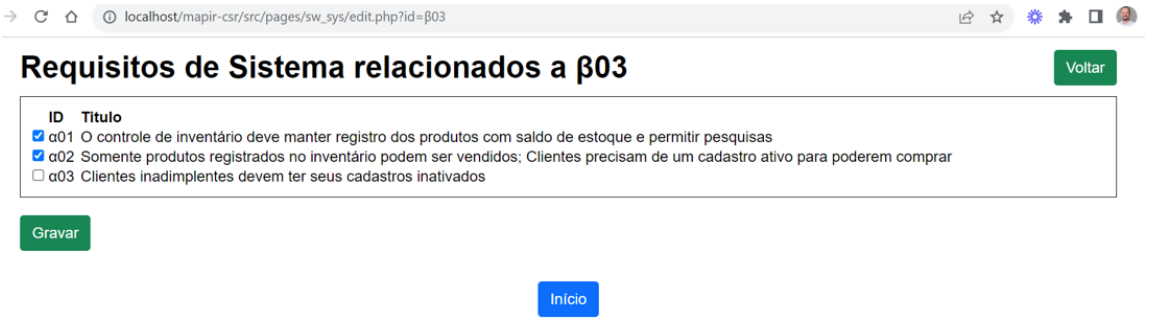

Figura 7: *Exemplo de Relacionamento entre Requisitos no Protótipo*

As informações presentes no cadastro do Requisito de Software, juntamente ao seu relacionamento com um ou mais Requisitos de Sistema, ilustrado na Figura 7, são as entradas necessárias ao processamento do MAPIR-CSR e por meio delas se determina a taxa de impacto potencial em caso de mudança em cada um dos requisitos, além da lista de requisitos que serão potencialmente impactados por esta mudança.

O Apêndice 3 apresenta o código-fonte escrito em *PHP* e *MySQL* para recuperação da lista de requisitos potencialmente impactados pela solicitação de mudança, além do cálculo de sua taxa de impacto potencial, informações apresentadas na Figura 8.

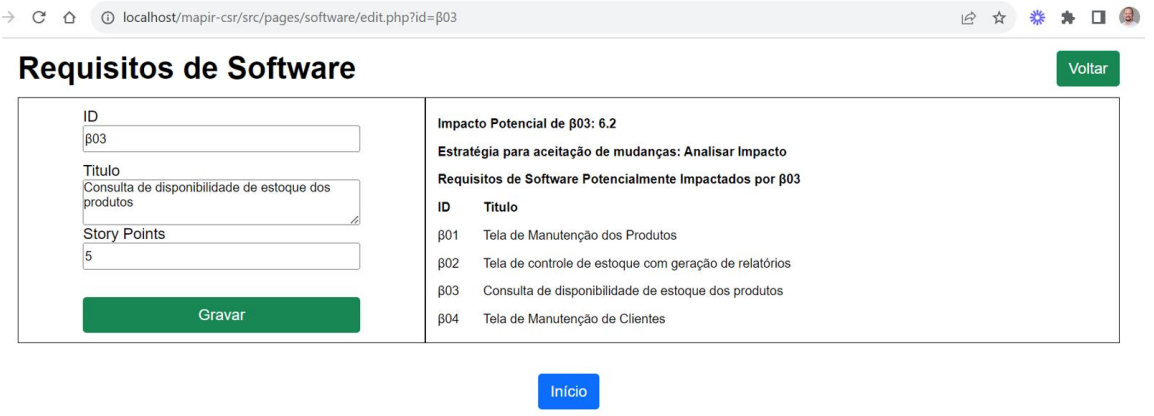

Figura 8: *Exibição do retorno do MAPIR-CSR*

Cabe destacar que a interface ilustrada acima apresenta também a estratégia para aceitação das mudanças, que é discutido na Seção 5.4.2 e exibido na Figura 8. O código-fonte completo para implementação do protótipo encontra-se disponível na Plataforma GitHub (Amaral & Rosa, 2023a).

5 CENÁRIO DE APLICAÇÃO DO MAPIR-CSR

Para demonstrar o uso de MAPIR-CSR e ilustrar melhor como ele pode apoiar o impacto das mudanças e a previsão de esforço em metodologias ágeis, o desenvolvimento de um sistema de comércio eletrônico foi emulado, com base nos requisitos detalhados no Apêndice 1. Todos os requisitos foram propostos por dois engenheiros de software seniores, cada um com mais de 20 anos de experiência na área. Um deles possui doutorado em engenharia de software e o outro possui certificações em Scrum e gerenciamento de projetos.

Os engenheiros de software assumiram várias personas (Cleland-Huang, 2013) para representar o cliente e os diferentes papéis de uma equipe com 3 desenvolvedores de software, além de um *Scrum Master* e um *Product Owner*, adotando o Framework Scrum para executar um conjunto de Sprints com eventos simulados e solicitações intencionais de mudança de escopo. A descrição de cada persona adotada neste processo está contida no Apêndice 5.

A partir disso, os resultados do método para cada *sprint* foram calculados por meio do protótipo desenvolvido com base no algoritmo descrito no Apêndice 2 para o processamento do MAPIR-CSR e tabulados em uma planilha do Microsoft Excel.

Como a definição de relacionamento de requisitos é uma tarefa complexa, baseada em fatores subjetivos (Benedicenti et al., 2016) e crítica para o sucesso do projeto (Ahmad et al., 2021), a experiência real em engenharia de requisitos foi a referência para estruturar os requisitos conforme apresentado na matriz de rastreabilidade de requisitos (Apêndice 1 - Tabela 8). A automação do mapeamento de relacionamento de requisitos está fora do escopo deste trabalho.

Desta forma, o cenário de aplicação (ciclo de desenvolvimento simulado) assume o objetivo de implementar um sistema de comércio eletrônico por meio do Framework Scrum (Schwaber & Sutherland, 2020), com uma equipe de desenvolvimento composta por 5 personas (3 desenvolvedores de software, 1 Scrum Master e 1 Product Owner), trabalhando com uma duração de *sprint* de 2 semanas. Esta composição combina o tamanho mínimo de equipe e a duração mínima de iteração recomendados pelo Scrum Guide (Schwaber & Sutherland, 2020).

Este cenário tem como objetivo representar um ambiente em constante mudança, no qual o tempo é um fator-chave para avaliar as mudanças de escopo (Ahmad et al., 2021). O papel de *Product Owner* adota uma abordagem de design centrada no ser humano (Garcia, 2015; Schön et al., 2017), presumindo entrevistas, observação de usuários, prototipagem e testes de usabilidade para apoiar as decisões do projeto, por meio de 3 sprints sequenciais descritas nas próximas subseções (5.1-3).

5.1 Sprint 1

Durante a sessão de planejamento simulada da primeira *sprint*, o *Product Owner* apresentou o *Backlog do Produto* para a equipe, composto pelos Requisitos do Sistema α01, α02 e α03, concordando com os desenvolvedores em uma decomposição inicial destes requisitos, resultando nos Requisitos de Software β01, β02, β03, β04 e β05, conforme mostrado na Figura 4.

Como decisão técnica, a equipe de desenvolvimento informou ao *Product Owner* que os requisitos deveriam ser agregados para descrever dois softwares interligados (um controle de inventário e um site de vendas), que compõem o Sistema de Comércio Eletrônico desejado, conforme apresentado na Figura 9.

Em resposta, o *Product Owner* solicitou à equipe de desenvolvimento que se concentrasse em entregar o controle de inventário o mais rápido possível, pois isto permitiria ao cliente começar a trabalhar e, consequentemente, agregaria valor ao projeto.

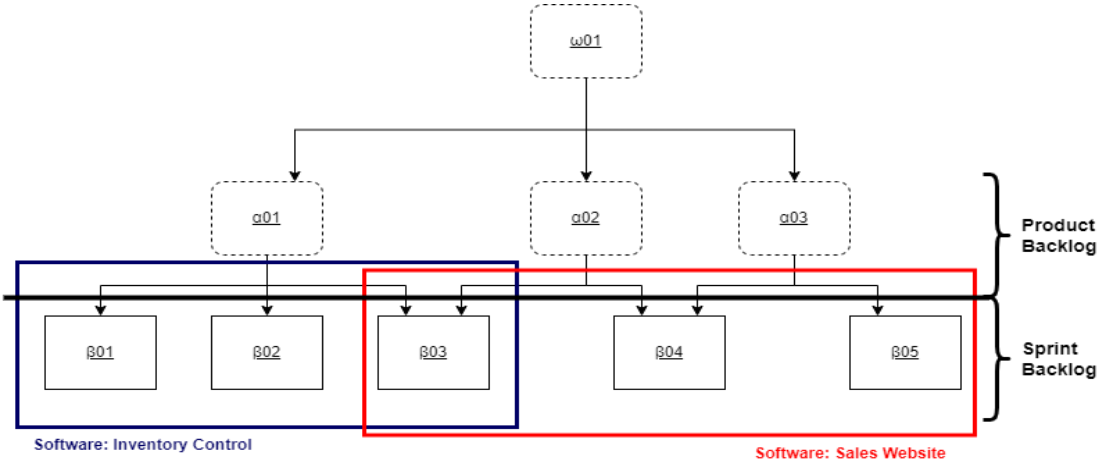

Figura 9: *Softwares que compõem o Sistema de Comércio Eletrônico (Amaral & Rosa, 2023b). Reproduzido com permissão da Springer Nature.*

O acordo entre o time de desenvolvimento e o *Product Owner* foi que, para a Sprint 1, o *Backlog da Sprint* seria composto pelos Requisitos de Software β01 e β02, totalizando 18 *Story Points* de complexidade, conforme apresentado na Tabela 3.

Uma vez que a equipe concordou com a estrutura dos requisitos e a estratégia geral para entregar o sistema, o impacto potencial de cada requisito pôde ser calculado usando o MAPIR-CSR, com base no conjunto de requisitos correntes.

A Tabela 4 apresenta as entradas e saídas resultantes no início da Sprint 1, ordenadas pela Taxa de Impacto Potencial, pois o método retorna a lista de Requisitos Potencialmente Impactados (δ) e a Taxa de Impacto Potencial (ε) de todos os requisitos de software presentes no projeto, independentemente de esses requisitos serem abordados ou não na *sprint* atual. Desta forma, na Tabela 4 são listados os requisitos de software e seus requisitos potencialmente impactados (δ) classificados pela Taxa de Impacto Potencial (ε).

Tabela 4: *Entradas e Saídas do MAPIR-CSR no início da Sprint 1*

Entradas			**Saídas**	
Requisitos de Software (β)	**Requisitos de Sistema Relacionados (α)**	**Story Points (γ)**	**Requisitos Potencialmente Impactados (δ)**	**Taxa de Impacto Potencial (ε)**
β03	α01, α02	5	{β01, β02, β03, β04}	6,2
β02	α01	13	{β01, β02, β03}	5,2
β01	α01	8	{β01, β02, β03}	5,2
β04	α02, α03	5	{β03, β04, β05}	2,6
β05	α03	3	{β04, β05}	1,6

Conforme apresentado na Tabela 4, o Requisito de Software β01 possui uma relação com o Requisito do Sistema α01, potencialmente impactando os requisitos (δ) β01 (ele mesmo), β02 e β03 em caso de uma solicitação de mudança de escopo. Este impacto potencial é resultado direto de β01, β02 e β03 compartilharem a relação com α01 e caracteriza a primeira saída do MAPIR-CSR, orientando a análise eventual de impacto, de forma qualitativa.

A segunda saída do método é a Taxa de Impacto Potencial (ε) de uma mudança de escopo relacionada a β01, que é obtido somando os *Story Points* (γ) atribuídos pela equipe de desenvolvimento a cada requisito potencialmente impactado (δ) de β01 (β01, β02 e β03), respectivamente 8, 13 e 5 *Story Points* (γ), totalizando 26 *Story Points* (∑γ). Esse valor resultante é então dividido pelo total de Requisitos de Software existentes (n), que são 5 neste momento (β01, β02, β03, β04 e β05), resultando em uma Taxa de Impacto Potencial (ε) de 5,2 (26/5) para β01, conforme descrito pela equação $\varepsilon = (\sum\gamma)/n$.

Ao conhecer a Taxa de Impacto Potencial (ε) de alterar cada requisito de software no projeto, a equipe de desenvolvimento tem um argumento quantitativo para negociar com o *Product Owner* eventuais compensações durante as sessões de planejamento das *sprints*, fornecendo um indicador claro de quão complexa a mudança tende a ser, uma vez que os *Story Points* são, por definição, a medida de complexidade definida pelos desenvolvedores (Schwaber & Sutherland, 2020; Sjoberg et al., 2010).

A mesma lógica é usada para calcular as saídas de todos os requisitos de software pelo MAPIR-CSR. Vale ressaltar que, neste exemplo, β03 e β04 apresentam relações com mais de um requisito do sistema. Isso significa que, para β03, a lista de requisitos

potencialmente impactados (δ) é {β01, β02, β03, β04}, e para β04, a lista é {β03, β04, β05}, conforme apresentado na Tabela 4.

Por fim, a Taxa de Impacto Potencial (ε) de β03 é obtida somando os *Story Points* (γ) atribuídos a cada um dos seus requisitos potencialmente impactados (δ), totalizando 31 *Story Points* (∑γ), que são divididos por 5, o total de Requisitos de Software (n) recebidos pelo método, resultando em uma Taxa de Impacto Potencial (ε) de 6,2.

5.2 Sprint 2

Durante a sessão de revisão da Sprint 1, o time de desenvolvimento apresentou o trabalho realizado ao cliente, que solicitou uma alteração no *Backlog do Produto*. O pedido foi a criação de uma nova tela para configurar a geração de relatórios. Este novo requisito foi adicionado ao *backlog* como o Requisito de Software β06 e nomeado "Tela de Configuração da Geração de Relatórios", sob uma relação com o Requisito do Sistema "α01", conforme apresentado na matriz de rastreabilidade de requisitos (Apêndice I - Tabela 8).

Posteriormente, na sessão de planejamento da Sprint 2, o time de desenvolvimento comprometeu-se com o *Product Owner* em entregar os Requisitos de Software β03 ("Tela de Manutenção de Produtos") e β06 ("Tela de Configuração da Geração de Relatórios") e concordou com uma pontuação de complexidade de 2 *Story Points* para o β06. Na Tabela 5, são apresentadas as entradas e saídas do MAPIR-CSR, considerando o novo requisito adicionado ao projeto.

Tabela 5: *Entradas e Saídas do MAPIR-CSR no início da Sprint 2*

Entradas			**Saídas**	
Requisitos de Software (β)	**Requisitos de Sistema Relacionados (α)**	**Story Points (γ)**	**Requisitos Potencialmente Impactados (δ)**	**Taxa de Impacto Potencial (ε)**
β03	α01, α02	5	{β01, β02, β03, β04, β06}	5,5
β02	α01	13	{β01, β02, β03, β06}	4,7
β01	α01	8	{β01, β02, β03, β06}	4,7
β06	α01	2	{β01, β02, β03, β06}	4,7
β04	α02, α03	5	{β03, β04, β05}	2,2
β05	α03	3	{β04, β05}	1,3

A partir da adição de um novo requisito ao projeto, o método redefiniu a taxa de impacto potencial de cada requisito de software (β), refletindo a mudança na granularidade dos requisitos. Como o objetivo da mudança foi adicionar um novo requisito e não de modificar um existente, não há uma análise formal de impacto da mudança a ser conduzida, mas o MAPIR-CSR antecipa a visibilidade sobre o impacto potencial das mudanças para o novo conjunto de requisitos.

Todos os Requisitos de Software que compartilham uma relação de dependência com o mesmo Requisito do Sistema que β06 (β01, β02 e β03) também tiveram sua lista de Requisitos Potencialmente Impactados (δ) atualizada pelo método.

5.3 Sprint 3

Após a conclusão da Sprint 2, o cliente solicitou ao *Product Owner* uma nova mudança no escopo do projeto, indicando que a tela de manutenção de produtos precisaria contemplar campos e fórmulas adicionais para cumprir com uma nova legislação vigente, o que resultou em uma mudança solicitada por força de lei.

Com base na lista de requisitos potencialmente impactados (δ) do MAPIR-CSR, a equipe de desenvolvimento foi capaz de identificar que a alteração do requisito β01 (tela de manutenção de produtos) poderia resultar em impactos nos requisitos β01 (ele mesmo), β02, β03 e β06. Com essa orientação, durante a sessão de planejamento da Sprint 3, a equipe concordou em elevar a pontuação de complexidade de β01 para 13, assumindo que seriam necessários esforços adicionais de análise e atenção para evitar impactos nos outros 3 requisitos listados (β02, β03 e β06).

O *Product Owner*, por sua vez, atualizou a matriz de rastreabilidade de requisitos, presente no Apêndice I (Tabela 8), e então verificou novamente as especificações de β02, β03 e β06 para garantir que nenhuma mudança adicional fosse necessária nestes requisitos, concordando com a equipe sobre a estratégia de implementar a mudança em β01 e seguir com o desenvolvimento de β04 e β05 (não impactados pela mudança) na Sprint 3.

A Tabela 6 apresenta as alterações nos resultados do MAPIR-CSR após as mudanças acordadas no escopo do projeto.

Tabela 6: *Entradas e Saídas do MAPIR-CSR no início da Sprint 3*

Entradas			**Saídas**	
Requisitos de Software (β)	**Requisitos de Sistema Relacionados (α)**	**Story Points (γ)**	**Requisitos Potencialmente Impactados (δ)**	**Taxa de Impacto Potencial (ε)**
β03	α01, α02	5	{β01, β02, β03, β04, β06}	6,3
β02	α01	13	{β01, β02, β03, β06}	5,5
β01	α01	13	{β01, β02, β03, β06}	5,5
β06	α01	2	{β01, β02, β03, β06}	5,5
β04	α02, α03	5	{β03, β04, β05}	2,2
β05	α03	3	{β04, β05}	1,3

Como nenhum requisito novo foi adicionado ao projeto, o cenário na Sprint 3 é diferente da Sprint 2, com β04 e β05 não apresentando alterações em suas respectivas Taxas de Impacto Potencial (ε), pois eles não têm relação com o requisito sujeito à mudança solicitada (β01).

A lista de requisitos potencialmente afetados (δ) não sofreu alterações em todos os requisitos, pois a mudança solicitada apenas refletiu no impacto potencial de β01, β02, β03 e β06.

5.4 Discussão sobre os resultados

O MAPIR-CSR apoiou as análises de impacto qualitativa e quantitativa das solicitações de mudanças de escopo, fornecendo à equipe de desenvolvimento visibilidade sobre quais outros requisitos poderiam ser afetados pela mudança, por meio da saída "Requisitos Potencialmente Impactados (δ)" (análise qualitativa) e sobre o quão complexa a mudança solicitada tenderia a ser, por meio da saída "Taxa de Impacto Potencial (ε)" para cada requisito (análise quantitativa).

5.4.1 Considerações sobre a diferença entre alterar um requisito existente e adicionar um novo requisito ao projeto

A Tabela 7 apresenta uma visão de como a taxa de impacto potencial (ε) de cada requisito mudou ao longo das sprints, fornecendo uma percepção interessante sobre a Sprint 2, quando os requisitos tiveram sua taxa de impacto reduzida devido à mudança na granularidade do projeto, acomodando um requisito recém-adicionado (β06).

Também foi observado que uma mudança em um requisito específico impacta principalmente os requisitos diretamente relacionados a ele, como aconteceu na Sprint 3, enquanto a adição de novos requisitos ao escopo do projeto afeta todo o conjunto de requisitos do projeto, como na Sprint 2, exigindo uma análise qualitativa mais abrangente para validar que nenhuma mudança adicional resulte desta ação.

Com estes recursos em mãos, o *Product Owner* também pode determinar em quais requisitos evitar mudanças, indicando quando buscar soluções técnicas alternativas com a equipe de desenvolvimento para atender a uma solicitação de mudança do cliente. Um exemplo é o requisito β03, que teve a maior pontuação na taxa de impacto potencial (ε) durante todo o projeto, o que significa que seria o requisito mais impactante se for alterado.

Tabela 7: *Taxa de Impacto Potencial por Requisito de Software (ε) em cada Iteração*

Iteração	**Taxa de Impacto Potencial por Requisito de Software (ε)**					
	β01	**β02**	**β03**	**β04**	**β05**	**β06**
Sprint 1	5,2	5,2	6,2	2,6	1,6	N/A
Sprint 2	4,7	4,7	5,5	2,2	1,3	4,7
Sprint 3	5,5	5,5	6,3	2,2	1,3	5,5

5.4.2 Considerações sobre a estratégia para aceitação de solicitações de mudança

Levando-se em conta que a pontuação de 5 *Story Points* foi definida como valor de referência para uma "tarefa normal", *i.e.*, que não apresenta fatores de incerteza e não é considerada trivial pela equipe de desenvolvimento, é importante destacar que qualquer requisito mais simples que ela receberá uma pontuação de no máximo 3 *Story Points*, enquanto qualquer requisito mais complexo receberá uma pontuação mínima de 8, segundo a sequência de Fibonacci.

Com base no cenário de aplicação, é, portanto, viável estabelecer faixas aceitáveis para a taxa de impacto potencial calculada pelo MAPIR-CSR dentro deste intervalo entre 3 e 8.

Uma taxa de impacto maior do que 8 possivelmente indica um requisito que está interligado a vários outros requisitos de alta complexidade, representando um risco para o projeto, indicando que a solicitação de mudança deve ser recusada imediatamente pelo time de desenvolvimento, de modo que o *Product Owner* deve negociar outra solução para o problema junto ao cliente.

Por outro lado, uma taxa menor do que 3 tende a ser o resultado de uma mudança em um requisito relacionado a vários requisitos de baixa complexidade, sendo sua mudança potencialmente menos arriscada, de modo que o time de desenvolvimento pode aceitar a mesma sem maiores preocupações.

Desta forma, a adoção do MAPIR-CSR possibilita também a implementação de uma estratégia de aceitação das solicitações de mudança, cuja implementação é ilustrada no Apêndice 4 e apresentada no protótipo construído para implementação do método, onde a estratégia de aceitação é definida como (i) “Aceitar”, nos casos em que a taxa de impacto potencial seja menor que 3, (ii) “Recusar”, nos casos em que a taxa de impacto potencial seja maior que 8 ou (iii) “Analisar Impacto”, nos demais casos.

6 CONCLUSÕES

Neste trabalho, foi apresentada uma abordagem inovadora, que leva em consideração aspectos específicos de projetos baseados em metodologias ágeis, com o objetivo de apoiar equipes de desenvolvimento na análise de impacto e estimativa de esforço de solicitações de mudança de escopo.

A partir da aplicação do método proposto, os principais resultados do método foram: (i) forneceu uma taxa de impacto potencial para mudanças de escopo em projetos baseados em metodologias ágeis e uma lista de requisitos potencialmente impactados pela mudança; (ii) definiu e aplicou uma métrica para classificar os requisitos de software com base em seu impacto potencial, apoiando a estimativa de esforço e análise de impacto; (iii) possibilitou a identificação de uma faixa de valores para determinar o risco de aceitar uma solicitação de mudança de escopo; e (iv) possibilitou o desenvolvimento de um protótipo de software para automatizar cálculos e validar o processo de aplicação do método.

6.1 Limitações e Trabalhos Futuros

Durante o desenvolvimento deste trabalho, algumas limitações foram encontradas. Por exemplo, i) a coleta das informações sobre os requisitos não é automatizada, ii) faz-se necessário um refinamento de requisitos no momento de sua criação, iii) o processo de desenvolvimento de software foi emulado por meio de personas e não se baseou em dados de um projeto real e iv) o uso do método em projetos baseados em outros ciclos de vida e processos de desenvolvimento não foi abordado como parte da pesquisa.

Como trabalhos futuros, abordando a primeira limitação (i), o método poderia ser usado de maneira combinada (ou integrado) com ferramentas automatizadas voltadas ao rastreamento de tarefas (*e.g.*, Jira, Trello), para calcular e apresentar dinamicamente os resultados do MAPIR-CSR como indicadores do projeto para a equipe de desenvolvimento, permitindo a parametrização do intervalo de valores e código de cores para definição da estratégia de aceitação de riscos. Este trabalho poderia resultar em componentes de software com uso destinado a empresas de desenvolvimento de sistemas

e sua implementação e testes podem resultar na publicação de um artigo científico complementar a esta dissertação.

Com relação à limitação ii), poderia ser avaliada a modificação do método para desconsiderar requisitos que não tenham sido refinados, em um comparativo com os resultados tabulados neste trabalho, originando uma pesquisa completa e aprofundada a respeito de diferentes cenários de aplicação do MAPIR-CSR. A definição de diferentes estratégias de aceitação de solicitações de mudança de escopo, com base na faixa de valores da Taxa de Impacto Potencial retornada pelo método, também pode ser aprofundada neste cenário.

A limitação iii) poderia ser compensada com o uso de dados anônimos de projetos reais compartilhados por empresas ou órgãos públicos, validando a aplicação do método em um conjunto de diferentes projetos com variados graus de mudança em seus requisitos. Esta atividade também poderia resultar na publicação de um artigo científico complementar a esta dissertação, além de pautar a discussão sobre cenários onde o tema pode ser aprofundado.

Por fim, abordando a limitação iv), poderia ser conduzido um estudo comparativo dos resultados do MAPIR-CSR em projetos baseados em diferentes metodologias de desenvolvimento de software, sejam elas ágeis ou preditivas, gerando uma revisão de literatura relevante para trabalhos futuros e um artigo científico derivado desta dissertação.

6.2 Resultados da Pesquisa

Como resultados efetivos da pesquisa, cabe destacar um artigo completo (*full paper*) publicado em conferência internacional, além de um protótipo de software que foi registrado junto ao INPI, detalhados a seguir.

1) **Artigo do Método** (Amaral & Rosa, 2023b) – Incluso na área temática "*Human Interface and the Management of Information*" da *25th International Conference on Human-Computer Interaction* (HCII-2023) (Qualis CC B1), este artigo apresentou o núcleo da proposta de MAPIR-CSR e foi publicado como capítulo do livro *Human Interface and the Management*

of Information, no volume 14016 da série *Lecture Notes in Computer Science*, publicada pela Springer Nature. O *abstract* do artigo é apresentado a seguir:

Abstract. The main issues in impact analysis of scope changes are the need for identifying other requirements potentially affected by the change and determining how complex is the change. These challenges are potentialized into projects based on agile methodologies, characterized by broad communication with the customer, which leads to more opportunities for change requests during the software development life cycle. We propose a new method in which agile method-ologies can benefit from requirement traceability and effort prediction on scope changes. By applying the method to a software development project, the results are synthesized into indicators presenting how scope changes realized through different development iterations impact other software requirements and how the complexity of the requirement being changed can be used to determine the effort estimate. The main contributions of our work are (i) method aimed to provide a potential impact rate on scope changes in agile methodologies-based projects; (ii) metric for ranking software requirements based on their potential impact, supporting effort prediction and impact analysis; (iii) value range to determine the risk of accepting a scope change request; and (iv) discussion on the results from applying the approach in an agile software development scenario. Our proposal is intended to be used by software development teams in the context of agile projects.

2) **Protótipo de Software** – Com o objetivo automatizar cálculos e validar o processo de aplicação do método, um protótipo de software foi desenvolvido. O código-fonte encontra-se disponível na plataforma GitHub (Amaral & Rosa, 2023a). O protótipo foi registrado junto ao INPI sob o título "Protótipo de Sistema para Avaliar o Impacto Potencial das Mudanças nos Requisitos de Software nos Projetos baseados em Metodologias Ágeis", por meio do processo número BR512023002978-5 em 04/10/2023. As seguintes

linguagens de programação foram utilizadas: i) HTML, para a interface do sistema, conforme apresentado no Apêndice 4 e ii) PHP, para a implementação da lógica de processamento do método, em conjunto a um banco de dados MySQL, conforme apresentado no Apêndice 3. As funcionalidades providas pela aplicação são: (i) manutenção do cadastro de requisitos de usuários, (ii) manutenção do cadastro de requisitos de sistema, (iii) manutenção do cadastro de requisitos de software, (iv) relacionamento entre requisitos de sistema e requisitos de usuário, (v) relacionamento entre requisitos de software e requisitos de sistema, (vi) exibição da taxa de impacto potencial em caso de alteração de um requisito de software, calculada de acordo com o método MAPIR-CSR, (vii) exibição da estratégia para aceitação da solicitação de mudança no requisito de software e (viii) lista de requisitos de software potencialmente impactos pela mudança de um requisitos específico.

BIBLIOGRAFIA

Abou Khalil, Z. (2019). Studying the Impact of Policy Changes on Bug Handling Performance. *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 590–594. https://doi.org/10.1109/ICSME.2019.00093

Ahmad, J., Khan, A. W., & Khan, H. U. (2021). Role of Critical Success Factors in Offshore Quality Requirement Change Management Using SLR. *IEEE Access*, *9*, 99680–99698. https://doi.org/10.1109/ACCESS.2021.3096663

Amaral, A., & Nicoletti, M. do C. (2021). Considerações sobre Metodologias Direcionadas a Projeto e Desenvolvimento de Sistemas Computacionais. *Anais Do WCF, Vol 8 - XVII WCF*, 1–6.

Amaral, A., & Rosa, F. de F. (2023a). *MAPIR-CSR*. https://github.com/angelo-amaral/mapir-csr

Amaral, A., & Rosa, F. de F. (2023b). Method for Assessing the Potential Impact of Changes in Software Requirements of Agile Methodologies Based Projects. In Y. Mori Hirohiko and Asahi (Ed.), *Human Interface and the Management of Information* (pp. 3–21). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-35129-7_1

Asl, M. H., & Kama, N. (2013). A Change Impact Size Estimation Approach during the Software Development. *2013 22nd Australian Software Engineering Conference*, 68–77. https://doi.org/10.1109/ASWEC.2013.18

Ayed, H., Vanderose, B., & Habra, N. (2014). Supported Approach for Agile Methods Adaptation: An Adoption Study. *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, 36–41. https://doi.org/10.1145/2593812.2593820

Barton, B., & Campbell, E. (2007). Implementing a Professional Services Organization Using Type C Scrum. *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 275a–275a. https://doi.org/10.1109/HICSS.2007.265

Basri, S., Kama, N., Haneem, F., & Ismail, S. A. (2016). Predicting Effort for Requirement Changes during Software Development. *Proceedings of the Seventh Symposium on Information and Communication Technology*, 380–387. https://doi.org/10.1145/3011077.3011096

Basri, S., Kama, N., Sarkan, H. M., Adli, S., & Haneem, F. (2016). An Algorithmic-Based Change Effort Estimation Model for Software Development. *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, 177–184. https://doi.org/10.1109/APSEC.2016.034

Bass, J. M. (2014). Scrum Master Activities: Process Tailoring in Large Enterprise Projects. *2014 IEEE 9th International Conference on Global Software Engineering*, 6–15. https://doi.org/10.1109/ICGSE.2014.24

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., & others. (2001). *Agile Manifesto*. The Agile Alliance, www. agilemanifesto. org.

Benedicenti, L., Cotugno, F., Cianfrini, P., Messina, A., Pedrycz, W., Sillitti, A., & Succi, G. (2016). Applying Scrum to the Army - A Case Study |

IEEE Conference Publication | IEEE Xplore. *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 725–727. https://ieeexplore.ieee.org/document/7883385

Campfield, M., Pingleton, M., McNally, S. T., Samuel, T. K., Packard, M. A., & Maiden, T. (2013). The XSEDE Ticket System: From Concept to Implementation. *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery*. https://doi.org/10.1145/2484762.2484792

Cao, L., Ramesh, B., & Abdel-Hamid, T. (2010). Modeling Dynamics in Agile Software Development. *ACM Trans. Manage. Inf. Syst.*, *1*(1). https://doi.org/10.1145/1877725.1877730

Cleland-Huang, J. (2013). Meet Elaine: A Persona-Driven Approach to Exploring Architecturally Significant Requirements. *IEEE Software*, *30*(4), 18–21. https://doi.org/10.1109/MS.2013.80

Díaz, J., Pérez, J., Garbajosa, J., & Yagüe, A. (2013). Change-Impact Driven Agile Architecting. *2013 46th Hawaii International Conference on System Sciences*, 4780–4789. https://doi.org/10.1109/HICSS.2013.127

Duarte, A. P., Herrera, V. A. S., Herrera, R. S., & Melendez, G. Y. T. (2011). Using IBM Rational Application Developer to Develop Enterprise Applications with Java EE, Dojo Server Faces and Interconnecting Them Using SOA. *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, 373–374.

Garcia, C. K. Z. (2015). Human-Centered Product Owner: How Human-Centered Design Can Sharpen Scrum Methodology. In C. Stephanidis (Ed.), *HCI International 2015 - Posters' Extended Abstracts* (pp. 409–413). Springer International Publishing.

Gary, K. A., & Xavier, S. (2015). Agile learning through continuous assessment. *2015 IEEE Frontiers in Education Conference (FIE)*, 1–4. https://doi.org/10.1109/FIE.2015.7344278

Girma, M., Garcia, N. M., & Kifle, M. (2019). Agile Scrum Scaling Practices for Large Scale Software Development. *2019 4th International Conference on Information Systems Engineering (ICISE)*, 34–38. https://doi.org/10.1109/ICISE.2019.00014

Habib, B., & Romli, R. (2021). A Systematic Mapping Study on Issues and Importance of Documentation in Agile. *2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS)*, 198–202. https://doi.org/10.1109/ICSESS52187.2021.9522254

Hamed, A. M. M., & Abushama, H. (2013). Popular agile approaches in software development: Review and analysis. *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, 160–166. https://doi.org/10.1109/ICCEEE.2013.6633925

Hannay, J. E., Benestad, H. C., & Strand, K. (2019). Agile Uncertainty Assessment for Benefit Points and Story Points. *IEEE Software*, *36*(4), 50–62. https://doi.org/10.1109/MS.2018.2875845

Heath, F. (2020). *Managing Software Requirements the Agile Way: Bridge the gap between software requirements and executable specifications to deliver successful projects*. Packt Publishing Ltd.

Jakobsen, C. R., & Sutherland, J. (2009). Scrum and CMMI Going from Good to Great. *2009 Agile Conference*, 333–337. https://doi.org/10.1109/AGILE.2009.31

Jonkers, R. K., & Eftekhari Shahroudi, K. (2021). A Design Change, Knowledge, and Project Management Flight Simulator for Product and Project Success. *IEEE Systems Journal*, *15*(1), 1130–1139. https://doi.org/10.1109/JSYST.2020.3006747

Kama, N., & Azli, F. (2012). A Change Impact Analysis Approach for the Software Development Phase. *2012 19th Asia-Pacific Software Engineering Conference*, *1*, 583–592. https://doi.org/10.1109/APSEC.2012.89

Kaur, K., Khurana, M., & Manisha. (2021). Impact of Agile Scrum Methodology on Time to Market and Code Quality – A Case Study. *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 1673–1678. https://doi.org/10.1109/ICAC3N53548.2021.9725375

Kim, T., Kim, K., & Kim, W. (2010). An Interactive Change Impact Analysis Based on an Architectural Reflexion Model Approach. *2010 IEEE 34th Annual Computer Software and Applications Conference*, 297–302. https://doi.org/10.1109/COMPSAC.2010.37

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. *Keele University Technical Report TR/SE-0401*, 1–26.

Krzanik, L., Rodriguez, P., Simila, J., Kuvaja, P., & Rohunen, A. (2010). Exploring the Transient Nature of Agile Project Management Practices. *2010 43rd Hawaii International Conference on System Sciences*, 1–8. https://doi.org/10.1109/HICSS.2010.204

Malhotra, R., & Chug, A. (2016). Comparative analysis of agile methods and iterative enhancement model in assessment of software maintenance. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1271–1276.

Manisha, Khurana, M., & Kaur, K. (2021). Impact of Agile Scrum Methodology on Team's Productivity and Client Satisfaction – A Case Study. *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 1686–1691. https://doi.org/10.1109/ICAC3N53548.2021.9725505

Murphy, B., & Williams, L. (2013). To branch or not to branch that is the question. *2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 55. https://doi.org/10.1109/ISSREW.2013.6688869

Ochoa, O., Towhidneiad, M., Wilson, T., Pembridze, J., Bowen, E., & Castro, C. (2021). Adopting Agility in Academia through Pilot Projects. *2021 IEEE Frontiers in Education Conference (FIE)*, 1–5. https://doi.org/10.1109/FIE49875.2021.9637458

Popli, R., & Chauahn, N. (2015). Managing uncertainty of story-points in Agile software. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1357–1361.

Pressman, R. S., & Maxim, B. R. (2021). *Engenharia de software* (9th ed.). McGraw Hill Brasil.

Project Management Institute. (2017). *Agile Practice Guide*. Project Managament Institute, Inc.

Raj, G., Yadav, K., & Jaiswal, A. (2015). Emphasis on testing assimilation using cloud computing for improvised agile SCRUM framework. *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, 219–225. https://doi.org/10.1109/ABLAZE.2015.7154995

Rajlich, V. (2010). Teaching undergraduate software engineering. *2010 IEEE International Conference on Software Maintenance*, 1–2. https://doi.org/10.1109/ICSM.2010.5609587

Rajlich, V. (2013). Teaching developer skills in the first software engineering course. *2013 35th International Conference on Software Engineering (ICSE)*, 1109–1116. https://doi.org/10.1109/ICSE.2013.6606661

Rubasinghe, I., Meedeniya, D., & Perera, I. (2018). Traceability Management with Impact Analysis in DevOps based Software Development. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1956–1962. https://doi.org/10.1109/ICACCI.2018.8554399

Schön, E.-M., Thomaschewski, J., & Escalona, M. J. (2017). Agile Requirements Engineering: A systematic literature review. *Computer Standards & Interfaces*, *49*, 79–91.

Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*.

Silva, A. G. F., Andrade, W. L., & Alves, E. L. G. (2018). A Study on the Impact of Model Evolution in MBT Suites. *Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing*, 49–56. https://doi.org/10.1145/3266003.3266009

Sjoberg, D. I. K., Odberg, E., & Warlo, B. (2010). The Challenge of Assessing and Controlling Complexity in a Large Portfolio of Software Systems. *Proceedings of the 11th International Conference on Product Focused Software*, 71–74. https://doi.org/10.1145/1961258.1961276

Sletholt, M. T., Hannay, J., Pfahl, D., Benestad, H. C., & Langtangen, H. P. (2011). A Literature Review of Agile Practices and Their Effects in Scientific Software Development. *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering*, 1–9. https://doi.org/10.1145/1985782.1985784

Sommerville, I. (2011). *Engenharia de software* (9th ed.). Pearson Prentice Hall.

Taromirad, M., & Paige, R. F. (2012). Agile Requirements Traceability Using Domain-Specific Modelling Languages. *Proceedings of the 2012 Extreme Modeling Workshop*, 45–50. https://doi.org/10.1145/2467307.2467316

Tufano, M., Sajnani, H., & Herzig, K. (2019). Towards Predicting the Impact of Software Changes on Building Activities. *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, 49–52. https://doi.org/10.1109/ICSE-NIER.2019.00021

Utz, W. (2019). Design of a Domain-Specific Metamodel for Industrial Business Process Management. *2019 8th International Congress on*

Advanced Applied Informatics (IIAI-AAI), 821–826. https://doi.org/10.1109/IIAI-AAI.2019.00167

Vasic, M., Parvez, Z., Milicevic, A., & Gligoric, M. (2017). File-Level vs. Module-Level Regression Test Selection for .NET. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 848–853. https://doi.org/10.1145/3106237.3117763

Wang, C., Xie, X., Liang, P., & Xuan, J. (2018). Multi-Perspective Visualization to Assist Code Change Review. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2017-December*, 564–569. https://doi.org/10.1109/APSEC.2017.66

Wang, Y., Bogicevic, I., & Wagner, S. (2017). A Study of Safety Documentation in a Scrum Development Process. *Proceedings of the XP2017 Scientific Workshops*. https://doi.org/10.1145/3120459.3120482

Wiegers, K. E. (2009). *More about software requirements: thorny issues and practical advice* (2nd ed.). Microsoft Press.

Wiegers, K. E., & Beatty, J. (2013). *Software requirements* (3rd ed.). Pearson Education.

Włodarski, R., Poniszewska-Marańda, A., & Falleri, J.-R. (2020). Comparative Case Study of Plan-Driven and Agile Approaches in Student Computing Projects. *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 1–6. https://doi.org/10.23919/SoftCOM50211.2020.9238196

Zahraoui, H., & Janati Idrissi, M. A. (2015). Adjusting story points calculation in scrum effort & time estimation. *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, 1–8. https://doi.org/10.1109/SITA.2015.7358400

Zapotecas-Martinez, S., Garcia-Nájera, A., & Cervantes, H. (2020). Multi-Objective Optimization in the Agile Software Project Scheduling Using Decomposition. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 1495–1502. https://doi.org/10.1145/3377929.3398146

APÊNDICE 1 – DETALHAMENTO DOS REQUISITOS DE USUÁRIO, SISTEMA E SOFTWARE APRESENTADOS

Tabela 8: *Matriz de Rastreabilidade dos Requisitos*

Tipo	Id	Descrição	Story Points	Relacionamento	Criação
Requisito de Usuário	ω01	Criar um e-Commerce com site de vendas e controle de inventário.	N/A	N/A	Sprint 1
Requisito de Sistema	α01	O controle de inventário deve manter registro dos produtos com saldo de estoque e permitir pesquisas.	N/A	ω01	Sprint 1
Requisito de Sistema	α02	Somente produtos registrados no inventário podem ser vendidos; Clientes precisam de um cadastro ativo para poderem comprar.	N/A	ω01	Sprint 1
Requisito de Sistema	α03	Clientes inadimplentes devem ter seus cadastros inativados.	N/A	ω01	Sprint 1
Requisito de Software	β01	Tela de Manutenção dos Produtos.	8 (13 na Sprint 3)	α01	Sprint 1
Requisito de Software	β02	Tela de controle de estoque com geração de relatórios.	13	α01	Sprint 1
Requisito de Software	β03	Consulta de disponibilidade de estoque dos produtos	5	α01, α02	Sprint 1
Requisito de Software	β04	Tela de Manutenção de Clientes.	5	α02, α03	Sprint 1
Requisito de Software	β05	Bloqueio de cadastro de clientes por inadimplência.	3	α03	Sprint 1
Requisito de Software	β06	Tela de configuração da geração de relatórios.	2	α01	Sprint 2

APÊNDICE 2 – PSEUDOCÓDIGO (ALGORITMO) PARA IMPLEMENTAÇÃO DO MAPIR-CSR

```
//Retorno da Lista de Requisitos Potencialmente Impactados pela Mudança
DEFINIR LISTA REQs
//Retorno da Taxa de Impacto Potencial da Mudança
DEFINIR VARIAVEL Impacto
//Story Points Acumulados
DEFINIR VARIAVEL SP
//Quantidade de Requisitos de Software Existentes no Projeto
DEFINIR VARIAVEL QTDRSw
//Requisito de Software que será modificado
DEFINIR OBJETO RSw
//Requisito de Sistema que relacionado ao Requisito de Software modificado
DEFINIR OBJETO RSi
//Objeto Auxiliar
DEFINIR OBJETO RSwAUX

SP = 0
QTDRSw = 0

PARA CADA Requisito de Software no Projeto, INCREMENTAR QTDRSw em 1

ATRIBUIR Requisito de Software Modificado a RSw

PARA CADA Requisito de Sistema relacionado a RSw, RECUPERAR RSi
        PARA CADA Requisito de Software relacionado a RSi, RECUPERAR RSwAUX
                ACUMULAR EM SP os Story Points de RSwAUX
                //Lista de Requisitos Potencialmente Impactados
                ADICIONAR RSwAUX à lista REQs
        FIM
FIM

//Taxa de Impacto Potencial
Impacto = SP / QTDRSw

RETORNAR Impacto, REQs
```

APÊNDICE 3 – CÓDIGO EM PHP E MYSQL PARA OBTER A TAXA DE IMPACTO POTENCIAL E A LISTA DE REQUISITOS POTENCIALMENTE IMPACTADOS

```
function findImpactoReqSwDb($conn, $id) {

	$id = mysqli_real_escape_string($conn, $id);
	$ImpactoReqSw;

	$sql = "SELECT Round((a.story_points +
		(Select
		sum(e.story_points)
		From
		m_requisito_software_sistema b INNER JOIN
		m_requisito_sistema c on c.ID_Req_Sis = b.ID_Req_Sis INNER JOIN
		m_requisito_software_sistema d on d.ID_Req_Sis = c.ID_Req_Sis INNER JOIN
		m_requisito_software e ON e.ID_Req_Sw = d.ID_Req_Sw
					and e.ID_Req_Sw <> a.ID_Req_Sw
		Where
		b.ID_Req_Sw = a.ID_Req_Sw))
		/
		(select COUNT(ID_Req_Sw) from m_requisito_software), 1) as impacto_potencial
		FROM
		m_requisito_software a
		WHERE
		a.ID_Req_Sw = ?";

	$stmt = mysqli_stmt_init($conn);

	if(!mysqli_stmt_prepare($stmt, $sql))
		exit('SQL error');

	mysqli_stmt_bind_param($stmt, 's', $id);
	mysqli_stmt_execute($stmt);

	$ImpactoReqSw = mysqli_fetch_assoc(mysqli_stmt_get_result($stmt));

	mysqli_close($conn);
	return $ImpactoReqSw;
}


function findReqSwRelDb($conn, $id) {

	$id = mysqli_real_escape_string($conn, $id);

	$ReqSwRel = [];

	$sql = "SELECT DISTINCT
		d.ID_Req_Sw,
		d.Titulo
		FROM
		m_requisito_software as a INNER JOIN
		m_requisito_software_sistema as b ON b.ID_Req_Sw = a.ID_Req_Sw INNER JOIN
```

```
                m_requisito_software_sistema as c  ON c.ID_Req_Sis = b.ID_Req_Sis INNER JOIN
                m_requisito_software as d ON d.ID_Req_Sw = c.ID_Req_Sw
                WHERE
                a.ID_Req_Sw = ?
                ORDER BY
                d.ID_Req_Sw";

        $stmt = mysqli_stmt_init($conn);

        if(!mysqli_stmt_prepare($stmt, $sql))
                exit('SQL error');

        mysqli_stmt_bind_param($stmt, 's', $id);
        mysqli_stmt_execute($stmt);

        $result = mysqli_stmt_get_result($stmt);

        $result_check = mysqli_num_rows($result);

        if($result_check > 0)
                $ReqSwRel = mysqli_fetch_all($result, MYSQLI_ASSOC);

        mysqli_close($conn);
        return $ReqSwRel;
}
```

APÊNDICE 4 – CÓDIGO EM HTML E PHP PARA EXIBIR O RETORNO DO MAPIR-CSR E A ESTRATÉGIA DE ACEITAÇÃO DA MUDANÇA

```
<?php

require_once '../../../config.php';
require_once '../../actions/software.php';
require_once '../partials/header.php';

if (isset($_POST["ID_Req_Sw_Original"]) && isset($_POST["ID_Req_Sw"]) &&
isset($_POST["Titulo"]) && isset($_POST["Story_Points"]))
   updateReqSwAction($conn, $_POST["ID_Req_Sw_Original"], $_POST["ID_Req_Sw"],
$_POST["Titulo"], $_POST["Story_Points"]);

$req = findReqSwAction($conn, $_GET['id']);
$reqRel = findReqSwRelAction($conn, $_GET['id']);
$reqImpacto = findImpactoReqSwAction($conn, $_GET['id']);
?>
<div class="container">
          <div class="row">
      <a href="../../pages/software/read.php"><h1>Requisitos de Software</h1></a>
      <a class="btn btn-success text-white" href="../../pages/software/read.php">Voltar</a>
   </div>
   <div class="row flex-center">
      <div class="form-div">
         <form class="form" action="../../pages/software/edit.php" method="POST">
                                    <input type="hidden" name="ID_Req_Sw_Original"
value="<?=htmlspecialchars($req['ID_Req_Sw'])?>">
            <label>ID</label>
            <input type="text" name="ID_Req_Sw" value="<?=htmlspecialchars($req['ID_Req_Sw'])?>"
size="6" maxlength="6" required/>
            <label>Titulo</label>
            <textarea name="Titulo" rows="3" cols="50" maxlength="255"
required><?=htmlspecialchars($req['Titulo'])?></textarea>
            <label>Story Points</label>
            <input type="number" name="Story Points"
value="<?=htmlspecialchars($req['story_points'])?>" size="2" min="1" max="21" required/><br>
            <button class="btn btn-success text-white" type="submit">Gravar</button>
         </form>
      </div>
                  <table class="table-users" <?php if(count($reqRel)==0) echo "hidden";?>>
                        <tr>
                              <td class="cell" colspan="2">
                                    <small><b>Impacto Potencial de
<?=htmlspecialchars($req['ID_Req_Sw'])?>:
<?=htmlspecialchars($reqImpacto['impacto_potencial'])?></b></small>
                              </td>
                        </tr>
                        <tr>
                              <td class="cell" colspan="2">
                                    <small><b>Estratégia para aceitação de mudanças: <?php
if($reqImpacto['impacto_potencial'] > 8) echo "Recusar"; else if($reqImpacto['impacto_potencial'] < 3)
echo "Aceitar"; else echo "Analisar Impacto"; ?></b></small>
                              </td>
```

```
                                </tr>
                                <tr>
                                        <td class="cell" colspan="2">
                                                <small><b>Requisitos de Software Potencialmente
Impactados por <?=htmlspecialchars($req['ID_Req_Sw'])?></b></small>
                                        </td>
                                </tr>
                                <tr>
                                        <th align="left"><small>ID</small></th>
                                        <th align="left"><small>Titulo</small></th>
                                </tr>
                                <?php foreach($reqRel as $rowRel): ?>
                                <tr>
                                        <td
class="cell"><small><?=htmlspecialchars($rowRel['ID_Req_Sw'])?></small></td>
                                        <td
class="cell"><small><?=htmlspecialchars($rowRel['Titulo'])?></small></td>
                                </tr>
                                <?php endforeach; ?>
                        </table>


   </div>
</div>
<?php require_once '../partials/footer.php'; ?>
```

APÊNDICE 5 – PERSONAS ADOTADAS PARA REPRESENTAR O CLIENTE E OS DIFERENTES PAPÉIS DA EQUIPE DE DESENVOLVIMENTO

1. Cliente – Jairo, o Artesão

Jairo possui uma pequena loja de artesanato e está ansioso para expandir seu negócio online. É uma pessoa ocupada, com conhecimento básico de tecnologia, mas bastante alinhado com as tendências digitais. Jairo busca um sistema de *e-commerce* fácil de usar, que permita um controle efetivo do estoque de sua loja e que evite que clientes inadimplentes sigam fazendo compras.

2. Desenvolvedora *Full-Stack* Sênior – Carolina, a Arquiteta de Software

Carolina tem sólida experiência em arquitetura de software, atuando em projetos de *B2B* e *B2C*. É responsável por liderar a equipe no desenvolvimento do *e-commerce* da loja de Jairo, projetando a arquitetura do sistema e de suas eventuais integrações. Os demais membros da equipe a veem como uma referência técnica, que apoia a cada um na definição das melhores estratégias para o desenvolvimento.

3. Desenvolvedora *Front-End* – Márcia, a Especialista em *Web Design*

Márcia é uma desenvolvedora Web especialista em interfaces de usuário modernas e práticas. Se concentra na usabilidade e no design centrado no cliente (*human-centered design*). Colabora diretamente com a *Product Owner* para assegurar que as necessidades de Jairo sejam atendidas pela interface do sistema.

4. Desenvolvedora *Back-End* – Lúcia, a Responsável pelo Banco de Dados

Lúcia é uma desenvolvedora especializada em bancos de dados relacionais e integrações de sistemas. Está focada na performance, disponibilidade e segurança do *e-commerce*. Trabalha em colaboração com Carolina para garantir que a modelagem do sistema seja adequada.

5. Scrum Master – Lívia, a Figura de Apoio

Lívia é uma gerente de projetos experiente, que atua como *Scrum Master* da equipe. Habituada ao mundo corporativo, facilita reuniões e ajuda o time removendo impedimentos do projeto, além de manter os desenvolvedores alinhados às melhores práticas de desenvolvimento ágil. Lívia promove a colaboração e potencializa os benefícios da comunicação osmótica dentro da equipe e com a *Product Owner*.

6. *Product Owner* – Meg, a Analista de Sistemas

Meg é a *Product Owner* e atua como um avatar do cliente. Define as prioridades dos *backlogs do produto* e da *sprint*, com base nas expectativas de Jairo a respeito da ordem das entregas e seu valor agregado. É a ligação entre o time de desenvolvimento e o cliente, garantindo que os requisitos do sistema atendam às necessidades mapeadas junto a Jairo.