FORMATO PARA ESPECIFICAÇÃO DE REQUISITOS CONJECTURAIS NO CONTEXTO DA EXPERIMENTAÇÃO CONTÍNUA EM ENGENHARIA DE SOFTWARE

Vladimir Marques Erthal

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

Orientador: Guilherme Horta Travassos

Rio de Janeiro

Outubro de 2023

FORMATO PARA ESPECIFICAÇÃO DE REQUISITOS CONJECTURAIS NO CONTEXTO DA EXPERIMENTAÇÃO CONTÍNUA EM ENGENHARIA DE SOFTWARE

Vladimir Marques Erthal

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

Orientador: Guilherme Horta Travassos

Banca de defesa: Prof. Guilherme Horta Travassos
Profª. Luciana Maria Azevedo Nascimento
Profª. Claudia Maria Lima Werner
Prof. Johnny Cardoso Marques

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 2023

Erthal, Vladimir Marques

Especificação de Requisitos Conjecturais no Contexto da Experimentação Contínua em Engenharia de Software / Vladimir Marques Erthal. – Rio de Janeiro: UFRJ/COPPE, 2023.

XII, 100 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2023.

Referências Bibliográficas: p. 90 – 96.

1. Experimentação Contínua. 2. Engenharia de Software. 3. Engenharia de Requisitos. I. Travassos, Guilherme Horta. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Em primeiro lugar, agradeço a Deus, criador e sustentador de todas as coisas, por me incentivar a fazer ciência.

Agradeço à minha esposa, Marcela, pelo apoio, suporte, compreensão, amor, e por me estimular e me animar a dar início ao mestrado mesmo 13 anos após a obtenção do meu diploma de graduação. Aos meus pais, Solange e Luiz Augusto, por terem lutado para me dar uma educação de qualidade e por sempre me incentivarem a ler, estudar, pesquisar e me aprofundar no conhecimento. Aos meus filhos, Heitor e Gabriel, pois mesmo em sua pouca idade me enchem de esperança e amor.

Ao meu orientador, professor Guilherme Horta Travassos, pelos ensinamentos, compreensão, pela parceria nos artigos e principalmente pelo apoio no direcionamento do foco da minha pesquisa. À professora Luciana Maria Azevedo Nascimento pelas incontáveis revisões, reuniões e conversas que viabilizaram a finalização deste trabalho. Seu conhecimento, apoio e paciência infinita foram essenciais para que eu pudesse atingir esse resultado.

À professora Claudia Maria Lima Werner e ao professor Johnny Cardoso Marques, por aceitarem participar da minha banca e pelas contribuições. Aos professores Toacy Cavalcante de Oliveira, Ana Regina Cavalcanti da Rocha, Jano Moreira de Souza, Henrique Luiz Cukierman e Claudio Miceli de Farias pelas disciplinas ministradas durante o meu mestrado, e que muito contribuíram para o meu crescimento acadêmico.

Ao colega Bruno Pedraça de Souza e ao professor Paulo Sérgio M. dos Santos pela parceria nos artigos, que proporcionaram a base para esta dissertação. A todos os participantes do grupo ESE (Engenharia de Software Experimental) pelas discussões e pela troca de informações.

Ao amigo Diego Cerqueira pela parceria nas disciplinas e aos secretários do PESC Gutierrez da Costa e Ricardo Cezar Vieira pela presteza e apoio nas questões burocráticas.

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

FORMATO PARA ESPECIFICAÇÃO DE REQUISITOS CONJECTURAIS NO CONTEXTO DA EXPERIMENTAÇÃO CONTÍNUA EM ENGENHARIA DE SOFTWARE

Vladimir Marques Erthal

Outubro/2023

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

A Experimentação Contínua tem sido cada vez mais utilizada na construção de sistemas de software devido ao entendimento de que existem incertezas em relação às características do software que não podem ser resolvidas na fase de identificação de requisitos nem na fase de implementação, mas somente através da experimentação com software em execução. Porém, existe uma lacuna na literatura técnica sobre técnicas para especificação e atualização das incertezas sobre características do software. Neste sentido, esta dissertação propõe a categoria de Requisitos Conjecturais para encapsular essas incertezas no contexto da experimentação contínua, além de dois instrumentos para apoiar a sua especificação: o Formato de Escrita dos Requisitos Conjecturais e o Quadro de Experimentação para Suposições de Solução. Os benefícios esperados desta proposta são o apoio ao registro, à atualização e à priorização dos requisitos conjecturais. A proposta foi avaliada por meio de uma prova de conceito e de um estudo de viabilidade sobre o registro dos requisitos conjecturais. Os resultados desses estudos indicam a viabilidade da proposta e a suficiência do Formato de Escrita dos Requisitos Conjecturais para registrar características do software com suas incertezas.

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

CONJECTURAL REQUIREMENTS SPECIFICATION ON CONTINUOUS EXPERIMENTATION IN SOFTWARE ENGINEERING CONTEXT

Vladimir Marques Erthal

October/2023

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

Continuous Experimentation has been increasingly used in the construction of software systems due to the understanding that there are uncertainties regarding the characteristics that cannot be solved in the requirements identification or implementation phases but only through experimentation with software running. However, there is a gap in the technical literature on techniques for specifying and updating uncertainties about software characteristics. In this sense, this dissertation proposes the category of Conjectural Requirements to encapsulate these uncertainties in the context of continuous experimentation, in addition to two instruments to support their specification: the Conjectural Requirements Writing Format and the Experimentation Framework for Solution Assumptions. The expected benefits of this proposal are support for recording, updating, and prioritizing conjectural requirements. The proposal was evaluated through a concept proof and a feasibility study on recording conjectural requirements. The results of these studies indicate the feasibility of the proposal and the sufficiency of the Conjectural Requirements Writing Format to record software characteristics with their uncertainties.

Sumário

Lista de Figuras

Lista de Quadros

1 Introdução

Neste capítulo são apresentadas a motivação e contexto para a realização deste trabalho, além dos objetivos e da metodologia utilizada. E, por fim, a organização dessa dissertação.

1.1 Motivação e Contexto

A Experimentação Contínua (EC) é uma prática para projetos de software cujo objetivo é oferecer suporte à engenharia de sistemas de software através da definição sistemática de hipóteses, entrega contínua de software funcional para os usuários finais e monitoramento de métricas para determinar a aceitação ou rejeição das hipóteses baseada em evidências de utilização real do software (Fagerholm *et al.*, 2017). Ela surgiu no contexto das práticas de desenvolvimento ágil com forte ênfase na metodologia *Lean Startup* de "construir-medir-aprender" (Fagerholm *et al.*, 2014), tendo sido frequentemente aplicada em conjunto com outras práticas contínuas, como a Integração Contínua e a Entrega/Implantação Contínua (Lindgren e Münch, 2016).

Dan McKinley cunhou o conceito de "Experimentação Contínua" (*Continuous Experimentation*) em 2012 em uma apresentação sobre como a Etsy (uma empresa americana de comércio eletrônico) utilizava a experimentação para avaliar o retorno do investimento de potenciais funcionalidades (Auer e Felderer, 2018). Porém, a prática da experimentação em engenharia de software precede este termo em mais de uma década (Cole, 2001). Além disso, de acordo com (Auer *et al.*, 2021), o artigo seminal sobre esse conceito foi publicado em 2017 (Kohavi *et al.*, 2007), o que deu início à discussão acadêmica sobre esse tópico. Os autores deste artigo utilizaram o termo "experimentos controlados" para referirem-se a essa prática.

A partir disso, diversos termos foram propostos e utilizados para a mesma técnica, como Desenvolvimento Orientado a Dados (*Data-Driven Development*) (Bosch e Olsson, 2017), Sistemas de Experimentação e Inovação (*Innovation Experiment System*) (Bosch, 2012), Experimentos Controlados Online (*Online Controlled Experiments*) (Kohavi *et al.*, 2013), entre outros. Os estudos relatam vários processos e estratégias associados aos diversos termos (Erthal *et al.*, 2023). Apesar disso, os estudos secundários neste tema

normalmente utilizam o termo “Experimentação Contínua” para englobar todos os outros. Eu seguirei os colegas e farei o mesmo neste trabalho.

A EC exerce um papel fundamental ao oferecer suporte para a colaboração entre potenciais usuários e a equipe de desenvolvimento com o objetivo de descobrir as necessidades reais de utilização do software (Olsson and Bosch, 2013) e de validar as soluções propostas com o objetivo de gerar valor e inovação no produto. Além disso, a utilização de experimentos controlados permite que a EC guie as organizações de desenvolvimento de software para avaliar e priorizar seus esforços de desenvolvimento, como a implementação de requisitos/funcionalidades específicas baseada em dados de utilização dos usuários (Olsson e Bosch, 2014).

Isso representa uma mudança no processo de desenvolvimento tradicional, como o modelo Cascata, no qual um conjunto completo de requisitos precede o desenvolvimento do software, e mesmo nos modelos iterativos e incrementais, que também dependem de um conjunto bem definido de requisitos para cada iteração. Essa abordagem também contrasta com os métodos ágeis de desenvolvimento, como Scrum e *eXtreme Programming* (XP), nos quais os requisitos do software, mesmo que superficialmente definidos, são os objetivos a serem atingidos. Além disso, os métodos ágeis fornecem estratégias limitadas para priorização do backlog baseada em dados.

Ao contrário, na EC, o aprendizado sobre os clientes, os usuários e o mercado, é mais importante do que o código em si (Melegati *et al.*, 2019a). Portanto, as estratégias de experimentação possuem um papel central na priorização, elicitação e validação dos requisitos do software (Olsson e Bosch, 2019), guiando o desenvolvimento do produto e alterando os requisitos dependendo dos dados coletados dos usuários.

Usualmente, a EC utiliza um *design* experimental do tipo A/B para executar experimentos, onde usuários interagem com duas versões do mesmo software para validar a aceitação de uma hipótese específica (Olsson *et al.*, 2017). A EC tem sido largamente implementada na indústria, principalmente no contexto de aplicações web e móveis. Grandes empresas como Facebook, Google, Microsoft, LinkedIn e Netflix têm relatado suas experiências utilizando EC para evoluir seus produtos de software (Auer *et al.*, 2021), principalmente através da utilização de testes A/B. Por exemplo, em 2013 a

Microsoft afirmava que apenas um terço das ideias testadas com experimentos controlados indicava melhoria nas métricas (Kohavi *et al.*, 2013).

Assim, esta prática evita o desperdício de tempo e recursos com ideias ruins, garantindo um alto retorno sobre o investimento (ROI) (Kohavi *et al.*, 2007). Porém, outras estratégias de experimentação para EC também são relatadas na literatura técnica, como *Beta Tests*, *Canary Release*, MVP (*minimum viable product – produto mínimo viável*), e *quasi-Controlled Experiments*. Como cada técnica possui os seus próprios objetivos (Mattos *et al.*, 2021), é necessário escolher a estratégia apropriada para cada contexto, de acordo com o propósito do experimento.

Como se percebe, não é fácil observar um entendimento comum sobre a EC no desenvolvimento de software. Além disso, os diversos entendimentos e perspectivas dificultam a realização de uma síntese nas contribuições dos artigos. Algumas vezes, são utilizadas diferentes expressões para o mesmo conceito ou palavras similares para outros conceitos. A falta de uma terminologia comum motivou um estudo secundário para melhor entendimento e consolidação das definições, processos e estratégias de experimentação no contexto da EC (Erthal *et al.*, 2022) (Erthal *et al.*, 2023).

Os resultados deste estudo secundário também indicaram que a literatura técnica sobre EC geralmente trata da fase de execução dos experimentos e da fase de análise dos resultados para tomada de decisão. Raros estudos se preocupam em detalhar as atividades da fase de planejamento, mais especificamente a criação e o gerenciamento de requisitos e hipóteses, além da sua integração com o desenvolvimento do produto. A diligente condução dessa fase e o gerenciamento dos requisitos e hipóteses é importante para garantir a qualidade da execução e da análise dos experimentos, de modo que eles produzam as informações necessárias para a redução das incertezas sobre características do software e a tomada de decisão sobre o seu desenvolvimento.

A certeza de que existem incertezas em relação às características do software que não serão resolvidas na fase de elicitação nem na fase de implementação é um aspecto essencial da EC, que contrasta com outros processos de desenvolvimento de software. Essas incertezas poderão ser resolvidas apenas através da experimentação com software funcional, preferencialmente com a sua utilização pelos usuários finais. A literatura técnica sobre EC fornece diversas técnicas de experimentação e processos para a

integração entre experimentação e desenvolvimento. Porém, os resultados do estudo secundário relatado anteriormente indicaram não existir ainda um entendimento sobre a especificação e o gerenciamento das hipóteses e das incertezas em relação às características do software.

1.2 Problema de Pesquisa

Muitos trabalhos preocupam-se com a execução dos experimentos, mas poucos tratam da especificação das incertezas e da atualização dessas incertezas utilizando os resultados dos experimentos, com o objetivo de reduzir sistematicamente as incertezas associadas aos requisitos e validar suposições de solução por meio delas (Erthal *et al.*, 2023). Os poucos trabalhos que consideram essa perspectiva expõem os problemas de falta de uso sistemático de experimentos e feedback dos clientes, falta de técnicas para tratar e especificar hipóteses, falta de um artefato para representar hipóteses (Melegati e Wang, 2019), e falta de um conjunto claro de atividades referentes a hipóteses na literatura técnica (Melegati *et al.*, 2020).

Segundo os autores, esses problemas afetam a integração da experimentação no processo de desenvolvimento (Melegati e Wang, 2019), o que pode potencialmente causar frustração das equipes, medo que os usuários não apreciem ou não utilizem as características implementadas, baixa adoção da experimentação (Melegati e Wang, 2019), baixo uso de técnicas para a descoberta das hipóteses que os experimentos devem ser construídos para testar, e baixo uso de técnicas de priorização para facilitar a definição das hipóteses ou a ordem dos testes (Melegati *et al.*, 2020).

Dado que, na EC, existe o pressuposto de que existem incertezas sobre as características do software que não poderão ser resolvidas sem um processo de experimentação e que precisarão ser consideradas, priorizadas e atualizadas, é importante que elas sejam registradas, de modo a facilitar essas atividades. A falta de instrumentos dedicados ao registro das incertezas sobre características do software pode potencialmente fazer com que as incertezas não sejam registradas, permanecendo apenas como conhecimento tácito. Isso pode levar os participantes do projeto a projetarem diferentes compreensões sobre as incertezas, bem como sobre a atualização dessas incertezas a partir do conhecimento gerado pelos resultados dos experimentos. A falta de

registro das incertezas também dificulta sua priorização para experimentação e a transferência de seu conhecimento para novos participantes. Considerando este cenário, esta dissertação tem seu problema de pesquisa descrito na Quadro 1.

Quadro 1 – Problema de pesquisa

O problema de	Falta de um formato de escrita para especificar, gerenciar e atualizar o conhecimento sobre as incertezas nos requisitos (requisitos conjecturais), obtidos por meio de estratégias de experimentação ao longo do processo de desenvolvimento do software.
Afeta	A prática do registro das incertezas, a recuperação do conhecimento sobre as incertezas nos requisitos, a complexidade de entendimento dos experimentos e dos seus resultados, a transferência de conhecimento para novos participantes e sua utilização na priorização das características para desenvolvimento do software.
Cujo risco é	Desconsiderar e não utilizar o conhecimento gerado sobre as incertezas, aumentar o risco de entregar no software características inadequadas às necessidades dos usuários e o desperdício de esforço devido a trabalho inadequado.

1.3 Objetivo e Proposta

A literatura técnica mostra que a aplicação da EC gera um aumento do conhecimento sobre as características do software, a redução do risco de entrega de características inadequadas e a integração da experimentação no processo de desenvolvimento de software. A proposta denominada Especificação de Requisitos Conjecturais (ERC) se encontra nesse contexto, e tem como **objetivo apoiar o registro dos requisitos que possuem incertezas (requisitos conjecturais) por meio da utilização de um formato de escrita para estes requisitos conjecturais e também por meio de um formato de escrita dos aprendizados gerados sobre as incertezas como resultado da experimentação**.

Neste sentido, a ERC propõe a categoria de Requisitos Conjecturais para encapsular as incertezas sobre características do software que precisarão ser tratadas através de experimentação, em conjunto com dois instrumentos para apoiar o seu registro e atualização: o Formato de Escrita dos Requisitos Conjecturais e o Quadro de Experimentação para Suposições de Solução. Esses instrumentos serão apresentados em detalhes no Capítulo 3. A expectativa é que eles sejam utilizados para apoiar o registro das incertezas e dos aprendizados resultantes dos ciclos experimentais, para facilitar a priorização das incertezas para novos experimentos e para possibilitar a atualização das incertezas a partir dos aprendizados obtidos nos experimentos realizados.

Resumidamente, a ERC propõe que:

- Os requisitos ainda incertos sejam registrados juntamente com os demais requisitos do sistema, utilizando um **formato específico de escrita**;
- Os experimentos executados tenham os seus resultados sumarizados em um **documento de síntese da experimentação**; e
- Um resumo do aprendizado obtido a partir dos experimentos sobre uma incerteza específica seja **registrado juntamente com o requisito conjectural** associado a ela e utilizado para **atualizar** essa incerteza.

Utilizando um exemplo retirado do estudo de viabilidade (apresentado no quinto capítulo desta dissertação), em um software existe o desejo de que seja possível obter a localização de equipamentos hospitalares em tempo real. Porém, não é de conhecimento da equipe do projeto qual modelo de sensor possibilitará a correta localização dos equipamentos nem a forma como esse sensor será localizado. Essas incertezas só poderão ser resolvidas através de soluções supostas, construídas e experimentadas. Nesse cenário, não é possível escrever requisitos sobre o modelo de sensor que deverá ser utilizado e a forma de localização. Assim, essas incertezas possivelmente não seriam registradas, o que poderia dificultar o seu entendimento, priorização e gerenciamento. Os instrumentos propostos nesta dissertação tem o objetivo de apoiar esse registro por meio de um modelo para descrever um requisito conjectural e encapsular as incertezas associadas com a localização de equipamento hospitalaress e de um modelo para o registro dos aprendizados obtidos como resultado dos experimentos sobre diferentes tipos de sensores. Estes aprendizados, por sua vez, também podem ser utilizados para atualizar as incertezas

do requisito conjectural registrado. Por meio desses dois modelos, as incertezas a serem experimentadas podem ser explicitadas de modo a fornecer um entendimento comum aos participantes do projeto, facilitando a sua priorização, atualização e gerenciamento.

Os benefícios esperados desta proposta no contexto de apoio à especificação de requisitos, em projetos de desenvolvimento de produtos de software com experimentação contínua, tendo como público-alvo os responsáveis pela identificação dos requisitos e tomadores de decisões sobre as características do software, são:

- Apoiar a priorização de requisitos para desenvolvimento com o **uso de um documento de requisitos contendo os requisitos conjecturais,** através da organização de um único artefato com os requisitos funcionais, não funcionais, conjecturais e suas incertezas;
- Em contextos nos quais não há documentos de registro da experimentação, **apoiar a recuperação do conhecimento sobre as incertezas dos requisitos conjecturais** obtidos na execução dos experimentos, o qual poderia permanecer apenas como conhecimento tácito;
- Em contextos que possuem documentação sobre a experimentação, **facilitar o acesso às lições aprendidas (conhecimento) dos experimentos sobre possíveis soluções para os requisitos conjecturais**, durante a elicitação de requisitos e no planejamento de novos ciclos experimentais;
- **Promover um artefato de requisitos passível de atualização e revisão** a partir do conhecimento obtido pela experimentação, evitando a entrega de características inadequadas e desperdício de trabalho; e
- **Facilitar a transferência de conhecimento sobre os requisitos conjecturais** para os novos participantes e associados.

1.4 Metodologia

A forma de condução desta pesquisa foi influenciada pela metodologia proposta por Shull, Carver e Travassos (Shull *et al.*, 2001), embora não a aplique em sua totalidade, que consiste na realização de estudos experimentais para o desenvolvimento de uma nova

tecnologia baseada em evidências. A Figura 1 mostra as etapas utilizadas para a criação da Especificação de Requisitos Conjecturais.

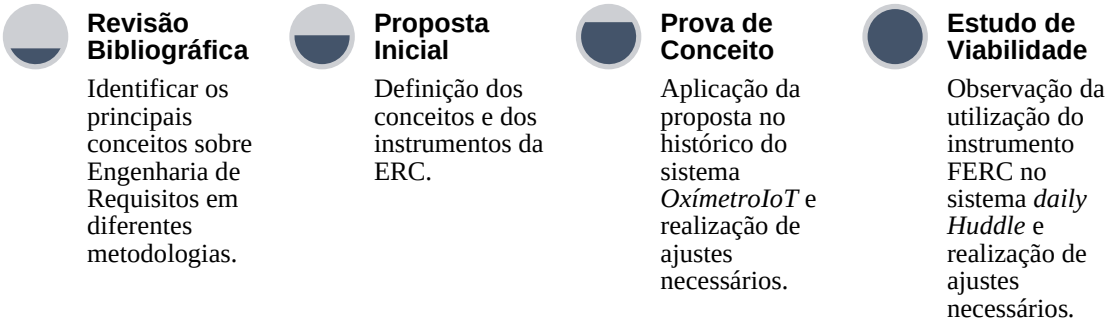

Figura 1 – Metodologia utilizada nesta pesquisa.

1. *Revisão Bibliográfica:* pesquisa da literatura técnica sobre Engenharia de Requisitos para identificar os principais conceitos e sua aplicação em diferentes metodologias de desenvolvimento de software.
2. *Proposta Inicial:* definição dos conceitos utilizados na proposta de Especificação de Requisitos Conjecturais e definição dos seus instrumentos: o Formato de Escrita dos Requisitos Conjecturais e o Quadro de Experimentação para Suposições de Solução.
3. *Prova de Conceito:* avaliação inicial da proposta da ERC através da sua aplicação no histórico de experimentações do sistema *OxímetroIoT*[1], e realização de ajustes na proposta para aprimorá-la de acordo com as necessidades encontradas.
4. *Estudo de Viabilidade:* verificação da suficiência e da facilidade de uso do instrumento FERC através da observação da sua utilização no sistema *daily Huddle*[2], e realização de ajustes na proposta para aprimorá-la de acordo com as necessidades encontradas.

[1] O *OxímetroIoT* é um sistema de software que visa a criação de uma solução computacional com equipamentos de baixo custo para monitoramento de sinais vitais de pacientes em enfermarias de hospitais.

[2] O *daily Huddle* é um sistema de software que visa otimizar a comunicação e a gestão de recursos que afetam a rotina hospitalar (como estruturas, equipamentos, materiais, pessoas e processos), funcionando

1.5 Organização do Texto

Esta dissertação está organizada em outros cinco capítulos e um apêndice, além deste primeiro que descreveu a introdução, motivação e o contexto no qual esta dissertação está inserida. A organização desse trabalho segue a estrutura abaixo:

Capítulo 2 – Revisão Bibliográfica e Conceituação*:* Apresenta os principais conceitos sobre a engenharia de requisitos em diferentes metodologias de desenvolvimento de software, e sobre experimentação contínua em engenharia de software, disponíveis na literatura técnica.

Capítulo 3 – Proposta para a Especificação de Requisitos Conjecturais*:* Descreve o conceito de Requisitos Conjecturais e propõe sua especificação através de dois instrumentos: o Formato de Escrita de Requisitos Conjecturais e o Quadro de Experimentação para Suposições de Solução.

Capítulo 4 – Prova de Conceito*:* Descreve a prova de conceito realizada sobre o uso dos instrumentos da ERC para o registro de requisitos conjecturais, executado ao longo dos ciclos experimentais do projeto OxímetroIoT, desenvolvido na Universidade Federal do Rio de Janeiro.

Capítulo 5 – Estudo de Viabilidade de Aplicação do Formato de Escrita dos Requisitos Conjecturais: Descreve o estudo de viabilidade realizado sobre a utilização do instrumento FERC para o registro de requisitos conjecturais, executado no contexto do projeto *daily Huddle*, desenvolvido na Universidade Federal do Rio de Janeiro.

Capítulo 6 – Considerações Finais*:* Descreve os principais resultados e contribuições deste trabalho, além de mostrar as perspectivas futuras da pesquisa em questão.

APÊNDICE A - Versão Final da Especificação de Requisitos Conjecturais: Apresenta a versão final da proposta da Especificação de Requisitos Conjecturais, evoluída de acordo com os aprendizados obtidos por meio da prova de conceito e do

como uma ferramenta para localizar, alocar e comunicar ocorrências, pendências ou restrições, atribuições de responsabilidade e prazos relacionados aos aspectos monitorados.

estudo de viabilidade, composta por seus instrumentos: o Formato de Escrita dos Requisitos Conjecturais e o Quadro de Experimentação para Suposições de Solução.

APÊNDICE B – Materiais Referentes ao Estudo de Viabilidade: Apresenta os materiais fornecidos aos participantes do estudo de viabilidade e os materiais produzidos como resultado do estudo.

2 Revisão Bibliográfica e Conceituação

Neste capítulo, são abordados os principais conceitos disponíveis na literatura técnica sobre a engenharia de requisitos em diferentes metodologias de desenvolvimento de software, e sobre a experimentação contínua em engenharia de software.

2.1 Introdução

Na Engenharia de Software, chamamos de "requisito" a expressão de um comportamento ou atributo desejado para o software que será desenvolvido (Pfleeger e Atlee, 2010). Um requisito é uma condição que restringe a execução do software ou uma capacidade que precisa ser implementada no software de modo a satisfazer um padrão ou as necessidades do cliente e/ou usuário (Pandey *et al.*, 2010). Tais necessidades podem ser declaradas em um contrato, uma especificação ou outro documento formalmente imposto.

No paradigma de desenvolvimento de software orientado a objetos, requisitos relacionam-se com objetos ou entidades, os estados que eles podem assumir e as funções que são executadas para mudar estados ou características dos objetos (Pfleeger e Atlee, 2010). Requisitos relacionados com comportamentos e funcionalidades de um sistema de software são chamados de Requisitos Funcionais, enquanto requisitos relacionados com as restrições sob as quais o sistema deve operar são chamados de Requisitos Não Funcionais (Valente, 2020).

Ao processo composto por atividades responsáveis pela identificação, documentação e gerenciamento dos objetivos, funções e restrições de um sistema de software, damos o nome de Engenharia de Requisitos (Nuseibeh e Easterbrook, 2000) (Arif *et al.*, 2009). A Engenharia de Requisitos é parte importante da Engenharia de Software, sendo responsável por explicitar e comunicar as funcionalidades desejadas de modo a suprir as necessidades dos usuários, respeitando também as restrições do contexto (Arif *et al.*, 2009) (Rehman *et al.*, 2013). Ela envolve todas as atividades relacionadas com a descoberta, análise do problema ou oportunidade, documentação dos resultados

dessa análise em algum formato de representação, validação da precisão do conhecimento obtido e manutenção dos requisitos de um sistema de software (Vegendla *et al.*, 2018).

Essas atividades devem ser realizadas de modo sistemático e ao longo de todo o ciclo de vida do software através da utilização de técnicas bem definidas (Valente, 2020). Porém, essas técnicas podem variar de acordo com a metodologia utilizada no processo de desenvolvimento do software. As próximas subseções apresentam algumas dessas metodologias com suas práticas, a saber: engenharia de requisitos tradicional, engenharia de requisitos no contexto dos métodos ágeis e engenharia de requisitos e experimentação. Além disso, também são exploradas as vantagens e desvantagens da experimentação contínua e o papel das incertezas em requisitos de software.

2.2 Engenharia de Requisitos Tradicional

A Engenharia de Requisitos tradicional está relacionada com processos de desenvolvimento de software nos quais toda a descoberta e especificação de requisitos é realizada antes da construção de um entregável do software, de modo a garantir que a equipe do projeto conheça todo o escopo do software para possibilitar o seu desenvolvimento de forma assertiva. O principal representante desse tipo de desenvolvimento é o **Modelo de Desenvolvimento em Cascata**, tendo sido organizado principalmente devido à construção de grandes e complexos projetos aeroespaciais e governamentais (Sommerville, 2020). Outros modelos de desenvolvimento, como o **Faseado** e o **Espiral** (Pfleeger e Atlee, 2010), também são considerados como tradicionais, pois preveem a especificação de requisitos separada e completa antes do desenvolvimento de cada entregável do software.

Na literatura técnica, as etapas da Engenharia de Requisitos tradicional variam de acordo com os autores. Algumas etapas citadas pela maioria dos autores estão apresentadas no Quadro 2.

Quadro 2 – Etapas da Engenharia de Requisitos Tradicional

Etapa	Descrição
(1) Elicitação	Etapa onde ocorre a coleta dos requisitos desejados e dos objetivos, necessidades e restrições do sistema de software, obtidos pela utilização de técnicas e métodos aplicados junto com as partes interessadas (Vegendla *et al.*, 2018). Algumas das técnicas mais comuns utilizadas nessa etapa são questionários, observação, entrevistas, *brainstorming* e prototipação (Aguilar Calderón *et al.*, 2016).
(2) Análise	Etapa cujo objetivo é realizar o entendimento e a modelagem do comportamento desejado para o sistema de software (Pfleeger e Atlee, 2010), incluindo a criação de modelos conceituais (Aguilar Calderón *et al.*, 2016), com o objetivo de garantir que os requisitos são claros, completos e não possuem ambiguidades (Vegendla *et al.*, 2018). Em caso de conflitos nas definições dos requisitos, é necessário executar uma atividade de negociação, a fim de resolver os conflitos e obter a concordância das partes interessadas (Vegendla *et al.*, 2018).
(3) Especificação (ou Documentação)	Etapa onde as características propostas para o sistema de software serão descritas de forma integral em um documento formal ou em vários documentos complementares (Pandey *et al.*, 2010). As técnicas mais comuns de documentação são modelos, cenários, casos de uso e linguagem natural (Aguilar Calderón *et al.*, 2016).
(4) Validação	Etapa onde os requisitos especificados são verificados juntamente com as partes interessadas, garantindo que o entendimento realizado sobre o problema coincide com o entendimento das partes (Pressman, 2016). Assegura que os requisitos são corretos (validação) e que estão descritos da forma correta (verificação) (Pandey *et al.*, 2010).
(5) Gerenciamento	Etapa que ocorre em paralelo a todas as etapas de construção do software, com o objetivo de manter a coerência dos requisitos em relação ao software desenvolvido (Pressman, 2016). Ela inclui

	atividades para controle e rastreamento das mudanças nos requisitos após a especificação, relacionamentos entre os requisitos e dependências de requisitos com fontes externas durante todas as fases do desenvolvimento do software (Pandey *et al.*, 2010), incluindo também técnicas de gerenciamento de configuração e controle de versão (Aguilar Calderón *et al.*, 2016). O controle de mudanças é especialmente importante, pois é natural que detalhes das funcionalidades do produto mudem conforme o desenvolvimento avança e as partes envolvidas aprendem mais sobre ele (Sommerville, 2020). Por isso, é preciso que a especificação acompanhe as mudanças solicitadas e realizadas.

Algumas das técnicas mais utilizadas para documentação de requisitos são: Casos de Uso, Perfis UML, Modelos Textuais e Diagramas de Atividades (Aguilar Calderón *et al.*, 2016). Destas, a documentação por casos de uso é a técnica mais popular. Ela consiste em uma descrição textual de cada cenário de interação entre os atores e o sistema de software, incluindo os diversos fluxos que podem ocorrer nessa interação. Diversos casos de uso podem ser representados conjuntamente de maneira gráfica através de um diagrama de casos de uso (Valente, 2020).

A literatura técnica deixa claro, porém, que a etapa de elicitação de requisitos é uma tarefa extremamente complexa e difícil de ser realizada de modo efetivo, pois não é simplesmente uma forma de fazer perguntas de modo a "sugar os requisitos da mente do cliente" (Pfleeger e Atlee, 2010). Nos estágios iniciais do projeto, os requisitos ainda estão mal-formados e mal-entendidos por todos, pois os clientes não são tão bons em descrever precisamente o que eles precisam quanto são em realizar o seu trabalho (Pfleeger e Atlee, 2010). Da mesma forma, desenvolvedores de software também não são tão bons em entender as preocupações concernentes a outros contextos de negócio e como eles podem ser impactados quanto são em compreender soluções computacionais (Pfleeger e Atlee, 2010).

Muitas vezes, os limites do sistema não são bem definidos ou os clientes/usuários especificam detalhes técnicos desnecessários que podem confundir, ao invés de esclarecer, os objetivos gerais do sistema (Pressman, 2016). Além disso, os

clientes/usuários não possuem certeza das necessidades que deverão ser atendidas pelo sistema, possuem pouco entendimento das capacidades e limitações do ambiente computacional, não possuem entendimento completo do domínio do problema, possuem problemas de comunicação com a equipe do projeto, omitem informações que para eles são “óbvias”, especificam requisitos conflitantes com suas próprias necessidades, ou especificam requisitos ambíguos ou impossíveis de serem testados (Pressman, 2016). Entender as diferentes visões e suposições e uni-las em um conjunto coerente de requisitos, obtendo a concordância de todos os envolvidos é o grande desafio da Engenharia de Requisitos e, caso isso não seja realizado, o projeto está fadado ao fracasso (Pfleeger e Atlee, 2010).

Além desses problemas, também há a questão de os requisitos serem definidos e fechados antes do início do desenvolvimento, o que faz com que mudanças certamente ocorram, tornando os documentos de requisitos obsoletos e exigindo grande atuação na atividade de gerenciamento de mudanças. Adicionalmente, um longo processo de desenvolvimento com base apenas em documentos, sem contato com o cliente, pode fazer com que ao final da codificação o cliente conclua que aquele sistema não atende mais às necessidades do negócio, pois a visão estratégica da empresa e suas prioridades mudaram (Valente, 2020).

Variações do modelo Cascata foram propostas ainda dentro do desenvolvimento tradicional, com o objetivo de reduzir o tempo gasto em cada etapa e entregar software de maneira mais rápida. O modelo de Desenvolvimento Faseado e o modelo Espiral, por exemplo, sugeriam a divisão do projeto em diversas partes entregáveis incrementais e complementares, montando ao final a funcionalidade completa requisitada (Pfleeger e Atlee, 2010). Porém, mesmo com a divisão em entregáveis, esses modelos ainda apresentam grande rigidez nas atividades da engenharia de requisitos realizadas em cada fase. Ou seja, os requisitos de cada fase ainda precisam ser completamente definidos antes do início do desenvolvimento da fase.

As etapas apresentadas da engenharia de requisitos tradicional surgiram nesse contexto, porém, elas também estão presentes nas demais metodologias da engenharia de requisitos. Nas próximas subseções, serão apresentadas outras metodologias e as diferenças de abordagens em relação à essas etapas.

2.3 Engenharia de Requisitos no Contexto dos Métodos Ágeis

Até os anos 1990, havia um pensamento comum na engenharia de software de que, para que qualquer sistema de software fosse bem construído, era necessário um minucioso planejamento prévio e controle rígido sobre o processo de desenvolvimento, também conhecido como “desenvolvimento orientado a planejamento” (Sommerville, 2020). Esse tipo de abordagem é justificável em sistemas críticos, onde o software será mantido e atualizado durante muitos anos utilizando como base os documentos de especificação de requisitos.

Porém, a aplicação do desenvolvimento orientado a planejamento em produtos de software de tamanho pequeno ou médio gera grande e desnecessária sobrecarga no processo, demandando muito tempo e recursos na escrita de documentos que serão pouco utilizados no desenvolvimento e dificilmente utilizados depois dele. Como consequência, torna-se praticamente impossível entregar software rapidamente e responder adequadamente às mudanças necessárias que surgem após o início do desenvolvimento (Sommerville, 2020).

O sucesso comercial de um novo produto ou serviço de software é definido pela sua aceitação por parte dos clientes (quem paga pelo produto) e usuários (quem utiliza o produto), por isso, o seu desenvolvimento é difícil e arriscado, e muitas vezes as empresas desenvolvem o software errado (Lindgren e Münch, 2016), por mais detalhada que seja a etapa de elicitação de requisitos. Assim, a insatisfação com o modelo de desenvolvimento orientado a planejamento levou à criação dos métodos ágeis ainda nos anos 1990, permitindo que a equipe de desenvolvimento concentrasse esforços mais na construção do software do que na sua especificação e documentação completa. Isso gerou entregas mais rápidas de software funcional para os clientes, que passaram a validar resultados mais cedo e sugerirem novas funcionalidades com base nessa validação, que seriam implementadas nas próximas entregas, reduzindo atividades burocráticas e evitando trabalho que geraria pouca utilidade prática (Sommerville, 2020).

O Manifesto Ágil (Beck *et al.*, 2021) foi proposto em 2001 com base nessa experiência, tendo o objetivo de melhorar os processos de desenvolvimento de software através da valorização maior da satisfação do usuário, mais pela entrega de software

contínua com valor agregado e menos por processos, planos e documentação abrangente. Os princípios do Manifesto Ágil advogavam por uma redução no rigor dos processos de desenvolvimento, como extensa documentação e detalhamento, em favor de uma maior flexibilidade nas atividades, o que poderia gerar resultados na forma de software produzido mais rapidamente e com maior eficiência (Pfleeger e Atlee, 2010). Essa flexibilidade não significava a remoção das etapas de elicitação, análise e documentação de requisitos, mas sim que essas etapas seriam realizadas de forma suficiente, e não exaustiva, dado que um representante do cliente ou usuário estaria acompanhando todo o processo de desenvolvimento e poderia esclarecer as eventuais dúvidas que surgissem.

Esses princípios levaram a uma mudança estrutural nos processos de engenharia de software. As abordagens lineares, baseadas em um planejamento prévio de todo o projeto, foram substituídas pelas abordagens de ciclos iterativos, baseadas no planejamento constante e adaptável, em parceria contínua com um representante do cliente ou usuário, e entrega frequente (Brhel *et al.*, 2015). Isso deu aos projetos de software a capacidade de responder às mudanças imprevisíveis e de aproveitar a criatividade das pessoas envolvidas. Os valores expressos no Manifesto Ágil foram aplicados em novas metodologias de gestão e desenvolvimento de projetos de software, sendo *Scrum* e *Extreme Programming* as que tiveram maior aplicação na indústria (Madampe, 2017).

A Engenharia de Requisitos no contexto das metodologias ágeis possui grandes diferenças em relação ao contexto tradicional. Não existem requisitos funcionais e não funcionais descritos no formato de extensos e detalhados documentos, sendo a descrição das características necessárias do software reduzida a um nível suficiente para possibilitar o início do desenvolvimento (Sommerville, 2020).

Por exemplo, em comparação com as etapas da Engenharia de Requisitos Tradicional (Quadro 2), no método *Scrum* (Schwaber e Sutherland, 2011) existe uma etapa de elicitação e análise da demanda realizada pelo representante do usuário, que realiza a especificação preferencialmente no formato de histórias de usuários ou cenários. A etapa de validação é realizada por toda a equipe responsável pelo desenvolvimento, juntamente com o representante do cliente ou usuário, com o objetivo de esclarecer dúvidas e fornecer maiores detalhes. A etapa de gerenciamento é de grande importância,

pois nesse cenário é natural que ocorram mudanças nos requisitos e o aparecimento de novos requisitos durante o desenvolvimento. Para isso, a participação do representante do usuário é essencial durante todo o processo de desenvolvimento, bem como para realizar a validação prévia dos resultados e garantir que o software suprirá as necessidades dos usuários.

Na etapa de elicitação de requisitos nos métodos ágeis, uma técnica de descrição de requisitos que se tornou bastante popular foi a das Histórias de Usuários. Essa técnica é especialmente relevante para a proposta deste trabalho, pois o seu formato de descrição foi utilizado como base para o Formato de Escrita dos Requisitos Conjecturais, conforme será explicado na seção 3.3.3.

As Histórias de Usuários são uma técnica de requisitos ágil composta por três partes: cartão, conversas e confirmação (Valente, 2020). No cartão é escrito pelo cliente ou usuário, em linguagem natural e de forma resumida, a característica desejada para o sistema. Muitas vezes, essa característica é escrita no seguinte formato: “eu, como <papel do usuário>, gostaria de <característica desejada>, para <motivo>” (Sommerville, 2020). As conversas ocorrem entre o cliente ou usuário e os desenvolvedores responsáveis pela construção do software de forma a explicar o que foi escrito no cartão e solucionar dúvidas. Por fim, o cliente ou usuário também escreve sua expectativa de aceitação, novamente em linguagem natural, para explicitar como ele saberá se o produto desenvolvido atende às suas necessidades.

A participação do cliente ou usuário, porém, não se restringe a esse momento inicial. Pelo contrário, ele participa de todo o processo de desenvolvimento, solucionando dúvidas e conferindo o resultado de cada etapa do desenvolvimento. É essa presença constante que permite que os requisitos sejam escritos sem muitos detalhes.

Entretanto, organizações em contextos específicos de desenvolvimento de software, como ambientes regulados cuja segurança de uso é crítica, tiveram dificuldades para adotar os métodos ágeis em seu processo de desenvolvimento. Exemplos desse tipo de ambiente são softwares desenvolvidos para os domínios automobilístico, médico, ferroviário, espacial, nuclear e da aviação. Nesse contexto, especificações de requisitos relacionados com segurança, confiabilidade, proteção de dados e rastreabilidade não podem ser entendidas como características burocráticas do processo.

Para superar essas dificuldades e usufruir dos benefícios trazidos pelos métodos ágeis, modelos híbridos entre processos tradicionais e ágeis foram propostos. Um exemplo é o contexto de ambientes regulados por padrões de software, como contextos médico, nuclear e automotivo (Marques e da Cunha, 2019). Assim, técnicas da engenharia de requisitos tradicional são resgatadas e mescladas com princípios e conceitos de metodologias ágeis para criar processos híbridos que satisfaçam as necessidades desse tipo de ambiente, como segurança e confiabilidade, por exemplo.

2.4 Engenharia de Requisitos e Experimentação

Apesar das empresas já estabelecidas terem se beneficiado das metodologias ágeis, esses conceitos tiveram especial aceitação em software startups, empresas recém-criadas com o foco em produtos de software inovadores (Melegati *et al.*, 2019b). Porém, esse tipo de organização muitas vezes não possui clientes nem usuários definidos no início do desenvolvimento do software. Assim, muitas vezes não é possível aplicar completamente o paradigma ágil de ter a participação de um representante do cliente ou usuário durante o desenvolvimento, pois ele ainda não existe. Em 2007, Steven Blank propôs o *Customer Development* (Blank, 2013), método de desenvolvimento para startups baseado na premissa de que o aprendizado sobre o cliente e seus problemas deve ser realizado o mais cedo possível no processo de desenvolvimento.

O *Customer Development* está fundamentado em quatro etapas: descoberta do cliente, validação do cliente, criação de clientela e construção da empresa. A principal contribuição de Blank é a sua ênfase no processo de descoberta do cliente, pois não é possível criar um mercado no qual não existe interesse dos clientes, não importa a quantidade de dinheiro investido. Assim, seu modelo é orientado para o aprendizado sobre os clientes e seus problemas o mais cedo possível no processo de desenvolvimento. Porém, Blank considera seu modelo como complementar aos métodos de desenvolvimento de software, e não como substituto. Ele é uma etapa preliminar, importante para que o desenvolvimento seja realizado sob bases mais assertivas para aceitação dos clientes.

Em 2011, Eric Ries baseou-se no raciocínio de Blank para propor o *Lean Startup* (Ries, 2011). O termo "*lean*" refere-se ao *lean manufacturing*, cujo objetivo é reduzir o

desperdício. Ele define *startup* como "uma instituição humana projetada para criar novos produtos e serviços sob condições de extrema incerteza". Essa definição engloba organizações de qualquer tamanho e em qualquer setor, não estando mais restrita a organizações pequenas e recém-criadas, e reconhece que o desenvolvimento de software precisa lidar com as incertezas presentes nos requisitos do sistema, especialmente no contexto de mercados pouco explorados. A forma que ele propõe para tratar essas incertezas é através da experimentação.

Segundo ele, o objetivo das *startups* não é criar coisas, gerar dinheiro nem mesmo servir aos clientes, mas sim gerar o aprendizado necessário para a construção de um negócio sustentável. De acordo com esse ponto de vista, esse aprendizado precisa ser validado cientificamente através da execução frequente de experimentos que permitam que os empreendedores testem cada elemento da sua visão de negócio. O objetivo é transformar ideias em produtos, medir a resposta dos usuários e aprender sobre os pontos nos quais deve-se investir e naqueles que devem ser repensados ou abandonados. Assim, ao invés de agir baseado em um plano complexo formado por muitas suposições, o *Lean Startup* sugere o ajuste constante no produto ou serviço baseado no ciclo "construir-medir-aprender".

Entretanto, uma organização não deve permanecer sendo uma *startup* durante um longo tempo. Este é um estado temporário da organização, enquanto ela ainda está em busca de um modelo de negócios repetível e escalável (Blank, 2013). Após a obtenção desse aprendizado, ela pode tornar-se uma organização com departamentos funcionais executando e mantendo esse modelo de negócios com a utilização das metodologias ágeis.

A principal contribuição do *Lean Startup* para a engenharia de requisitos encontra-se no estabelecimento de um novo paradigma, onde alguns requisitos não podem ser pré-definidos e não existem representantes dos clientes ou usuários para esclarecê-los. Existem crenças dos fundadores da empresa ou dos patrocinadores sobre o mercado e os clientes potenciais, porém, há um entendimento de que elas podem estar erradas, precisando ser validadas de forma prática.

De acordo com as etapas da Engenharia de Requisitos Tradicional (Quadro 2), esse conceito afeta principalmente as etapas de elicitação, análise e validação. Nas etapas de elicitação e análise, os requisitos passam a ser entendidos não como características

fixas, mas sim como suposições ou hipóteses, que podem estar certas ou erradas. A etapa de validação adquire a responsabilidade de provar as hipóteses juntamente com usuários finais através da utilização prática para demonstrar se eles compõem uma visão válida ou se são “apenas alucinações”. Se antes os requisitos eram fechados e aprovados no início do projeto, agora isso não é mais possível, pois não há certeza sobre as necessidades. Por isso, introduz-se a experimentação no processo de desenvolvimento, com o objetivo de validar as suposições sobre os requisitos na prática.

Além disso, ele também propõe um processo sistemático para elaboração, construção e avaliação de hipóteses, introduzindo o conceito do aprendizado constante, ou seja, a impossibilidade de completude dos requisitos e a avaliação frequente para identificar novas possibilidades e mudanças necessárias nas características do software. O principal conceito proposto para validação de hipóteses é o MVP (*minimum viable product* - produto mínimo viável), que consiste na construção de um software funcional que entregue algum valor para o usuário, mas que gere a menor quantidade possível de trabalho de desenvolvimento. Seu objetivo é indicar a aceitação ou não da hipótese principal do projeto por parte dos usuários. Em caso positivo, o software será evoluído com novas características; em caso negativo, a hipótese será abandonada antes que gere maior desperdício de esforço, e uma nova hipótese poderá ser formulada.

Paralelamente a esses trabalhos, outros autores também trabalhavam em ideias similares, expandindo o alcance do desenvolvimento de software baseado em experimentação, definindo metodologias e aplicando-o em diversos contextos de software. Essas metodologias foram inicialmente chamadas de *Online Controlled Experiment* (experimentos controlados online) (Kohavi *et al.*, 2007) (Crook *et al.*, 2009) e *Data-Driven Development* (desenvolvimento orientado a dados) (Kohavi *et al.*, 2013).

Os contextos iniciais desses trabalhos eram diferentes do contexto abordado no *Lean Startup*, sendo principalmente direcionados para a definição do formato de apresentação de novas características do software, sobretudo no ambiente web. Eles propunham a experimentação de novas características do software por meio do formato de Teste A/B, técnica também utilizada no *Lean Startup* (chamada de *split testing*), onde dois grupos de usuários utilizam duas versões do software paralelamente, uma com as

novas características e outra sem elas. Os dados gerados pela comparação entre a utilização dos dois grupos indicarão a aceitação ou não das novas características.

A falta de requisitos definidos e da presença de um representante do cliente ou usuário gerou a necessidade da criação de novos métodos para obtenção de informações sobre as características necessárias para o software. Portanto, novos métodos baseados em experimentação foram propostos e desenvolveram-se para abordar diversas técnicas experimentais, mas mantendo o Teste A/B como técnica principal. Atualmente, essa abordagem costuma ser mais conhecida pelo termo Experimentação Contínua (EC) (Auer *et al.*, 2021), que ressalta a inclusão da experimentação como parte do processo de identificação de requisitos e desenvolvimento, e não apenas para validação do software após o seu desenvolvimento. Alguns autores também costumam incluir a EC como parte da Engenharia de Software Contínua (*Continuous Software Engineering*), que engloba uma série de iniciativas para conectar todas as partes do processo de desenvolvimento, evitando descontinuidades prejudiciais para o software (Fitzgerald e Stol, 2017).

A experimentação contínua transforma as etapas da Engenharia de Requisitos Tradicional (Quadro 2), tornando o processo de elicitação, análise, documentação e validação de requisitos em ciclos experimentais iterativos e contínuos de testes de hipóteses baseados em dados de experimentos. Cada hipótese é validada diretamente com os usuários finais através de experimentos, com o objetivo de decidir se uma ideia deve ser expandida ou abandonada, inspirado pelo ciclo "construir-medir-aprender" do *Lean Startup* (Bosch e Olsson, 2016). Assim, a conclusão de cada ciclo experimental utiliza os seus resultados para dar início a um novo ciclo, de forma contínua.

2.4.1 Vantagens e desafios da Experimentação Contínua

A importância da experimentação é observada na literatura técnica, que mostra que a intuição das organizações de software sobre as preferências dos usuários pode ser inadequada em até 90% das vezes (Fabijan *et al.*, 2017) (Kohavi *et al.*, 2013). O motivo disso é que os produtos de software criados para contextos não familiares ou altamente dinâmicos possuem muitas incertezas sobre as necessidades e expectativas dos usuários. Nesses tipos de situação, é muito difícil ou até mesmo impossível prever as funcionalidades que irão gerar valor para os usuários, mesmo que eles sejam previamente

consultados. Além disso, também existe a dificuldade de entendimento entre “o que os usuários fazem” e “o que eles dizem que fazem” (Lindgren e Münch, 2016).

Assim, a EC permite que os engenheiros de software realizem experimentações com os usuários durante o processo de desenvolvimento, e decisões sejam tomadas mais cedo. Por exemplo, a falha rápida e a descoberta de que uma ideia não é tão útil quanto foi pensada permite que novas e melhores ideias sejam propostas e implementadas (Kohavi *et al.*, 2009), o que também permite a identificação de funcionalidades que prejudicariam o software, mesmo quando os *stakeholders* estão ávidos por elas (Kohavi *et al.*, 2013). Sem experimentação, decisões de implementação podem ser consideradas como baseadas apenas em opiniões, e são mais suscetíveis a politização ou a serem utilizadas para benefícios pessoais (Sauvola *et al.*, 2015). A EC oferece insumos para a tomada de decisão baseada em evidências, mitigando esse risco.

A EC agrega à engenharia de software a habilidade de definir entregas de valor baseada em necessidades reais, promovendo a validação de hipóteses guiada por dados empíricos de conceitos inovadores para o produto por meio de experimentos (Esteller-Cucala *et al.*, 2020), a descoberta de usos inesperados do produto (Feitelson *et al.*, 2013) e a tomada de decisão baseada em dados gerados pelos usuários finais (Schermann *et al.*, 2018). Além disso, a EC possibilita a entrega de software com valor para os clientes mesmo em meio a imprevisibilidade do mercado, requisitos complexos, pressão para entrega rápida e o acelerado avanço das ferramentas de informações (Gerostathopoulos *et al.*, 2018).

Além da melhor entrega de valor e da minimização do desperdício no sentido *Lean* (Fagerholm *et al.*, 2017), outros benefícios da EC são reportados na literatura técnica. Primeiramente, ela aumenta a chance de corresponder às expectativas dos usuários sobre privacidade, segurança, adaptação e autonomia de acordo com a evolução das condições (Fitzgerald e Stol, 2017). Em segundo lugar, ela otimiza a performance do produto (Olsson *et al.*, 2017). Em terceiro lugar, ela aumenta o engajamento dos usuários, prevenindo o abandono do serviço, reduzindo a migração para concorrentes (Gomez-Uribe e Hunt, 2015) e provendo um impacto significativo na receita anual (Olsson *et al.*, 2017).

Apesar dos vários benefícios da EC, a literatura técnica também apresenta alguns desafios. Por exemplo, em determinados contextos específicos de software é mais difícil implementar experimentação com usuários finais, como em sistemas críticos, sistemas legados, sistemas embarcados, sistemas que utilizam componentes de terceiros (Fitzgerald e Stol, 2017) e empresas com políticas de acesso que impedem o compartilhamento de dados dos usuários para experimentação (Fabijan *et al.*, 2017). A necessidade de mudanças na cultura organizacional das empresas também é um grande obstáculo (Lindgren e Münch, 2016), e a aplicação de experimentos em cenários inovadores requer criatividade e mudanças descontínuas no software (Fitzgerald e Stol, 2017).

Também são relatadas dificuldades acerca da avaliação dos dados dos experimentos, como o alinhamento entre os *KPIs* de negócio (*Key Performance Indicator* – indicadores-chave de performance) e as métricas específicas de desenvolvimento (Olsson *et al.*, 2017). A falta de métodos sistemáticos para coletar, analisar e incorporar dados dos clientes (Sauvola *et al.*, 2015), a pouca quantidade de usuários finais dos experimentos para garantir validade estatística, a falta de conhecimento sobre ciência de dados (Schermann *et al.*, 2018), entre outras, também são citadas em alguns trabalhos.

Além disso, o uso intensivo de dados dos usuários pode levar a decisões enviesadas devido à estratégia de amostragem usada para coletar os dados (Niculescu *et al.*, 2021). Por exemplo, uma amostragem aleatória para um Teste A/B dificilmente seria balanceada considerando identidades de gênero, etnia, classe social e subpopulações em geral. Portanto, essas considerações precisam ser ajustadas *a priori*, caso contrário, preocupações quanto à inclusividade podem permanecer despercebidas.

Alguns modelos foram propostos na literatura técnica com o objetivo de conduzir a prática da EC em organizações de desenvolvimento de software. Como exemplo, o *RIGHT Model for Continuous Experimentation* (Fagerholm *et al.*, 2017), o *Stairway to Heaven* (Karvonen *et al.*, 2015), o HYPEX (Olsson e Bosch, 2014), e o *Experimentation Growth Model* (Fabijan *et al.*, 2018). Entretanto, os papéis necessários para conduzir a EC de maneira apropriada são raramente especificados, exceto o papel de Cientista de Dados, responsável principalmente por projetar e analisar os experimentos, o qual é

frequentemente referenciado nos artigos. A falta desse papel pode levar as organizações a enfrentarem dificuldades na aplicação da EC (Schermann *et al.*, 2018).

A literatura técnica informa muitas vezes que a EC necessita de outras práticas contínuas para apoiar a sua implementação, como integração contínua, automação de testes e entrega contínua (Lindgren e Münch, 2016). Além disso, outros procedimentos podem auxiliar e facilitar a EC, como os conceitos de *DevOps* e Software como Serviço (SaaS – *Software as a Service*) (Lindgren e Münch, 2016). Embora não seja o objetivo desta dissertação, outros estudos mostram que contextos específicos de software possuem maiores dificuldades na adoção de todas essas práticas como requisitos para EC. Contextos como sistemas ciber-físicos de recursos restritos (Giaimo *et al.*, 2020), sistemas críticos (Giaimo *et al.*, 2016), ambientes de dados altamente regulados (Rissanen e Münch, 2015), entre outros que possuem dificuldade na adoção de todos os requisitos para EC, poderiam se beneficiar de uma abordagem orientada a dados do processo de desenvolvimento de software, mesmo sem a adoção de todas as práticas contínuas.

2.5 Incertezas em Requisitos de Software

No desenvolvimento de software com experimentação, existe o entendimento de que as incertezas associadas às características do software devem ser investigadas e validadas através de métodos de experimentação. Essas incertezas são relacionadas com diversos aspectos do software, como sua apresentação, usabilidade, suporte tecnológico, efetividade, aceitação dos usuários, entre outras. A literatura técnica relata diversos exemplos dessas incertezas. Um caso apresentado pela Microsoft (Fabijan *et al.*, 2017) mostra a incerteza sobre a forma de exibição dos preços em sua plataforma Xbox que resultará em maior engajamento e compras de produtos.

Incertezas podem ser identificadas a partir da observação de contextos e situações, da observação dos demais requisitos, ou podem ser ideadas através da suposição de cenários. A revisão da literatura apresentada em (Erthal *et al.*, 2023), porém, mostrou que não existem processos nem diretrizes claras para a identificação de incertezas sobre o software.

As incertezas estão associadas com a falta de conhecimento que os responsáveis pela especificação possuem sobre as características do software. Por vezes, esse

conhecimento pode ser adquirido através de técnicas de elicitação conhecidas junto aos clientes/usuários. Outras vezes, porém, não é possível que os responsáveis os obtenham antes que o software esteja desenvolvido e sendo utilizado pelos usuários finais. Sutcliffe e Sawyer (Sutcliffe e Sawyer, 2013) classificam o conhecimento sobre as características de software em quatro tipos:

- **Conhecimentos conhecidos (*known knowns*):** informações expressas e relevantes, de conhecimento tanto dos responsáveis pela especificação quanto do cliente/usuário. Esse tipo de conhecimento pode ser facilmente expresso na forma de requisitos funcionais e não funcionais ou de histórias de usuários.
- **Conhecimentos desconhecidos (*known unknowns*):** informações não conhecidas pelo cliente/usuário, porém relevantes e de conhecimento dos responsáveis pela especificação, que podem potencialmente validá-las através de técnicas de elicitação. Esse tipo de informação, se puder ser validada, também pode ser expressa em requisitos funcionais, não funcionais ou histórias de usuários. Entretanto, se não for possível validá-la, permanecerá como uma incerteza de característica do software.
- **Desconhecimento conhecido (*unknown knowns*):** informações não expressas e não articuladas pelo cliente/usuário, porém relevantes e de conhecimento dele, mas desconhecidas pelo analista. Esse tipo de informação pode ser suprimido por razões políticas, sociais ou emocionais, e pode eventualmente surgir através do uso de técnicas de elicitação. Porém, caso não surja, ela provavelmente permanecerá desconhecida até que o cliente/usuário comece a utilizar o software e relate a sua falta, tornando-se então um requisito tardio e gerando retrabalho para possibilitar a sua implementação, o que pode produzir custos adicionais e atraso na entrega do software.
- **Desconhecimento desconhecido (*unknown unknowns*):** informações potencialmente relevantes não expressas, não articuladas e não conhecidas pelo cliente/usuário nem pelo analista. Esse tipo de informação pode ser desconhecido por falta de entendimento do contexto ou das possibilidades de solução, mas também é característica de mercados voláteis, sujeitos a mudanças constantes. Além disso, esse tipo de conhecimento também é característico de contextos nos

quais não há um mercado nem clientes/usuários definidos a princípio, e onde o aprendizado sobre as características é parte fundamental da construção do software. Essas são incertezas que não serão identificadas a princípio, podendo ser supostas, mas que provavelmente serão reveladas através do processo de experimentação, e deverão ser tratadas como incertezas de características do software quando surgirem. Esse é o cenário no qual a Especificação de Requisitos Conjecturais pode contribuir mais.

A literatura técnica em EC costuma utilizar o termo "hipótese" para descrever características do software que possuem incertezas e precisam ser validadas através de experimentos (Auer *et al.*, 2021). Porém, poucos trabalhos definem um modelo de registro das hipóteses com suas incertezas e a maneira como elas devem ser atualizadas (Erthal *et al.*, 2023). Apenas um formato de descrição de hipóteses foi encontrados na literatura técnica.

Melegati, Guerra e Wang (Melegati *et al.*, 2020) propõem um formato de descrição de hipóteses que consiste em quatro partes (Quadro 3). Esse formato de descrição é adaptado das histórias de usuários dos métodos ágeis, sendo as duas partes iniciais exatamente iguais, trocando apenas a parte final das histórias de usuários, que trata da finalidade da solução, por duas partes que descrevem o processo de experimentação para a solução proposta.

Quadro 3 – Formato de descrição de hipóteses segundo (Melegati *et al.*, 2020)

If a <role>	Se um <papel>
Wants/prefers to <action/characteristic>	Deseja/prefere <ação/característica>
Then <evaluation process>	Então <processo de avaliação>
Should <evaluation result>	Deverá <resultado da avaliação>

Além dessa, também é possível encontrar outras formas de escrita de hipóteses na literatura cinza[3]. O'Reilly (O'Reilly, 2013), por exemplo, propõe um modelo com maior

[3] Literatura cinza, ou literatura cinzenta (do inglês *grey literature*), aqui é entendida como um tipo de informação científica produzida e disponibilizada por meios eletrônico ou impresso não arbitrados conforme as normas de publicação usuais da literatura técnica e sem controle bibliográfico.

foco no processo de experimentação, dividido em três partes (Quadro 4). Este formato enfatiza mais o processo experimental de validação da suposição de solução do que o primeiro, o qual enfatiza mais o papel e a necessidade do usuário que irá interagir com o software. Ele também deixa mais claro que a proposta de solução é apenas uma conjectura, que pode estar errada. O seu valor será descoberto através de uma ou mais estratégias de experimentação, que podem ser quantitativas e/ou qualitativas.

Quadro 4 – Formato de descrição de hipóteses segundo (O'Reilly, 2013)

We believe that <capability>	Nós acreditamos que <funcionalidade>
Will result in <outcome>	Resultará em <resultado>
We will know we have succeeded when <measurable signal>	Nós saberemos que obtivemos sucesso quando <indício mensurável>

Weigel (Weigel, 2022) sugere um modelo parecido com o de O'Reilly, mas acrescentando novas informações (Quadro 5). Esse modelo de seis partes destaca primeiramente o problema que o software deverá resolver, justificando-o com a fonte que originou a hipótese e explicitando o impacto causado pela falta de uma solução. As outras três partes possuem as mesmas informações que o modelo de O'Reilly.

Quadro 5 – Formato de descrição de hipóteses segundo (Weigel, 2022)

Based on <data, observation, research>	Baseado em <dados, observações, pesquisa>
We believe that <customer problem>	Nós acreditamos que <problema do usuário>
Because of this <consequence of customer problem>	Por causa disso <consequência do problema do usuário>
If we <solution>	Se nós <solução>
Then <anticipated impact of solution>	Então <impacto antecipado dessa solução>
We will know this is true when we see <metric and expected impact on that metric>	Nós saberemos que isso é verdade quando nós virmos <métrica e impacto esperado na métrica>

Outra forma de escrita de hipóteses encontrada na literatura cinza é a apresentada por Sullivan (Sullivan, 2021). Ele divide a descrição em três partes - Teoria, Validação e Resultado - e subdivide a primeira em quatro partes (Quadro 6). Esse modelo também utiliza as informações do modelo de O'Reilly, acrescido de três partes. A primeira é a

motivação que gerou a suposição. Essa motivação é mais genérica que a do modelo de Weigel, pois não se relaciona apenas com um problema de usuário, podendo aplicar-se a qualquer motivação, seja ela um problema de usuário, de negócio, uma necessidade de adequação a alguma normativa, uma verificação de aceitação ou qualquer outra motivação que tenha gerado a suposição.

A segunda parte acrescida é a determinação da população à qual essa suposição se aplica. Essa delimitação é importante, especialmente para a seleção dos atores que participarão dos experimentos. Por fim, a terceira parte acrescida é o impacto do resultado positivo em caso de sucesso da validação da suposição.

Quadro 6 – Formato de descrição de hipóteses segundo (Sullivan, 2021)

Theory (teoria)	Based on <data/research>	Baseado em <dados/pesquisa>
	We believe that <change>	Nós acreditamos que <alteração>
	For <population>	Para <população>
	Will cause <impact>	Causará <impacto>
Validation (validação)	We will know this when we see <metric or feedback>	Nós saberemos disso quando virmos <métrica ou parecer>
Outcome (resultado)	This will be good for customers, partners and our business <because>	Isso será bom para os clientes, parceiros e para o nosso negócio <motivo>

2.6 Conclusão do Capítulo

A Engenharia de Requisitos passou por muitas mudanças ao longo das últimas décadas. No contexto tradicional, ela era uma fase inicial que deveria determinar todos os requisitos do software de maneira completa e correta, determinando todos os comportamentos do software antes da sua codificação. Isso determinou a criação de etapas que passaram a ser um paradigma para a Engenharia de Requisitos (Quadro 2). Posteriormente, as transformações no processo de desenvolvimento, principalmente através das metodologias ágeis, resultaram na integração da descoberta e da especificação dos requisitos com a codificação do sistema, permitindo o desenvolvimento e a entrega do software em etapas.

Por fim, o desenvolvimento de softwares contemporâneos foi além, mostrando que muitas vezes não é possível resolver todas as incertezas sobre os requisitos antes que o sistema esteja pronto para ser avaliado pelos usuários. Nesse último contexto, encontra-se o *Lean Startup* e a Experimentação Contínua, que procuram entender os requisitos como hipóteses e experimentá-las iterativamente, especialmente na EC, de modo a obter um aprendizado constante sobre as necessidades e os desejos dos usuários e guiar o processo de desenvolvimento de acordo com eles.

A EC enfatiza as hipóteses acima dos requisitos, pois são os dados colhidos sobre as hipóteses que guiarão o desenvolvimento do software. Por isso, alguns autores afirmam que o conceito de requisitos é inadequado para o contexto da EC, e propõem a criação da Engenharia de Hipóteses como substituta da Engenharia de Requisitos nesse caso (Melegati *et al.*, 2019a) (Melegati *et al.*, 2019b) (Melegati e Wang, 2020). Segundo eles, a engenharia de requisitos é um importante componente do "desenvolvimento tradicional orientado a requisitos", porém, no "desenvolvimento orientado à experimentação", são as hipóteses que guiam a elicitação das necessidades dos usuários e evoluem de acordo com a codificação. Assim, como um processo de desenvolvimento orientado à experimentação, a EC se beneficiaria da engenharia de hipóteses para melhor identificar, priorizar, especificar, analisar e gerenciar suas hipóteses, reduzindo assim o desperdício de recursos e tempo.

Apesar disso, o gerenciamento de hipóteses é um desafio pouco explorado na EC, e as relações entre os requisitos, as hipóteses e as propriedades conjecturais do software ainda são incertas (Erthal *et al.*, 2022). Existe uma linha tênue entre hipóteses e conjecturas. Hipótese é uma proposta de solução cuja validade está envolta em dúvidas, porém com indícios de validade devido a evidências limitadas, enquanto conjectura é uma crença que ainda não possui evidências que a embase (Endres e Rombach, 2003). A experiência em projetos de sistemas de software contemporâneos indica que ideias e crenças (ou seja, conjecturas) frequentemente representam futuras oportunidades de tornarem-se requisitos que poderão evoluir o sistema. Entretanto, essas conjecturas precisam ser construídas e passar por uma avaliação experimental antes de tornarem-se hipóteses e requisitos.

De acordo com esse conceito, portanto, é possível entender tanto conjecturas quanto hipóteses e requisitos coexistindo no processo da EC. Nesse cenário, a necessidade de um método sistemático para o gerenciamento desses objetos torna-se ainda mais urgente. Apoiar essa necessidade, nas etapas de elicitação, documentação e gerenciamento das conjecturas e das hipóteses, é o objetivo da proposta da Especificação de Requisitos Conjecturais.

3 Proposta para a Especificação de Requisitos Conjecturais

Este capítulo apresenta o conceito de Requisitos Conjecturais e propõe sua especificação através de dois instrumentos: o Formato de Escrita de Requisitos Conjecturais e o Quadro de Experimentação para Suposições de Solução.

3.1 Introdução

Três motivos podem nos levar ao entendimento da importância do registro das incertezas sobre características do software. Primeiramente, a existência da incerteza precisa ser compreendida pelos responsáveis pela elicitação de requisitos para possibilitar a definição de uma hipótese adequada para avaliá-la. Além disso, é possível que a solução proposta e experimentada não resulte na resolução da incerteza, necessitando de uma nova proposta de solução. Nesse caso, o aprendizado gerado pela experimentação deve ser utilizado para atualizar a incerteza, de modo a produzir uma nova hipótese baseada em dados. Se a incerteza não estiver especificada separadamente da primeira hipótese, o risco de não identificar corretamente a incerteza e de criar uma nova hipótese não direcionada corretamente é alto.

No exemplo do software para localização de equipamentos hospitalares em tempo real, citado na subseção 1.3, existem incertezas em relação à distância entre cada equipamento e os sensores de localização, e também em relação aos locais com maior interferência, entre outras incertezas. Se elas não estiverem registradas, é possível que algumas informações se percam, prejudicando a comparação entre diferentes hipóteses. Nesse caso, a hipótese de uso de um determinado modelo de sensor pode eventualmente ser rejeitada e uma nova hipótese de outro modelo de sensor pode ser proposta sem que se saiba exatamente a performance de cada modelo para resolução de cada incerteza. Assim, o registro das incertezas auxilia na validação das propostas de solução experimentadas.

Por fim, o terceiro motivo ocorre quando há casos de projetos de software nos quais existem muitas incertezas quanto às suas características, e não é possível realizar

experimentos para validar todas elas. Assim, os experimentos precisam ser priorizados e executados ordenadamente.

Portanto, o registro das incertezas permite que elas não sejam perdidas e possam ser avaliadas em cada ciclo experimental. Em suma, a especificação das incertezas de características do software no contexto da EC auxilia no seu entendimento e gerenciamento, além de facilitar a sua priorização em cada experimento. Porém, os modelos de descrição de hipóteses apresentados no capítulo anterior (subseção 2.5) enfatizam mais a solução que será experimentada do que as incertezas existentes nos requisitos. A hipótese de solução necessariamente é originada por alguma incerteza sobre características do software, e essa incerteza não é diretamente explicitada em nenhum dos modelos apresentados.

Além da falta de explicitação das incertezas, o modelo apresentado em (Weigel, 2022) possui um objetivo muito restrito em um problema de usuário. Apesar do foco no problema ser muito importante em algumas hipóteses, existem incertezas que não estão relacionadas com problemas de usuários. Algumas incertezas de usabilidade, por exemplo, podem ser identificadas com o objetivo de definir uma forma de apresentação que melhore determinadas métricas associadas com necessidades estratégicas da organização, e não com problemas dos usuários. Um exemplo disso são os testes A/B mostrados em (Fabijan *et al.*, 2017) para determinar a forma de exibição de funções e informações para os usuários de modo a melhorar as métricas de engajamento, retenção, aquisição de produtos e economia de recursos.

Incertezas podem ser evoluídas ou descartadas de acordo com o aprendizado obtido durante o processo de desenvolvimento e experimentação. Essa evolução pode significar a resolução da incerteza ao identificar uma solução que gere dados de uso satisfatórios, a redução da incerteza ao obter dados que auxiliem no seu entendimento, mesmo que a solução proposta não seja totalmente satisfatória, ou a criação de novas incertezas a partir das informações obtidas pelo experimento. Entretanto, os modelos apresentados também não preveem essa atualização, nem possuem seção para registro dos aprendizados, o que pode tornar mais difícil o resgate de aprendizados passados quando é necessário inserir novos integrantes no projeto e para planejar novas propostas de solução.

Desta forma, o presente trabalho propõe a Especificação de Requisitos Conjecturais (ERC), cujo objetivo é fornecer dois modelos de descrição para apoiar a elicitação, análise e registro dos requisitos conjecturais e suas incertezas no contexto da experimentação contínua (EC) em engenharia de software, de modo a auxiliar a sua priorização e facilitar a transferência de conhecimento sobre o software.

Conforme visto, as descrições de hipóteses previstas nos modelos discutidos no capítulo anterior englobam dois conceitos distintos: as incertezas e as propostas de solução. A proposta da ERC não utiliza o termo "hipótese" para evitar a confusão de conceitos. Em seu lugar, é utilizado o termo "requisitos conjecturais" para agrupamento das incertezas relacionadas entre si e "suposições de solução" para as soluções propostas para experimentação. O registro desses conceitos separadamente pode facilitar o gerenciamento das incertezas de características do software. Assim, o processo de experimentação pode ser mais bem direcionado para validar suposições de solução que possam reduzir essas incertezas através da análise dos dados de utilização do software.

Este trabalho denomina como "requisitos conjecturais" as características do software que possuem incertezas que não podem ser resolvidas previamente, sem o uso da experimentação. Requisitos Conjecturais, por sua natureza incerta, não são estáticos. Pelo contrário, eles devem ser atualizados através dos aprendizados obtidos como resultado das experimentações realizadas ao longo do processo de desenvolvimento e experimentação.

O propósito do registro de requisitos conjecturais é explicitar as incertezas sobre características do software e o aprendizado obtido sobre elas, visando resolver ou aceitar essas incertezas. Idealmente, novos ciclos experimentais devem ser executados até que se tenha informações suficientes para resolver todas as incertezas. Porém, as incertezas podem ser aceitas quando forem obtidos indícios suficientes para uma tomada de decisão, ou de acordo com alguma diretriz específica do projeto ou da organização. Quando esse objetivo é atingido, e não há mais incertezas a serem experimentadas sobre um requisito conjectural, ele se torna elegível a ser convertido em um requisito funcional ou não funcional.

Semelhantemente, este trabalho denomina como "suposição de solução" cada proposta de solução a ser avaliada através de experimentação que vise gerar aprendizado

sobre as incertezas associadas a algum requisito conjectural. Novas suposições de solução devem ser propostas e experimentadas até que todos os requisitos conjecturais tenham suas incertezas resolvidas ou aceitas. Não faz parte do escopo deste trabalho a definição do planejamento nem da execução dos estudos experimentais, apenas a identificação das incertezas que serão experimentadas e o aprendizado gerado sobre elas.

O formato de descrição dos dois instrumentos toma por base tanto o formato de Histórias de Usuários dos métodos ágeis (apresentado na subseção 2.3) quanto os formatos de descrição de hipóteses (apresentados na subseção 2.5). Foram identificadas as partes de cada modelo mais apropriadas de acordo com o propósito de cada instrumento, realizando as alterações necessárias. Também foram acrescentadas informações necessárias que não foram encontradas em nenhum dos modelos, as quais serão descritas nas próximas subseções.

Após essa discussão inicial, esse capítulo apresenta e detalhada a proposta da ERC com seus dois instrumentos. Em seguida, a proposta é ilustrada em um cenário de desenvolvimento de software com EC imaginado.

3.2 ERC – Especificação de Requisitos Conjecturais

A ERC propõe a diferenciação entre requisitos conjecturais e suposições de solução, enfatizando a importância do registro dos requisitos conjecturais e a sua atualização através dos aprendizados gerados pelos resultados da experimentação das suposições de solução. Para isso, a ERC oferece dois instrumentos: o Formato de Escrita dos Requisitos Conjecturais (FERC) e o Quadro de Experimentação para Suposições de Solução (QESS). Esses instrumentos apoiam a elicitação, análise, documentação, atualização e priorização dos requisitos conjecturais e suas incertezas durante todo o desenvolvimento do software.

A ERC atua em apoio à EC recebendo como entradas os requisitos funcionais e não funcionais já definidos e as incertezas sobre as características do software. A partir dessas entradas, as incertezas são registradas utilizando o Formato de Escrita dos Requisitos Conjecturais. As suposições de solução são geradas com base nas incertezas registradas no FERC, sendo objeto do processo de experimentação. Os resultados da experimentação serão sumarizados no Quadro de Experimentação para Suposições de

Solução e produzirão atualizações no FERC. Quando tiverem sido executados ciclos experimentais suficientes para possibilitar a resolução ou aceitação de todas as incertezas, os requisitos conjecturais podem ser confirmados como novos requisitos funcionais ou não funcionais. Essa abordagem está ilustrada na Figura 2.

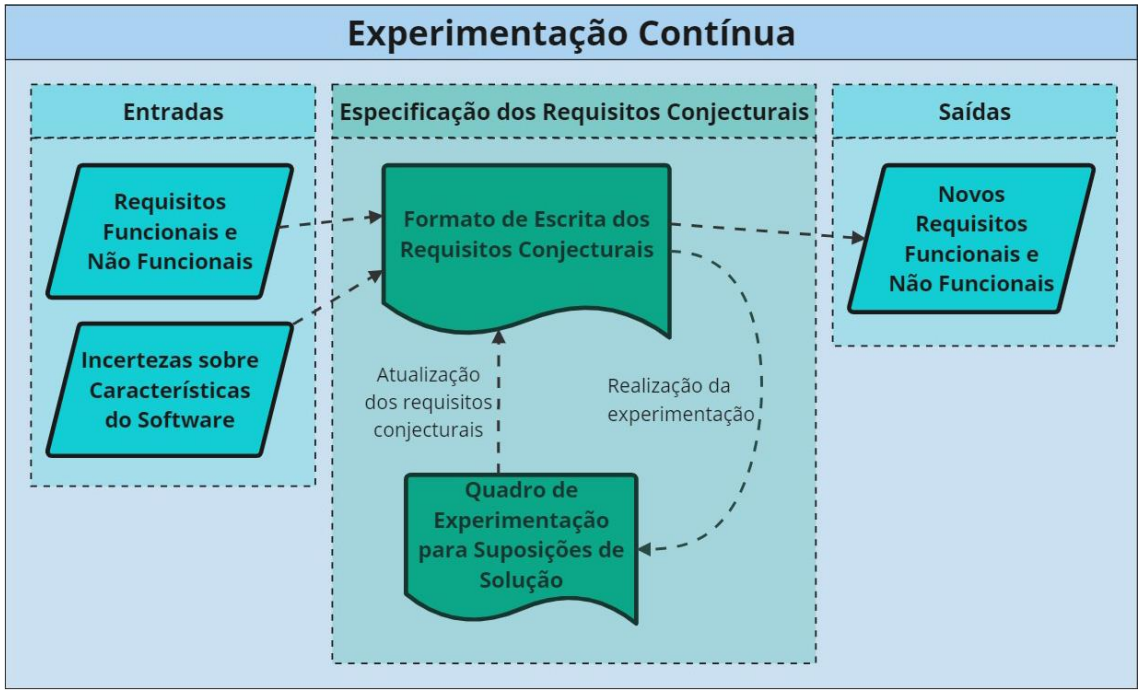

Figura 2 – Abordagem da ERC e seus instrumentos propostos.

3.2.1 Formato de Escrita dos Requisitos Conjecturais

Mesmo em um contexto de EC, nem todas as demandas de características do software são conjecturais. Dependendo do contexto, características de segurança, de ética, legais, entre outras, já possuem um alto nível de certeza para o desenvolvimento da sua solução, e não necessitam de uma estratégia de experimentação para que os seus resultados sejam avaliados. O mesmo se aplica quando todos os usuários finais são conhecidos, participam do processo de elicitação de requisitos e não há dúvidas sobre o contexto, o problema e nem sobre a solução que deve ser desenvolvida. Além disso, mesmo uma demanda com algum grau de incerteza pode ser definida como requisito funcional ou não funcional sem experimentação, se uma avaliação de custo e benefício mostrar que há vantagens nessa abordagem.

De forma geral, sempre haverá algum tipo de incerteza relacionada a uma característica do software, mesmo quando o nível de certeza é alto. A determinação sobre

se uma característica será tratada como requisito funcional, não-funcional ou como requisito conjectural é uma decisão de negócio, e deve ser tomada após uma avaliação de critérios relevantes para o contexto, como por exemplo, o nível de incerteza, os riscos envolvidos, o custo e o benefício.

O principal instrumento da ERC é o Formato de Escrita dos Requisitos Conjecturais (FERC), conforme mostrado na Quadro 7. Todas as incertezas sobre características do software relevantes para o projeto e que precisarem de experimentação para serem resolvidas devem ser registradas de forma agrupada de acordo com suas afinidades na forma de requisitos conjecturais.

Quadro 7 – Versão inicial do Formato de Escrita dos Requisitos Conjecturais

Requisitos Conjecturais	
ID	*RC[id]*
Descrição	**Espera-se que o sistema de software possua** *[comportamento desejado]* **De modo que** *[necessidade ou impacto positivo do atributo desejado]* **Porém, não sabemos:** ● *[incerteza associada com este requisito]* ● *[incerteza associada com este requisito]* ● ...
[O modelo se repete para cada requisito conjectural]	

O primeiro campo do FERC é o "ID", que é utilizado para que seja possível atribuir um identificador único a cada requisito conjectural, de modo que ele possa ser facilmente referenciado em outros artefatos. O segundo campo é a "Descrição", a qual é a principal característica de um requisito conjectural. A descrição deve ressaltar a expectativa e as incertezas do requisito conjectural. A formatação da descrição do FERC foi baseada nos modelos de descrição de hipóteses apresentados na subseção 2.5, buscando as informações mais adequadas de cada modelo para o conceito proposto de requisitos conjecturais, resultando em um modelo composto por três partes.

A primeira parte descreve o comportamento que o software deverá possuir. A segunda indica a necessidade desse atributo por parte dos usuários ou clientes, ou mesmo o impacto positivo que se espera que essa característica proporcione. Essas duas partes também estão presentes nos modelos de O'Reilly (O'Reilly, 2013), Weigel (Weigel, 2022) e Sullivan (Sullivan, 2021). É importante ressaltar que esse comportamento é uma necessidade identificada, e não uma proposta de solução para essa necessidade. Assim, ao descrever o requisito conjectural, deve-se identificar a causa primordial do

comportamento imaginado, de modo a identificar se esse comportamento é realmente a necessidade ou se é uma proposta de solução. Por exemplo, a utilização de um determinado modelo de sensor não deve ser registrada como requisito conjectural, pois é uma suposição de solução para um determinado comportamento desejado, que pode ser confiabilidade, estabilidade de medição, entre outros.

A última parte da descrição do FERC expressa o aspecto conjectural do requisito, pois trata das incertezas associadas a ele. Essa parte não está presente em nenhum dos modelos apresentados, mas é ela que possibilita a avaliação do requisito conjectural através dos diversos ciclos experimentais e o entendimento se o requisito conjectural pode ser aceito, convertido em requisito funcional/não funcional ou se ainda é necessária mais experimentação.

A partir dos aprendizados obtidos através dos ciclos experimentais, o requisito conjectural deve ser atualizado. Suas incertezas podem ser alteradas ou removidas, novas incertezas podem ser acrescentadas e até mesmo a sua descrição pode ser alterada. Quando os resultados dos experimentos gerarem a atualização do requisito conjectural de modo que todas as suas incertezas sejam removidas ou aceitas, esse requisito perde o seu aspecto conjectural, e torna-se elegível para ser convertido em um requisito do software. O tipo de requisito no qual ele será convertido dependerá das características do processo seguido e da organização, podendo ser, por exemplo, um requisito funcional, um requisito não funcional, uma história de usuário, um cenário, entre outros. O requisito conjectural também pode ser descartado, se essa for a decisão tomada pelos responsáveis. Nesse caso, ele pode ser excluído ou mantido com alguma observação para evitar a sua priorização.

O FERC pode ser utilizado de modo a integrar-se com processos de especificação de demandas, inclusive em contextos de desenvolvimento ágil, como em histórias de usuários. Porém, o FERC deve ser incluído juntamente com os demais requisitos do projeto, de modo a facilitar a sua priorização. Por exemplo, em contextos nos quais existe um documento de especificação de requisitos estruturado, com requisitos funcionais e não funcionais, os requisitos conjecturais podem ocupar uma nova seção. Já em contextos nos quais existe um *backlog* priorizado contendo as funcionalidades a serem desenvolvidas, os requisitos conjecturais podem ser incluídos entre essas funcionalidades, de acordo com a sua priorização.

3.2.2 Quadro de Experimentação para Suposições de Solução

No contexto da EC, a experimentação realiza um papel central para avaliar incertezas de características do software e determinar a aceitação ou rejeição das suposições de solução para construção do software. As suposições de solução são o objeto principal da atividade de experimentação, sendo implementadas, executadas e avaliadas para gerar aprendizado e possivelmente diminuir incertezas dos requisitos. A análise dos resultados do experimento indicará o aprendizado obtido, como ele reduziu ou não a incerteza associada e quais são as possibilidades de caminhos a serem seguidos na continuação do processo de desenvolvimento e experimentação.

O segundo instrumento proposto pela ERC é o Quadro de Experimentação para Suposições de Solução (QESS). Seu objetivo é resumir um ciclo experimental a fim de explicitar as incertezas de um requisito conjectural que foram o objeto principal desse ciclo, e os aprendizados resultantes dessa atividade. Não faz parte do QESS a maneira como a atividade de experimentação é realizada (planejamento, estratégia experimental selecionada, forma de execução, forma de análise, entre outras atividades associadas), apenas o resumo da expectativa de redução das incertezas e os resultados que estiverem relacionados com os requisitos conjecturais e suas incertezas.

Quadro 8 – Versão inicial do Quadro de Experimentação para Suposições de Solução

Ciclo Experimental CE *[id-ce]*
Expectativa do Ciclo Experimental
Esperamos que *[descrição da suposição de solução]* **Resulte na atualização das incertezas sobre** *[incertezas que serão avaliadas]* **Como resultado de** *[descrição da observação e análise que resultará na atualização das incertezas]*
Requisitos Conjecturais do Ciclo Experimental
● RC *[id]*: *[incerteza]* (versão *[n]*) ● RC *[id]*: *[incerteza]* (versão *[n]*) ● …
Resultados do Ciclo Experimental
● *[descrição do aprendizado dos resultados da suposição de solução executada nesta iteração para uma incerteza]* ● *[descrição do aprendizado dos resultados da suposição de solução executada nesta iteração para outra incerteza]* ● ...

O QESS é composto por três partes (Quadro 8). A primeira parte é a descrição da expectativa inicial do ciclo experimental, e deve descrever a solução que supostamente pode ajudar a esclarecer uma ou mais incertezas associadas com os requisitos conjecturais. Ela deve seguir um modelo de escrita cuja construção foi baseada nos modelos de escrita de hipóteses apresentados na subseção 2.5. Esse modelo é dividido em descrição da solução proposta, incertezas sobre as quais deverá ser obtido algum aprendizado e a forma de avaliação dos dados gerados pelo experimento sobre as incertezas.

A segunda parte do QESS é a listagem dos requisitos conjecturais e incertezas que serão avaliados no ciclo experimental. Estes devem referenciar as incertezas listadas no FERC. É recomendado inserir a versão do FERC para facilitar a leitura após a atualização dos requisitos conjecturais.

Dando seguimento ao exemplo anterior, teríamos a versão 1 do requisito conjectural contendo a incerteza sobre a confiabilidade dos dados obtidos pelo sensor. Após a experimentação de uma suposição de solução sobre um determinado modelo de sensor, os aprendizados seriam sumarizados no QESS, indicando se o sensor utilizado obteve o nível de confiabilidade desejado. A partir desse aprendizado, o requisito conjectural seria atualizado, gerando uma versão 2. No QESS, estaria referenciada a versão 1 do requisito conjectural, para indicar o estado do requisito no momento da experimentação realizada, conforme mostrado no Quadro 9. Se o sensor utilizado na suposição de solução obtiver um nível de confiabilidade aceitável e não existir mais nenhuma incerteza associada, esse requisito conjectural pode ser convertido em um requisito não funcional, indicando o modelo de sensor que deverá ser utilizado no sistema de software.

A terceira parte do QESS é o registro do aprendizado resultante do ciclo experimental em relação às incertezas dos requisitos conjecturais referenciados nessa experimentação, e mostra como as incertezas foram impactadas por esse aprendizado. Esse aprendizado deve ser registrado separadamente para cada incerteza, de modo a facilitar a atualização dos requisitos conjecturais. Ele deve também explicar quais atualizações foram feitas no requisito conjectural.

Quadro 9 – Exemplo de preenchimento do QESS

Ciclo Experimental CE01
Expectativa do Ciclo Experimental
Esperamos que a utilização de um sensor do tipo X **Resulte na atualização das incertezas sobre** a confiabilidade dos dados medidos, **Como resultado** da observação dos dados gerados pelo sensor, em comparação com dados da fonte Y.
Requisitos Conjecturais do Ciclo Experimental
● RC01: Confiabilidade dos dados gerados pelo sensor X (versão 1)
Resultados do Ciclo Experimental
● Os dados gerados pelo sensor X mostraram que as medições são idênticas aos dados gerados pela fonte Y.

3.3 Cenário ilustrativo da Aplicação da Especificação de Requisitos Conjecturais

Para ilustrar a aplicação da ERC em um processo de desenvolvimento de software com EC, considere o cenário hipotético apresentado na Figura 3, adaptado do *RIGHT model for Continuous Experimentation* (Fagerholm *et al.*, 2017). A aplicação dos instrumentos da ERC nesse cenário poderia gerar uma alteração no processo, resultando em um novo cenário, conforme mostrado na Figura 4.

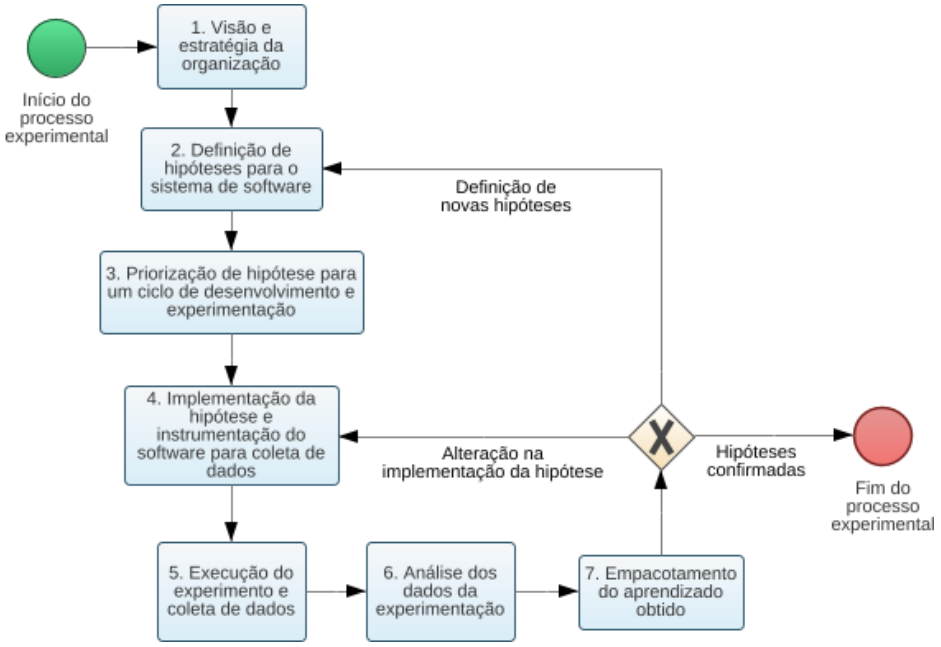

Figura 3 – Cenário ilustrativo no formato BPMN do processo de desenvolvimento adaptado de (Fagerholm *et al.*, 2017).

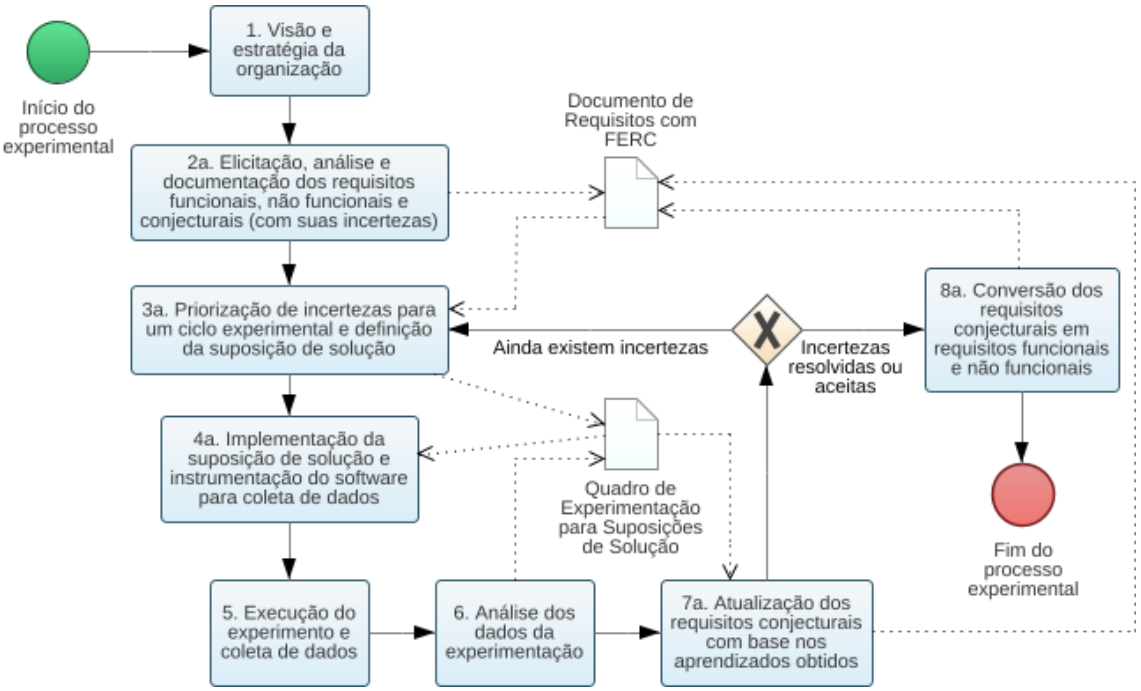

Figura 4 – Adaptação do cenário ilustrativo no formato BPMN, com a inclusão dos instrumentos da ERC.

Nesse exemplo, o processo de desenvolvimento de software com experimentação é iniciado a partir da visão e da estratégia da organização (etapa 1), resultando na definição de hipóteses a serem implementadas no sistema de software (etapa 2). Na adaptação para a ERC, ao invés de definir hipóteses, são definidos os requisitos do software, tanto os requisitos funcionais e não funcionais quanto os requisitos conjecturais, com suas incertezas (etapa 2a). Esses requisitos são registrados no mesmo documento, sendo que os requisitos conjecturais são registrados com a utilização do FERC.

Na terceira atividade, sem a ERC, a priorização era realizada a partir das hipóteses identificadas (etapa 3). Com a aplicação da ERC, essa priorização é realizada sobre as incertezas dos requisitos conjecturais a partir do documento de requisitos, identificando quais incertezas serão alvo do próximo ciclo experimental. Com base nessa priorização de incertezas, é escolhida uma suposição de solução para potencialmente reduzir essas incertezas ou resolvê-las (etapa 3a). A suposição de solução escolhida é registrada em um novo documento, de acordo com o modelo do QESS.

As próximas três etapas são as mesmas nos dois cenários: implementação, execução do experimento e análise dos dados gerados (etapas 4, 4a, 5 e 6). A diferença é que, no cenário com a ERC, os aprendizados obtidos sobre as incertezas são registrados

também no QESS. Após a análise dos dados de um ciclo experimental, sem a ERC, o aprendizado obtido era empacotado sem um formato de apoio à gestão do requisito conjectural (etapa 7). Com a adaptação para a ERC, esse aprendizado, que já está registrado no QESS, é utilizado para atualizar os requisitos conjecturais no documento de requisitos, com a utilização do FERC (etapa 7a).

Ao final de um ciclo experimental, após o registro dos aprendizados, a diferença entre os dois cenários é grande. No cenário sem a ERC, é preciso realizar uma tomada de decisão sobre os aprendizados para validar a hipótese experimentada, tendo três caminhos possíveis. Se a hipótese não for rejeitada, mas também não for totalmente satisfeita, a hipótese é alterada e é realizada uma nova implementação e instrumentação. Caso a hipótese seja rejeitada, o processo volta para a etapa de definição de hipóteses, pois é preciso definir uma nova hipótese a ser implementada e experimentada. Porém, no caso de a hipótese ser satisfeita e não existir mais hipóteses a serem experimentadas, o processo é finalizado. Ao aplicar a ERC, o final do ciclo experimental é alterado significativamente, devido ao registro e à possibilidade de alteração das incertezas identificadas.

Ao invés de existirem dois caminhos possíveis quando ainda há necessidade de experimentação, existe apenas um, pois enquanto ainda houver incertezas o processo sempre retornará para a etapa de priorização de incertezas para a definição de uma nova suposição de solução, com o objetivo de reduzir as incertezas priorizadas. Com o registro dos requisitos conjecturais, a possibilidade de visualização e gerenciamento das incertezas é potencializada, permitindo que mais incertezas sejam identificadas e permaneçam visíveis para os envolvidos no projeto, evitando a perda de informações. O processo permanece com tantos ciclos experimentais quanto forem necessários até que todas as incertezas sejam resolvidas, aceitas ou descartadas ou até que seja decidido interromper a abordagem experimental. Quando isso ocorre, os requisitos conjecturais que não forem cancelados são convertidos em requisitos funcionais ou não funcionais (etapa 8a) e o processo experimental é finalizado.

A partir desse cenário ilustrativo, é possível ver em quais momentos os instrumentos da ERC visam apoiar o processo de desenvolvimento de software com experimentação contínua, fornecendo visibilidade para as incertezas do projeto e

facilitando a priorização, a criação de suposições de solução e a tomada de decisão sobre a manutenção das características do software.

3.4 Diretrizes para Aplicação da Proposta

Resumidamente, a proposta da ERC pode ser apresentada por meio das seguintes diretrizes de uso:

- Ao identificar os requisitos do sistema de software, identificar também as incertezas sobre as características do software;
- Agrupar as incertezas identificadas em requisitos conjecturais, registrando-os com a utilização do FERC no mesmo artefato de requisitos onde se encontram os demais requisitos do sistema de software;
- Identificar as suposições de solução que podem supostamente reduzir ou resolver as incertezas identificadas;
- Considerar os requisitos conjecturais juntamente com os demais requisitos e suposições de solução para priorização das características a serem implementadas e experimentadas ao início de cada ciclo experimental;
- Registrar o resumo de cada ciclo experimental em um novo documento no formato do QESS, com o objetivo, incertezas experimentadas e aprendizados obtidos (informações extraídas dos artefatos produzidos conforme a metodologia de experimentação escolhida);
- Registrar o aprendizado obtido sobre as incertezas juntamente com os seus respectivos requisitos conjecturais ao final de cada ciclo experimental, de acordo com o modelo do FERC;
- Atualizar o registro das incertezas dos requisitos conjecturais ao final de cada ciclo experimental, a partir dos aprendizados obtidos sobre as incertezas;
- Registrar novas incertezas identificadas como resultado do aprendizado obtido ao final de cada ciclo experimental, sempre que existir;
- Remover incertezas resolvidas ou aceitas como resultado do aprendizado obtido ao final de cada ciclo experimental, sempre que ocorrer; e
- Converter requisitos conjecturais em outro tipo de requisito sempre que eles tiverem todas as suas incertezas resolvidas ou aceitas.

3.5 Conclusão do Capítulo

Neste capítulo, foi apresentada a proposta de Especificação de Requisitos Conjecturais (ERC), com seus dois instrumentos: o Formato de Escrita dos Requisitos Conjecturais (FERC) e o Quadro de Experimentação para Suposições de Solução (QESS). Conforme ilustrado por meio de um cenário de desenvolvimento de software hipotético, a ERC visa apoiar a elicitação e a atualização dos requisitos conjecturais e facilitar a especificação, avaliação e atualização das incertezas associadas aos requisitos do projeto de software no contexto da EC em engenharia de software. O registro das incertezas e sua atualização a partir dos aprendizados obtidos é importante para apoiar a priorização do desenvolvimento e da experimentação, além de facilitar a transferência de informações sobre os requisitos e incertezas do sistema de software. Esse registro é especialmente útil em contextos nos quais os requisitos não podem ser completamente definidos e explicitados antes da utilização do software por parte dos clientes ou usuários.

4 Prova de Conceito

Este capítulo apresenta uma prova de conceito sobre a proposta da Especificação de Requisitos Conjecturais através do uso de seus dois instrumentos ao longo dos ciclos experimentais do sistema de software OxímetroIoT, desenvolvido na Universidade Federal do Rio de Janeiro.

4.1 Introdução

Com o objetivo de realizar uma avaliação inicial da proposta da Especificação de Requisitos Conjecturais (ERC) e dos seus instrumentos, foi conduzida uma prova de conceito por meio da aplicação dos instrumentos para especificação de informações obtidas a partir do histórico de ciclos experimentais previamente realizadas no projeto do sistema de software OxímetroIoT, conduzido pela equipe de Engenharia de Software Experimental da Universidade Federal do Rio de Janeiro.

O OxímetroIoT é um projeto de desenvolvimento de um sistema de software IoT (Internet das Coisas) que visa a criação de uma solução computacional com equipamentos de baixo custo para monitoramento de sinais vitais (temperatura, oxigenação e frequência cardíaca) de pacientes em enfermarias de hospitais. Seu objetivo é apoiar a tomada de decisão em tempo hábil pelos profissionais da saúde sobre os cuidados com o paciente. De forma simplificada, ele é composto por dispositivos IoT com sensores de sinais vitais em contato com o paciente. Os dados coletados pelos dispositivos são transmitidos por meio de um *broker* para um *dashboard* para serem exibidos. A necessidade deste sistema surgiu na observação da carência de equipamentos desse tipo em enfermarias nos hospitais públicos da UFRJ devido ao alto volume de internações durante a pandemia de Covid-19.

O sistema de software OxímetroIoT está sendo desenvolvido com apoio de experimentação contínua. O projeto já executou iterações para a construção dos dispositivos IoT e do software, realizando ajustes a partir dos resultados da experimentação, chegando a uma versão que foi apresentada para profissionais da saúde em dois contextos diferentes: de hospital universitário e de unidade de pronto atendimento.

4.2 Planejamento

Nesta prova de conceito foi estabelecido que o Formato de Escrita dos Requisitos Conjecturais (FERC) e o Quadro de Experimentação para Suposições de Solução (QESS) seriam preenchidos de maneira retroativa pelo pesquisador a partir de entrevistas com dois participantes do projeto. Esses participantes possuíam papéis de liderança e atuaram no desenvolvimento dos componentes de hardware e de software do sistema. Ambos participaram desde o início do projeto. O objetivo do estudo está descrito na Quadro 10.

Quadro 10 – Objetivo da prova de conceito, de acordo com o paradigma GQM (Goal/Question/Metric - Objetivo/Questão/Métrica) (Basili *et al.*, 1994)

Objetivo do estudo	Analisar os instrumentos da ERC (FERC e QESS).
Propósito	Caracterizar qualitativamente a viabilidade do FERC e do QESS para o registro de requisitos conjecturais, com suas incertezas e aprendizados.
Ponto de vista	Pesquisador em Engenharia de Software.
Contexto	Utilizar os instrumentos do ERC para capturar em retrospectiva os requisitos conjecturais do processo de experimentação contínua do projeto do sistema de software OxímetroIoT.

4.3 Execução

A partir das entrevistas realizadas com os participantes do projeto, foi estruturada uma narrativa cronológica dos ciclos experimentais executados e dos resultados obtidos em cada ciclo, gerando aprendizados para os próximos ciclos. No momento da realização desta prova de conceito, haviam sido realizados cinco ciclos experimentais. Essa narrativa permitiu a coleta de informações sobre os requisitos, as incertezas, as experimentações realizadas e o aprendizado obtido durante o projeto. Essas informações foram então organizadas pelo pesquisador com a utilização dos instrumentos da ERC, a saber: o Formato de Escrita dos Requisitos Conjecturais (FERC) e o Quadro de Experimentação para Suposições de Solução (QESS).

4.3.1 Ciclo Experimental 1

Juntamente com as entrevistas, foi recuperado o documento de requisitos do sistema de software OxímetroIoT, contendo os requisitos funcionais e não funcionais elicitados no início do projeto. O objetivo deste trabalho não é tratar sobre esses tipos de requisitos, portanto, eles não serão listados aqui. Ao final da lista de requisitos, foi acrescentado um único requisito conjectural, referente à incerteza sobre os equipamentos que poderiam ser utilizados e que possuíssem baixo custo (Quadro 11). O primeiro ciclo experimental foi idealizado para avaliar essa incerteza, propondo uma suposição de solução com a utilização de um dispositivo do tipo clipe de dedo para encapsulamento dos sensores responsáveis pela medição dos sinais vitais (Quadro 12).

Ao preencher o QESS, foi identificada a necessidade de referenciar a incerteza que estava sendo alvo do ciclo experimental. Portanto, foi acrescentado um identificador único para a incerteza, de modo que, quando forem listadas novas incertezas, possa ser identificada facilmente a que participou do ciclo e as que não participaram.

O resultado dessa utilização foi a rejeição da suposição de solução, pois ela não resolveu a Incerteza In01a, e gerou novas incertezas sobre as características do software, que foram inseridas como novos requisitos conjecturais na lista de requisitos do projeto (Quadro 13). Ao registrar os novos requisitos conjecturais, porém, foi identificada a necessidade de registrar o aprendizado sobre a incerteza juntamente com o seu respectivo requisito conjectural. Essa necessidade teve origem no entendimento de que novas suposições de solução sobre uma incerteza devem ser propostas a partir do aprendizado obtido nos ciclos experimentais previamente executados, de modo a evitar desperdício de tempo com suposições já experimentadas. Assim, foi criada uma seção dentro do campo de descrição do requisito conjectural, com o objetivo de registrar os aprendizados já obtidos sobre ele.

Quadro 11 – Documento de Requisitos do Projeto OxímetroIoT versão 1 (Ciclo Experimental 1)

Sistema OxímetroIoT – Lista de Requisitos	
Requisitos Conjecturais	
ID	RC01
Descrição	**Espera-se que o sistema de software possua** equipamentos de baixo custo, **De modo que** o produto possa ser vendido por um valor mais baixo do que o de outros produtos atualmente no mercado com funções semelhantes. **Porém, não sabemos:** ● **Incerteza In01a:** quais equipamentos (sensores, vestíveis, cabos, conectores e display) são funcionais e possuem o menor custo.

Quadro 12 – Quadro de Experimentação para Suposições de Solução do Projeto OxímetroIoT no Ciclo Experimental 1

OxímetroIoT – Ciclo Experimental CE1
Expectativa do Ciclo Experimental
Esperamos que utilizar um clipe de dedo para obter dados sobre a oxigenação, temperatura e frequência cardíaca de um paciente para exibição em um display controlado por um processador de baixo custo NodeMCU **Resulte na atualização das incertezas sobre** a configuração do dispositivo de baixo custo que será utilizado para a construção do sistema de software, **Como resultado da** observação do funcionamento do oxímetro com clipe de dedo e dos dados gerados.
Requisitos Conjecturais do Ciclo Experimental
● RC01: Incerteza In01a (documento de requisitos versão 1)
Resultados do Ciclo Experimental
● **Incerteza In1a:** O clipe de dedo possui baixo custo e consegue realizar a medição da oxigenação, temperatura e frequência cardíaca que serão exibidos no display, mas ele ainda precisa de mais testes para verificar a confiabilidade dessa medição. Porém, a montagem do clipe de dedo é complexa, pois ele depende de dois cabos de dados, um digital para a oxigenação e outro analógico para temperatura e frequência cardíaca. Isso resultou na atualização da Incerteza In01a, na criação do RC02 com as incertezas In02a e In02b e na criação do RC03 com a incerteza In03a.

4.3.2 Ciclo Experimental 2

A partir da nova lista de requisitos conjecturais atualizada como resultado do Ciclo Experimental 1 (Quadro 13), foi pensada e desenvolvida uma nova suposição de solução. Essa suposição deveria testar a incerteza sobre a facilidade de montagem (Incerteza In02a) através da utilização de um único cabo de dados (Quadro 14), em contraste com os dois cabos do ciclo anterior.

Quadro 13 – Documento de Requisitos (mostrando apenas a seção de Requisitos Conjecturais) do Projeto OxímetroIoT versão 2 (Ciclo Experimental 2)

Requisitos Conjecturais	
ID	RC01
Descrição	**Espera-se que o sistema de software possua** equipamentos de baixo custo, **De modo que** o produto possa ser vendido por um valor mais baixo do que o de outros produtos atualmente no mercado com funções semelhantes. **Porém, não sabemos:** ● **Incerteza In01a:** quais equipamentos (sensores, vestíveis, cabos, conectores e display) são funcionais e possuem o menor custo. **Mas já aprendemos que:** ● **CE1:** O clipe de dedo possui baixo custo, porém sua montagem é complexa e não foi possível testar a sua confiabilidade.
ID	RC02
Descrição	**Espera-se que o sistema de software possua** um equipamento de fácil montagem **De modo que** o equipamento possa ser montado de maneira rápida por pessoas sem conhecimento de eletrônica. **Porém, não sabemos:** ● **Incerteza In02a:** quais modelos de cabos e conectores facilitam a montagem. ● **Incerteza In02b:** o tempo aceitável de montagem. **Mas já aprendemos que:** ● Não experimentado
ID	RC03
Descrição	**Espera-se que o sistema de software possua** confiabilidade, **De modo que** a medição dos sinais seja realizada sem interferência de iluminação externa. **Porém, não sabemos:** ● **Incerteza In03a:** qual tipo de dispositivo permite a medição sem interferências prejudiciais. **Mas já aprendemos que:** ● Não experimentado

Quadro 14 – Quadro de Experimentação para Suposições de Solução do Projeto OxímetroIoT no Ciclo Experimental 2

OxímetroIoT – Ciclo Experimental CE2
Expectativa do Ciclo Experimental
Esperamos que utilizar um clipe de dedo para obter dados sobre a oxigenação, temperatura e frequência cardíaca de um paciente para exibição em um display controlado por um processador de baixo custo NodeMCU por meio de um único cabo de dados **Resulte na atualização das incertezas sobre** a facilidade de montagem do dispositivo de baixo custo que será utilizado para a construção do sistema de software, **Como resultado da** observação do funcionamento do oxímetro com clipe de dedo e um único cabo de dados e dos dados gerados.
Requisitos Conjecturais do Ciclo Experimental
● RC02: Incerteza In02a (lista de requisitos versão 2)
Resultados do Ciclo Experimental
● **Sobre a incerteza In02a:** O cabo de dados único facilitou a montagem, mas ele ainda possui a inconveniência de ser um cabo muito espesso, devido ao sensor analógico. Além disso, foi verificado que a medição da temperatura sofreu interferências devido à entrada

de luz no clipe de dedo, o que afetou os dados medidos. Isso resultou na atualização das incertezas In01a, In02a e In03a.

4.3.3 Ciclo Experimental 3

Diante dos aprendizados obtidos nos ciclos experimentais anteriores, a suposição de solução utilizando um dispositivo do tipo clipe de dedo foi descartada e os requisitos conjecturais foram atualizados (Quadro 15). Uma nova suposição de solução foi então proposta para experimentar a incerteza sobre a confiabilidade dos dados medidos utilizando outro tipo de dispositivo (Quadro 16).

Quadro 15 – Documento de Requisitos (mostrando apenas a seção de Requisitos Conjecturais) do Projeto OxímetroIoT versão 3 (Ciclo Experimental 3)

Requisitos Conjecturais	
ID	RC01
Descrição	**Espera-se que o sistema de software possua** equipamentos de baixo custo, **De modo que** o produto possa ser vendido por um valor mais baixo do que o de outros produtos atualmente no mercado com funções semelhantes. **Porém, não sabemos:** ● **Incerteza In01a:** quais equipamentos (sensores, vestíveis, cabos, conectores e display) são funcionais e possuem o menor custo. **Mas já aprendemos que:** ● **CE1 e CE2:** O clipe de dedo possui baixo custo, porém mostrou-se inadequado devido ao não cumprimento de outros requisitos.
ID	RC02
Descrição	**Espera-se que o sistema de software possua** um equipamento de fácil montagem **De modo que** o equipamento possa ser montado de maneira rápida por pessoas sem conhecimento de eletrônica. **Porém, não sabemos:** ● **Incerteza In02a:** quais modelos de cabos e conectores facilitam a montagem. ● **Incerteza In02b:** o tempo aceitável de montagem. **Mas já aprendemos que:** ● **CE2:** Um cabo de dados único para os dois sensores mostrou-se mais fácil de montar do que dois cabos, porém, a existência de um sensor analógico faz com que seja necessária a utilização de um cabo espesso.
ID	RC03
Descrição	**Espera-se que o sistema de software possua** confiabilidade, **De modo que** a medição dos sinais seja realizada sem interferência de iluminação externa. **Porém, não sabemos:** ● **Incerteza In03a:** qual tipo de dispositivo permite a medição sem interferências prejudiciais. **Mas já aprendemos que:** ● **CE2:** O clipe de dedo é prejudicado pela interferência de luz externa na medição da temperatura.

Quadro 16 – Quadro de Experimentação para Suposições de Solução do Projeto OxímetroIoT no Ciclo Experimental 3

OxímetroIoT – Ciclo Experimental CE3
Expectativa do Ciclo Experimental
Esperamos que utilizar uma pulseira elástica com dois compartimentos de sensores para obter dados sobre a oxigenação, temperatura e frequência cardíaca de um paciente para exibição em um display controlado por um processador de baixo custo NodeMCU **Resulte na atualização das incertezas sobre** a confiabilidade da medição dos sinais vitais (medição sem interferências externas) do dispositivo de baixo custo que será utilizado para a construção do sistema de software, **Como resultado da** observação do funcionamento do oxímetro com a pulseira elástica e dos dados gerados.
Requisitos Conjecturais do Ciclo Experimental
● RC3: Incerteza In03a (lista de requisitos versão 3)
Resultados do Ciclo Experimental
● **Sobre a incerteza In03a:** A pulseira com os dois compartimentos para os sensores obteve medições dos sinais vitais sem interferências externas. Porém, foi observado que a pulseira não permanece estável no braço do paciente, o que faz com que os valores medidos fiquem instáveis. Além disso, foi identificado que, devido a utilização de um sensor de temperatura e frequência cardíaca digital, é possível substituir o cabo de dados espesso por um cabo de dados digital com conector RJ11. Isso resultou na atualização das incertezas In01a, In02a e In03a, e na criação do RC04 com a incerteza In04a.

4.3.4 Ciclo Experimental 4

Os bons resultados do ciclo experimental 3 permitiram a manutenção da solução que utilizava uma pulseira elástica com dois compartimentos para sensores, resultando na atualização dos requisitos conjecturais (Quadro 17). Assim, uma nova suposição de solução foi proposta para avaliar a incerteza sobre a facilidade de montagem do dispositivo (Quadro 18).

Quadro 17 – Documento de Requisitos (mostrando apenas a seção de Requisitos Conjecturais) do Projeto OxímetroIoT versão 4 (Ciclo Experimental 4)

Requisitos Conjecturais	
ID	RC01
Descrição	**Espera-se que o sistema de software possua** equipamentos de baixo custo, **De modo que** o produto possa ser vendido por um valor mais baixo do que o de outros produtos atualmente no mercado com funções semelhantes. **Porém, não sabemos:** ● **Incerteza In01a:** quais equipamentos (sensores, vestíveis, cabos, conectores e display) são funcionais e possuem o menor custo. **Mas já aprendemos que:** ● **CE1 e CE2:** O clipe de dedo possui baixo custo, porém mostrou-se inadequado devido ao não cumprimento de outros requisitos.

	● **CE3:** A pulseira de elástico possui baixo custo, porém mostrou-se inadequada devido ao não cumprimento de outros requisitos.
ID	RC02
Descrição	**Espera-se que o sistema de software possua** um equipamento de fácil montagem **De modo que** o equipamento possa ser montado de maneira rápida por pessoas sem conhecimento de eletrônica. **Porém, não sabemos:** ● **Incerteza In02a:** quais modelos de cabos e conectores facilitam a montagem. ● **Incerteza In02b:** o tempo aceitável de montagem. **Mas já aprendemos que:** ● **CE2:** Um cabo de dados único para os dois sensores mostrou-se mais fácil de montar do que dois cabos, porém, a existência de um sensor analógico faz com que seja necessária a utilização de um cabo espesso. ● **CE3:** A utilização de dois sensores digitais permite a utilização de apenas um cabo de dados digital com conector RJ11, o que facilita a montagem.
ID	RC03
Descrição	**Espera-se que o sistema de software possua** confiabilidade, **De modo que** a medição dos sinais seja realizada sem interferência de iluminação externa. **Porém, não sabemos:** ● **Incerteza In03a:** qual tipo de dispositivo permite a medição sem interferências prejudiciais. **Mas já aprendemos que:** ● **CE2:** O clipe de dedo é prejudicado pela interferência de luz externa na medição da temperatura. ● **CE3:** A pulseira com dois compartimentos para os sensores protege a medição de interferências externas.
ID	RC04
Descrição	**Espera-se que o sistema de software possua** estabilidade, **De modo que** a medição dos sinais se mantenha consistente em um mesmo paciente durante longo período, considerando os movimentos do corpo. **Porém, não sabemos:** ● **Incerteza In04a:** quais modelos de sensores garantem estabilidade de medição em períodos longos, considerando os movimentos do corpo. **Mas já aprendemos que:** ● **CE3:** A pulseira elástica não mantém os sensores estáveis no braço do paciente.

Quadro 18 – Quadro de Experimentação para Suposições de Solução do Projeto OxímetroIoT no Ciclo Experimental 4

OxímetroIoT – Ciclo Experimental CE4
Expectativa do Ciclo Experimental
Esperamos que utilizar uma pulseira elástica com dois compartimentos de sensores para obter dados sobre a oxigenação, temperatura e frequência cardíaca de um paciente para exibição em um display controlado por um processador de baixo custo NodeMCU por meio de um único cabo de dados digital com conector RJ11 **Resulte na atualização das incertezas sobre** a facilidade de montagem do dispositivo de baixo custo que será utilizado para a construção do sistema de software, **Como resultado da** observação do funcionamento do oxímetro com a pulseira elástica e dos dados gerados.
Requisitos Conjecturais do Ciclo Experimental
● RC03: Incerteza In02a (lista de requisitos versão 4)
Resultados do Ciclo Experimental
● **Sobre a incerteza In02a:** O cabo de dados digital com conector RJ11 gerou maior facilidade de montagem e na correta transmissão dos dados dos sensores para o display. Isso resultou na atualização da incerteza In02a.

4.3.5 Ciclo Experimental 5

O resultado do ciclo experimental 4 resultou na redução da incerteza sobre a facilidade de montagem do dispositivo (Quadro 19), mas ainda existia uma incerteza ameaçando diretamente a suposição de solução da pulseira elástica. Portanto, foi proposta uma nova suposição de solução utilizando uma pulseira rígida (Quadro 20).

Quadro 19 – Documento de Requisitos (mostrando apenas a seção de Requisitos Conjecturais) do Projeto OxímetroIoT versão 5 (Ciclo Experimental 5)

Requisitos Conjecturais	
ID	RC01
Descrição	**Espera-se que o sistema de software possua** equipamentos de baixo custo, **De modo que** o produto possa ser vendido por um valor mais baixo do que o de outros produtos atualmente no mercado com funções semelhantes. **Porém, não sabemos:** ● **Incerteza In01a:** quais equipamentos (sensores, vestíveis, cabos, conectores e display) são funcionais e possuem o menor custo. **Mas já aprendemos que:** ● **CE1 e CE2:** O clipe de dedo possui baixo custo, porém mostrou-se inadequado devido ao não cumprimento de outros requisitos. ● **CE3:** A pulseira de elástico possui baixo custo, porém mostrou-se inadequada devido ao não cumprimento de outros requisitos.
ID	RC02
Descrição	**Espera-se que o sistema de software possua** um equipamento de fácil montagem **De modo que** o equipamento possa ser montado de maneira rápida por pessoas sem conhecimento de eletrônica.

	Porém, não sabemos: ● **Incerteza In02a:** quais modelos de cabos e conectores facilitam a montagem. ● **Incerteza In02b:** o tempo aceitável de montagem. **Mas já aprendemos que:** ● **CE2:** Um cabo de dados único para os dois sensores mostrou-se mais fácil de montar do que dois cabos, porém, a existência de um sensor analógico faz com que seja necessária a utilização de um cabo espesso. ● **CE3 e CE4:** A utilização de dois sensores digitais permite a utilização de apenas um cabo de dados digital com conector RJ11, o que facilita a montagem.
ID	RC03
Descrição	**Espera-se que o sistema de software possua** confiabilidade, **De modo que** a medição dos sinais seja realizada sem interferência de iluminação externa. **Porém, não sabemos:** ● **Incerteza In03a:** qual tipo de dispositivo permite a medição sem interferências prejudiciais. **Mas já aprendemos que:** ● **CE2:** O clipe de dedo é prejudicado pela interferência de luz externa na medição da temperatura. ● **CE3:** A pulseira com dois compartimentos para os sensores protege a medição de interferências externas.
ID	RC04
Descrição	**Espera-se que o sistema de software possua** estabilidade, **De modo que** a medição dos sinais se mantenha consistente em um mesmo paciente durante longo período, considerando os movimentos do corpo. **Porém, não sabemos:** ● **Incerteza In04a:** quais modelos de sensores garantem estabilidade de medição em períodos longos, considerando os movimentos do corpo. **Mas já aprendemos que:** ● **CE3:** A pulseira elástica não mantém os sensores estáveis no braço do paciente.

Quadro 20 – Quadro de Experimentação para Suposições de Solução do Projeto OxímetroIoT no Ciclo Experimental 5

OxímetroIoT – Ciclo Experimental CE5
Expectativa do Ciclo Experimental
Esperamos que utilizar uma pulseira rígida em forma de meia lua com dois compartimentos de sensores para obter dados sobre a oxigenação, temperatura e frequência cardíaca de um paciente para exibição em um display controlado por um processador de baixo custo NodeMCU por meio de um único cabo de dados digital com conector RJ11 **Resulte na atualização das incertezas sobre** a estabilidade dos sensores no braço do paciente do dispositivo de baixo custo que será utilizado para a construção do sistema de software, **Como resultado da** observação do funcionamento do oxímetro com a pulseira rígida e dos dados gerados.
Requisitos Conjecturais do Ciclo Experimental
● RC04: Incerteza In04a (lista de requisitos versão 5)
Resultados do Ciclo Experimental
● **Sobre a incerteza In04a:** A pulseira rígida não permanece estável no braço do paciente devido à falta de um fecho em seu lado aberto. Isso resultou na atualização da incerteza In04a.

4.4 Análise dos Resultados

O resultado do ciclo experimental 5 gerou uma nova atualização dos requisitos conjecturais (Quadro 21). A prova de conceito foi encerrada nessa etapa, ao término dos dados históricos, pois o projeto encontrava-se no momento no planejamento do sexto ciclo experimental. As discussões com os participantes geraram o entendimento de que o objetivo de analisar os instrumentos da ERC de forma qualitativa foi atingido de maneira satisfatória. Durante a execução do estudo, foram identificadas melhorias a serem acrescentadas aos instrumentos (mostradas na subseção 4.5). A conclusão da prova de conceito foi que os instrumentos FERC e QESS, após a implementação dessas melhorias, poderiam ser viáveis para a especificação dos requisitos conjecturais, com suas incertezas e aprendizados, facilitando a recuperação e a comparação dessas informações.

Esta prova de conceito visou avaliar a utilidade dos resultados gerados pelos instrumentos da ERC. O aprendizado obtido foi que a ERC pode prover resultados úteis ao oferecer um instrumento para registro dos requisitos conjecturais (FERC) passível de atualizações como resultado dos ciclos experimentais, e ao oferecer um instrumento para sumarização dos resultados dos ciclos experimentais (QESS), explicitando os aprendizados sobre as incertezas dos requisitos conjecturais e gerando maior visualização sobre as informações que não estavam registradas em nenhum documento do projeto. Esses resultados possuem limitações devido ao contexto em que a prova de conceito foi realizada (conforme subseção 4.6) e necessitam de mais estudos de viabilidade para observação e evolução da proposta. Porém, foram obtidos bons indicadores sobre a eventual viabilidade e utilidade dos instrumentos.

Quadro 21 – Documento de Requisitos (mostrando apenas a seção de Requisitos Conjecturais) do Projeto OxímetroIoT versão 6

Requisitos Conjecturais	
ID	RC01
Descrição	**Espera-se que o sistema de software possua** equipamentos de baixo custo, **De modo que** o produto possa ser vendido por um valor mais baixo do que o de outros produtos atualmente no mercado com funções semelhantes. **Porém, não sabemos:** ● **Incerteza In01a:** quais equipamentos (sensores, vestíveis, cabos, conectores e display) são funcionais e possuem o menor custo. **Mas já aprendemos que:** ● **CE1 e CE2:** O clipe de dedo possui baixo custo, porém mostrou-se inadequado devido ao não cumprimento de outros requisitos.

	● **CE3:** A pulseira de elástico possui baixo custo, porém mostrou-se inadequada devido ao não cumprimento de outros requisitos.
ID	RC02
Descrição	**Espera-se que o sistema de software possua** um equipamento de fácil montagem **De modo que** o equipamento possa ser montado de maneira rápida por pessoas sem conhecimento de eletrônica. **Porém, não sabemos:** ● **Incerteza In02a:** quais modelos de cabos e conectores facilitam a montagem. ● **Incerteza In02b:** o tempo aceitável de montagem. **Mas já aprendemos que:** ● **CE2:** Um cabo de dados único para os dois sensores mostrou-se mais fácil de montar do que dois cabos, porém, a existência de um sensor analógico faz com que seja necessária a utilização de um cabo espesso. ● **CE3 e CE4:** A utilização de dois sensores digitais permite a utilização de apenas um cabo de dados digital com conector RJ11, o que facilita a montagem.
ID	RC03
Descrição	**Espera-se que o sistema de software possua** confiabilidade, **De modo que** a medição dos sinais seja realizada sem interferências externas. **Porém, não sabemos:** ● **Incerteza In03a:** qual tipo de dispositivo permite a medição sem interferências prejudiciais. **Mas já aprendemos que:** ● **CE2:** O clipe de dedo é prejudicado pela interferência de luz externa na medição da temperatura. ● **CE3:** A pulseira com dois compartimentos para os sensores protege a medição de interferências externas.
ID	RC04
Descrição	**Espera-se que o sistema de software possua** estabilidade, **De modo que** a medição dos sinais se mantenha consistente em um mesmo paciente durante longo período, considerando os movimentos do corpo. **Porém, não sabemos:** ● **Incerteza In04a:** quais modelos de sensores garantem estabilidade de medição em períodos longos, considerando os movimentos do corpo. **Mas já aprendemos que:** ● **CE3:** A pulseira elástica não mantém os sensores estáveis no braço do paciente. ● **CE5:** A pulseira rígida em formato de meia lua não mantém os sensores estáveis no braço do paciente, devido à falta de um fecho do lado aberto.

4.5 Evolução dos Instrumentos da ERC

A execução da prova de conceito mostrou a necessidade da inclusão de informações nos instrumentos FERC e QESS para que eles pudessem capturar as informações relevantes para o problema, e prover resultados úteis para a elicitação e a atualização dos requisitos conjecturais.

4.5.1 Evolução do Formato de Escrita dos Requisitos Conjecturais (FERC)

Duas informações foram acrescentadas ao FERC durante sua utilização. A primeira foi o identificador único das incertezas, com o objetivo de facilitar o seu

referenciamento no preenchimento do QESS. A segunda foi uma nova seção no campo de descrição do requisito conjectural, após a listagem das incertezas, para registro dos aprendizados previamente obtidos. Essa seção tem o objetivo de explicitar os aprendizados gerados a partir dos resultados dos ciclos experimentais relacionados com as incertezas de cada requisito conjectural, de modo a fornecer informações para apoiar a proposição de novas suposições de solução. Esses aprendizados possivelmente gerarão alterações nas incertezas e eventualmente nas outras partes da descrição. Eles devem ser mantidos na descrição do requisito conjectural para facilitar o entendimento de qualquer pessoa sobre os fatores que proporcionaram o requisito conjectural a evoluir até a descrição na qual se encontra no momento da leitura. Assim como as incertezas, cada aprendizado deve ser identificado por um identificador único, de modo a facilitar referências em outros artefatos do projeto.

Além disso, foram sugeridos dois novos campos (situação e prioridade) que não foram essenciais na prova de conceito, mas que poderiam apoiar determinados contextos de desenvolvimento. O campo “Situação” indica o estado atual do requisito conjectural, e deverá ser atualizado conforme a evolução do requisito, conforme a Figura 5. Quando um requisito conjectural é identificado, ele recebe a situação “Proposto”. O requisito conjectural proposto deve ser analisado para determinar se as suas incertezas serão alvo de experimentação para gerar maior aprendizado sobre elas. Caso se decida o desenvolvimento de uma solução sem abordagem de experimentação para o requisito conjectural, ele deve ser convertido em um requisito funcional, não funcional, estória de usuário ou outra forma de especificação de requisitos utilizada. Caso seja decidido que as incertezas precisam ser experimentadas, a situação do requisito será alterada para “Aprovado”.

A qualquer momento do processo de desenvolvimento, pode ser decidido que o requisito conjectural não é mais necessário. Nesse caso, a sua situação deve ser alterada para “Cancelado”. Quando todas as incertezas do requisito conjectural forem resolvidas ou aceitas, o requisito conjectural deixa de existir, sendo convertido em um requisito funcional, não funcional, estória de usuário ou outra forma de especificação de requisitos utilizada.

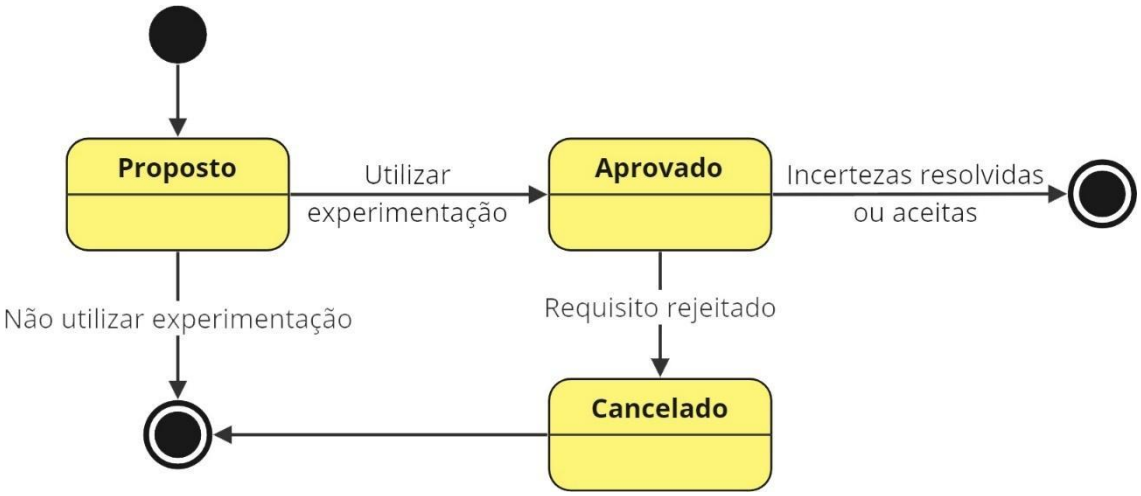

Figura 5 – Diagrama de estados do Requisito Conjectural.

Por fim, o campo "Prioridade" pode ser utilizado para facilitar a priorização de desenvolvimento e experimentação. Seus valores devem ser decididos de acordo com a prática adotada no projeto. Valores possíveis são, por exemplo, baixa, média ou alta, ou alguma escala de valores, como 1 a 5. A versão do FERC após essas evoluções pode ser vista na Quadro 22.

Quadro 22 – Evolução do Formato de Escrita dos Requisitos Conjecturais

<table>
<tr><th colspan="6">Requisitos Conjecturais</th></tr>
<tr><td>ID</td><td>RC[id]</td><td>Situação</td><td>[Proposto, Aprovado ou Cancelado]</td><td>Prioridade</td><td>[prioridade]</td></tr>
<tr><td>Descrição</td><td colspan="5">Espera-se que o sistema de software possua [comportamento desejado]
De modo que [necessidade ou impacto positivo do atributo desejado]
Porém, não sabemos:
● Incerteza In[id]a: [incerteza associada com este requisito]
● Incerteza In[id]b: [incerteza associada com este requisito]
● ...
Mas já aprendemos que:
● Ap[id]a: [aprendizado sobre as incertezas deste requisito conjectural]
● Ap[id]b: [aprendizado sobre as incertezas deste requisito conjectural]
● ...</td></tr>
<tr><td colspan="6">[O modelo se repete para cada requisito conjectural]</td></tr>
</table>

4.5.2 Evolução do Quadro de Experimentação para Suposições de Solução (QESS)

Nenhum campo foi acrescentado ao QESS. Porém, as referências às incertezas dos requisitos conjecturais foram evoluídas para conter o identificador único de cada incerteza (Quadro 23). O objetivo é evitar a repetição da descrição da incerteza e uma possível falha quando for necessária a atualização dos dados.

Quadro 23 – Evolução do Quadro de Experimentação para Suposições de Solução

Ciclo Experimental CE*[id-ce]*
Expectativa do Ciclo Experimental
Esperamos que *[descrição da suposição de solução]* **Resulte na atualização das incertezas sobre** *[incertezas que serão avaliadas]* **Como resultado de** *[descrição da observação e análise que resultará na atualização das incertezas]*
Requisitos Conjecturais do Ciclo Experimental
● RC*[id]*: Incerteza In*[id]* (versão *[n]*) ● RC*[id]*: Incerteza In*[id]* (versão *[n]*) ● …
Resultados do Ciclo Experimental
● **Incerteza In*[id][x]*:** *[descrição do aprendizado dos resultados da suposição de solução executada nesta iteração para essa incerteza]* ● **Incerteza In*[id][y]*:** *[descrição do aprendizado dos resultados da suposição de solução executada nesta iteração para essa incerteza]* ● …

4.6 Limitações da Prova de Conceito

Essa prova de conceito possui ameaças à validade relativas ao pesquisador, aos participantes do estudo e aos dados utilizados. Primeiramente, é considerável um possível viés tanto do pesquisador quanto dos participantes na avaliação dos resultados. Do pesquisador, pois foi o mesmo que definiu os instrumentos, e dos participantes, pois ambos participaram do projeto desde o seu início, possuindo amplo conhecimento sobre os seus requisitos e sobre os ciclos experimentais executados. Em segundo lugar, a forma de preenchimento dos instrumentos, realizada com a utilização de dados históricos, não acompanhando a evolução do projeto em tempo real, também traz a ameaça da perda de informações, o que pode possivelmente ter causado falhas no preenchimento e na análise da sua utilidade. Assim, os resultados gerados não podem ser generalizados, visto que a avaliação foi realizada em apenas um caso, com a percepção somente de duas pessoas, participantes originais do projeto.

4.7 Conclusão do Capítulo

Neste capítulo, foi apresentada a prova de conceito sobre a proposta da Especificação de Requisitos Conjecturais (ERC), com seus dois instrumentos: o Formato de Escrita dos Requisitos Conjecturais (FERC) e o Quadro de Experimentação para

Suposições de Solução (QESS). O estudo foi realizado pelo pesquisador a partir do histórico dos ciclos experimentais do projeto do sistema de software OxímetroIoT, a partir de entrevistas com dois participantes do projeto. A análise dos resultados gerou evoluções nos dois instrumentos, de modo a adequá-los para melhor atender aos objetivos. A prova de conceito sugeriu a viabilidade dos instrumentos da ERC para apoiar a elicitação e a atualização dos requisitos conjecturais, pois proveem meios de registro e rastreamento das informações. Entretanto, mais observações devem ser realizadas e estas serão objeto do próximo capítulo.

5 Estudo de Viabilidade de Aplicação do Formato de Escrita dos Requisitos Conjecturais

Neste capítulo é apresentado o estudo de viabilidade realizado sobre a utilização do instrumento FERC para o registro de requisitos conjecturais, executado no contexto do projeto daily Huddle, conduzido na Universidade Federal do Rio de Janeiro.

5.1 Introdução

Com o objetivo de observar a aplicação do formato de escrita dos requisitos conjecturais (FERC), parte da proposta de Gerenciamento dos Requisitos Conjecturais, foi conduzido um estudo de viabilidade. Para a sua realização, foi escolhido o projeto *daily Huddle*, que vem sendo desenvolvido por pesquisadores e estudantes envolvidos na construção de soluções para engenharia da saúde na Universidade Federal do Rio de Janeiro (UFRJ). O estudo de viabilidade é proposto por Shull, Carver e Travassos (Shull *et al.*, 2001) como uma fase inicial para a avaliação de novas tecnologias de software. Dois tipos de avaliação são sugeridos pelos autores para entender se os recursos gastos são compensatórios para a continuação da proposta: uma avaliação sobre a utilidade dos resultados e outra sobre se o tempo gasto foi bem utilizado. Neste estudo, a utilidade dos resultados foi avaliada através de uma análise sobre a suficiência do FERC, e o tempo gasto através de uma análise sobre a facilidade de uso do modelo.

O *daily Huddle* é um sistema de software, envolvendo dispositivos IoT, que visa otimizar a comunicação e a gestão de recursos que afetam a rotina hospitalar. A proposta do *daily Huddle* é criar um ambiente dinâmico para a gestão de informações sobre os recursos hospitalares (estruturas, equipamentos, materiais, pessoas e processos), funcionando como uma ferramenta para localizar, alocar e comunicar ocorrências, pendências ou restrições, atribuições de responsabilidade e prazos relacionados aos aspectos monitorados. No momento da execução deste estudo, o projeto encontrava-se em sua terceira iteração de desenvolvimento e seu segundo ciclo experimental, com o objetivo de desenvolver funcionalidades para ocorrências com equipamentos móveis, materiais esterilizados e estruturas.

5.2 Planejamento

Com o objetivo de observar a aplicação do FERC em um sistema de software em desenvolvimento e envolvendo desenvolvedores representados por graduandos em engenharia de computação e informação, a seguinte questão de pesquisa foi elaborada:

- ***O registro de requisitos conjecturais de acordo com o FERC é suficiente e possui facilidade de uso em um contexto de elicitação, análise e especificação de requisitos de software com experimentação contínua?***

Por **suficiência**, é esperado que o FERC proporcione aos participantes a oportunidade de registrar todas as informações identificadas sobre os requisitos conjecturais e que todas as informações previstas pelo modelo possam ser registradas. Por **facilidade de uso**, é esperado que o FERC exija algum esforço adicional, dado que é adicionada uma nova seção no documento de requisitos, mas que esse esforço para o seu preenchimento seja percebido como adequado para a tarefa.

O instrumento a ser utilizado com o FERC é o registro digital sobre um modelo escrito no software Microsoft Word, onde os participantes deverão registrar os requisitos conjecturais.

5.2.1 Objetivo

Quadro 24 – Objetivo do estudo, de acordo com o paradigma GQM (Basili *et al.*, 1994).

Objetivo do estudo	Avaliar o FERC para registro dos requisitos conjecturais.
Propósito	Caracterizar sua (1) suficiência, e (2) facilidade de uso.
Ponto de vista	Desenvolvedores de Software.
Contexto	Estudantes de graduação realizando a elicitação, análise e especificação de requisitos para o sistema de software *daily Huddle* como atividade prática de uma disciplina eletiva de período avançado em engenharia de computação e informação da UFRJ.

5.2.2 Seleção do contexto

O estudo encontra-se no contexto do registro das necessidades do sistema de software que serão objeto da terceira fase do projeto *daily Huddle*.

5.2.3 Seleção de variáveis

As seguintes variáveis foram selecionadas para a avaliação do estudo de viabilidade (Quadro 25):

Quadro 25 – Variáveis para a avaliação do FERC

Variáveis independentes	1. Projeto *daily Huddle*. 2. Cenários com a descrição das novas características desejadas para o sistema.
Variáveis dependentes	3. Grau de suficiência de informações do FERC para registro dos requisitos conjecturais e suas incertezas, medido através de uma escala VAS[4]; 4. Grau de facilidade de uso do FERC percebido pelos participantes, medido através de uma escala VAS.

5.2.4 Seleção de participantes

A seleção dos participantes foi realizada utilizando a técnica de amostragem não probabilística conhecida como amostragem por conveniência. Os participantes selecionados foram os alunos participantes em um curso de graduação com conhecimento de práticas de engenharia de software evoluindo a especificação de requisitos de um projeto de sistemas de software contemporâneo.

5.2.5 Desenho do estudo de viabilidade

Aleatoriedade. O objeto não é associado aleatoriamente aos participantes. Todos os participantes são alunos de graduação com conhecimento prévio sobre o projeto.

[4] A Escala Analógica Visual (*visual analog scale* - VAS) é um método para coletar classificações subjetivas quantificáveis em ambientes de pesquisa e clínicos. Consiste em uma escala de valores e foi inicialmente proposta para indicação do nível de dor sentido por pacientes em clínicas médicas (Aitken, 1969). Em uma escala VAS, o zero sempre significa a ausência do fenômeno, e o maior valor definido significa o seu nível máximo.

Agrupamento. Os participantes foram divididos em três grupos (Alpha, Beta e Gama). Cada grupo aplicou o instrumento na produção de um documento de requisitos para módulos específicos (gestão de Materiais, Equipamentos e Incêndio) do sistema de software. Cada grupo recebeu um cenário diferente para elicitação de requisitos. O grupo Alpha foi composto por cinco participantes, o grupo Beta igualmente por cinco participantes e o grupo Gama por seis participantes. A alocação dos participantes nos grupos, que também atuaram posteriormente no projeto e implementação das características especificadas, foi balanceada pelo perfil de desempenho dos estudantes com base no coeficiente de rendimento. Sendo assim, os grupos apresentaram distribuição de desempenho equivalente.

Tipo de estudo. O estudo de viabilidade é de um fator e um tratamento. O fator é o formato de especificação e o tratamento é o FERC.

5.2.6 Instrumentação

5.2.6.1 Objetos

Os objetos utilizados nesse estudo são os seguintes:

- Projeto *daily Huddle* e seus artefatos previamente construídos.
- Cenários descrevendo as novas características desejadas.
- Vídeo de treinamento explicando sobre a estrutura dos requisitos conjecturais e demonstrando a maneira como devem ser preenchidos com o FERC.
- Documento digital com o modelo do FERC.
- Questionário para aplicação individual (ver seção 5.2.6.3).

5.2.6.2 Diretrizes

Quadro 26 – Diretrizes para condução do estudo de viabilidade sobre o FERC.

Etapa	Formato
1. Exibição do vídeo de treinamento (30 minutos) sobre requisitos conjecturais e o FERC para os participantes.	Vídeo exibido em projetor
2. Distribuição dos novos cenários do projeto *daily Huddle* (diferentes para cada grupo), com abertura para dúvidas.	Documento impresso/digital
3. Distribuição do modelo de documento de requisitos digital contendo o modelo do FERC.	Documento digital
4. Execução do registro dos requisitos (incluindo os requisitos conjecturais no modelo do FERC) pelos grupos.	Documento digital
5. Aplicação do questionário para todos os participantes.	Questionário individual
6. Análise dos resultados dos questionários.	Executado posteriormente pelo pesquisador

5.2.6.3 Coleta de Dados

Foram coletados dados específicos ao final da execução do estudo de viabilidade, de modo a identificar qualitativamente a suficiência e facilidade de uso do FERC, através do seguinte questionário:

1. De qual grupo você participou?
    - Grupo Alpha
    - Grupo Beta
    - Grupo Gama
2. Qual é a sua experiência com desenvolvimento de sistemas? (escolha única)
    - Nenhuma
    - Apenas no contexto universitário
    - Desenvolvi sistemas não comerciais fora do contexto universitário
    - Desenvolvi sistemas comerciais por conta própria
    - Desenvolvi sistemas comerciais como parte de uma empresa
3. Qual é a sua experiência com identificação de requisitos? (múltipla escolha)

- Nenhuma
- Experiência no contexto universitário
- Experiência com criação de lista de requisitos funcionais e não funcionais
- Experiência com descrição de casos de uso
- Experiência com diagramas UML para especificação de requisitos
- Experiência com histórias de usuários e/ou cenários (metodologias ágeis)

4. Em uma escala de 0 a 10, como você considera que foi possível registrar todas as informações sobre os requisitos e as incertezas identificadas utilizando o Formato de Escrita de Requisitos Conjecturais? (sendo 0 = não foi possível registrar os requisitos e incertezas; e 10 = foi possível registrar todos os requisitos e incertezas)
    - Escala de 0 a 10 (em VAS)
5. Quais informações você identificou, porém não foi possível registrar?
    - Texto livre
6. Em uma escala de 0 a 10, como você considera que foi possível identificar todas as informações previstas pelo modelo do Formato de Escrita de Requisitos Conjecturais? (sendo 0 = não foi possível registrar nenhuma informação prevista; e 10 = foi possível registrar todas as informações previstas)
    - Escala de 0 a 10 (em VAS)
7. Quais informações previstas no modelo não foram possíveis de serem identificadas?
    - Texto livre
8. Em uma escala de 0 a 10, como você considera a sobrecarga de esforço exigida pelo Formato de Escrita de Requisitos Conjecturais, em comparação com o esforço para registro dos requisitos funcionais e não funcionais? (sendo 0 = não houve esforço adicional; e 10 = houve esforço adicional excessivo)
    - Escala de 0 a 10 (em VAS)

9. Qual parte do modelo exigiu maior esforço de preenchimento? Por quê?
    - Texto livre

5.3 Execução

Os três grupos receberam a documentação das iterações anteriores do projeto *daily Huddle* e os novos cenários a serem desenvolvidos para entendimento do contexto. Depois disso, houve a exibição do vídeo de treinamento sobre requisitos conjecturais e o FERC para os participantes. As dúvidas foram respondidas pelo professor da disciplina, e não pelo proponente da técnica. Após o treinamento, os grupos tiveram duas semanas para preparar o documento de requisitos contendo os requisitos conjecturais com a utilização do FERC.

Ao término da especificação dos requisitos, os grupos realizaram o desenvolvimento, experimentação e a atualização dos requisitos conjecturais. Por fim, todos responderam ao questionário sobre a utilização do FERC.

5.4 Análise dos Resultados

5.4.1 Análise do preenchimento do FERC

O estudo de viabilidade foi executado durante o mês de julho de 2023. Os três grupos criaram requisitos conjecturais no modelo do FERC. O grupo Alpha criou três requisitos conjecturais, o grupo Beta dois e o grupo Gama um. O grupo Alpha preencheu o modelo corretamente (Quadro 27), descrevendo os requisitos conjecturais e registrando incertezas e aprendizados com os devidos identificadores únicos. Porém, o RC01 do grupo Alpha é semelhante a dois outros requisitos listados pelo mesmo grupo, um funcional e outro não funcional (Quadro 28). O grupo utilizou essa prática para, ao mesmo tempo, registrar o requisito e indicar as suas incertezas nas diferentes seções do documento de requisitos.

Além disso, percebe-se uma dificuldade de diferenciação entre incertezas e suposições de solução no registro das incertezas In001c e In001e, pois a utilização da tecnologia RFID não é um requisito do software, devendo ser entendida como uma

suposição de solução que pode potencialmente atender ao requisito de localização dos equipamentos. Por fim, há um conflito no RC03 do grupo Alpha entre a incerteza In003b e o aprendizado ap003b, pois se o aprendizado diz que e-mails não são eficientes, o seu uso não deveria mais ser uma incerteza, podendo ser removida do requisito.

Quadro 27 – Requisitos conjecturais registrados pelo grupo Alpha utilizando o FERC.

ID	RC01	**Situação**	Proposto	**Prioridade**	Média
Descrição	**Espera-se que o sistema de software possua** localização dos equipamentos em tempo real; **De modo que** se tenha a informação de onde determinado equipamento está e/ou se foi abandonado. **Porém, não sabemos:** ● **Incerteza In001a:** se com a tecnologia de sensores haverá precisão nos dados, principalmente em relação a variação de andares. ● **Incerteza In001b:** se é possível inserir os sensores nas rotinas dos usuários no hospital de forma eficiente e sem burocracias. ● **Incerteza In001c:** quais equipamentos são ideais para troca de sinais, visto as interferências que ocorrem com os equipamentos médico e RFID. ● **Incerteza In001d:** se sistemas de triangulação são sequer possíveis de serem instalados no hospital dado as limitações de interferência e custo. ● **Incerteza In001e:** se o custo de colocar RFID em todos os equipamentos de integrantes da equipe hospitalar é grande demais para inviabilizar o projeto. **Já aprendemos que:** ● **Ap001a:** em ambientes com muitos metais os sensores não têm um bom funcionamento, e em hospitais as salas geralmente são metálicas. ● **Ap001b:** sabemos que há elementos obrigatórios de identificação dos funcionários, como o crachá por exemplo, que podem se relacionar com os sensores.				

ID	RC02	**Situação**	Proposto	**Prioridade**	Alta
Descrição	**Espera-se que o sistema de software possua** *check-in/check-out* de equipamentos que atualiza o status do equipamento em tempo real; **De modo que** se tenha a informação de quem fez o *check-in* e *check-out* de um dado equipamento em tempo real. **Porém, não sabemos:** ● **Incerteza In002a:** dado que mais de uma pessoa pode estar na sala de equipamentos, não sabemos como evitar conflitos para que o elo de RFID entre um usuário e o equipamento ocorra corretamente (assumindo que o usuário é identificado por RFID). ● **Incerteza In002b:** se existe uma base inteligente que trave o equipamento na sua localização de *stand-by*. ● **Incerteza In002c:** se o custo de colocar *tags* RFID em todos os crachás de integrantes da equipe hospitalar é grande demais de forma que inviabilize essa parte do projeto. ● **Incerteza In002d:** se faz sentido atrelar a retirada de um equipamento a leitura de um *QRCode* único por usuário - colocado no crachá do integrante da equipe (assumindo que o usuário é identificado por *QRCode*). **Já aprendemos que:** ● -				
ID	RC03	**Situação**	Proposto	**Prioridade**	Alta
Descrição	**Espera-se que o sistema de software possua** notificações/alarmes para os interessados nos eventos ocorrendo no **Huddle GOEM**; **De modo que** usuários interessados em certos eventos sejam notificados rapidamente. **Porém, não sabemos:** ● **Incerteza In003a:** se apenas a notificação na interface será suficiente para o usuário ser avisado de prontidão. ● **Incerteza In003b:** se o e-mail é a melhor forma de comunicar algo urgente para o usuário. ● **Incerteza In003c:** se enviar *push notifications* é viável, visto a necessidade do desenvolvimento de um aplicativo e integração com o huddle para o envio das *pushes*. ● **Incerteza In003d:** se faria sentido soar alarmes específicos no hospital, em algum equipamento, sinalizando a existência de um dado problema. **Já aprendemos que:** ● **Ap003a:** Alarmes sonoros não são a melhor opção por se tratar de um ambiente hospitalar. ● **Ap003b:** *E-mails* normalmente são problemáticos visto a baixa confidencialidade de visualização em tempo real do alarme do usuário.				

Quadro 28 – Requisitos funcionais e não funcionais registrados pelo grupo Alpha relacionados com o RC01 do mesmo grupo.

ID	RF04	**Característica IoT**	Identificação		
Descrição	Mostrar localização do equipamento na Listagem de Equipamentos.				
ID	RNF004			**Situação**	Proposto
Descrição	O dispositivo deverá estar a todo momento conectado à internet e disponível.			**Prioridade**	Alta

O grupo Beta descreveu corretamente os requisitos conjecturais e registrou incertezas e aprendizados (Quadro 29). Porém, o grupo não utilizou identificadores únicos para incertezas nem para aprendizados, o que dificulta o seu referenciamento. Além disso, o aprendizado registrado no RC01 desse grupo apresenta uma dúvida que não deveria estar nesse aprendizado, devendo constar apenas na lista de incertezas do requisito conjectural.

Quadro 29 – Requisitos conjecturais registrados pelo grupo Beta utilizando o FERC.

ID	RC01	**Situação**	Proposto	**Prioridade**	Alta
Descrição	**Espera-se que o sistema possua** medidas confiáveis; **De modo que** tenhamos a quantidade de medidas certas sem que haja repetição desnecessária das medições. **Porém, não sabemos:** ● **Incerteza:** De quanto em quanto tempo devemos coletar as medidas. **Já aprendemos que:** ● **Aprendemos:** Para a medida da temperatura, a variação de temperatura não deve mudar muito a cada segundo, podendo ser calculada a cada minuto. Porém de quanto em quantos minutos?				
ID	RC02	**Situação**	Proposto	**Prioridade**	Média
Descrição	**Espera-se que o sistema possua** um meio de alertar alguém de forma rápida sobre risco de incêndio; **De modo que** tenhamos uma intervenção humana o mais rápido possível. **Porém, não sabemos:** ● **Incerteza:** a quem enviar um alerta. ● **Incerteza:** qual forma envio seria mais efetiva. **Já aprendemos que:** ● **Aprendemos:** Apenas sinalizar com um alerta no *dashboard* pode não ser suficiente para ter uma intervenção rápida e acabar com o risco de incêndio.				

Finalmente, o grupo Gama descreveu o requisito conjectural e registrou uma incerteza e um aprendizado com os devidos identificadores únicos (Quadro 30). Porém, a incerteza registrada não está clara, tornando difícil o entendimento sobre qual é de fato a dúvida relativa à robustez do equipamento e à capacidade de processamento. A incerteza deveria ser escrita mais detalhadamente, de modo a facilitar o entendimento de outras pessoas que não participaram da elicitação do requisito.

Quadro 30 – Requisito conjectural registrado pelo grupo Gama utilizando o FERC.

ID	RC01	**Situação**	Proposto	**Prioridade**	Alta
Descrição	**Espera-se que o sistema de software possua** Módulos IoT capazes de processar os dados e verificar a conformidade com os limites pré-definidos; **De modo que** possamos determinar (mesmo sem uma conexão com o *Broker*) o estado atual de cada módulo. **Porém, não sabemos:** ● **Incerteza In01a:** Robustez do equipamento e capacidade de processamento. **Já aprendemos que:** ● **Ap01a:** Quando o Módulo IoT é independente, a verificação de Temperatura e Umidade pode ser feita por meio de inspeção visual nos LEDs presentes no caso da comunicação do Módulo IoT com o Broker (consequentemente com o restante do sistema) estar comprometida.				

A partir da observação do preenchimento do FERC pelos grupos, é possível observar que, no contexto do estudo, as informações solicitadas pelo modelo puderam ser obtidas e registradas. Além disso, os dados informados no instrumento fazem sentido e correspondem a incertezas adequadas aos cenários fornecidos. Entretanto, alguns conceitos aparentemente não foram bem assimilados pelos participantes, e precisariam ser reforçados para que o instrumento pudesse ser mais bem utilizado. Primeiramente, é importante enfatizar a importância da definição dos identificadores únicos tanto das incertezas quanto dos aprendizados.

Em segundo lugar, a diferença entre os conceitos de incerteza e suposição de solução poderia ser mais bem definida, para que as incertezas sejam bem escritas e para evitar a colocação indevida de incertezas na seção de aprendizados. Por fim, também seria importante deixar mais clara a natureza atualizável dos requisitos conjecturais, evitando que incertezas já resolvidas sejam mantidas na sua descrição.

5.4.2 Análise dos resultados do questionário

5.4.2.1 Período e quantidade de respostas

Foram recebidas 22 respostas ao questionário, sendo 13 completas e 9 incompletas no período de 10 de julho de 2023 a 17 de julho de 2023. Porém, o estudo contava apenas com 16 participantes, o que permite o entendimento de que alguns participantes iniciaram o questionário, não concluíram e posteriormente retornaram e preencheram o questionário completamente. Para evitar informações repetidas, foram removidas todas as respostas dos participantes que não completaram o questionário. Portanto, apenas as respostas de 13 participantes foram consideradas na análise. Desses, quatro pertencem ao grupo Alpha, cinco ao grupo Beta e quatro ao grupo Gama (Figura 6).

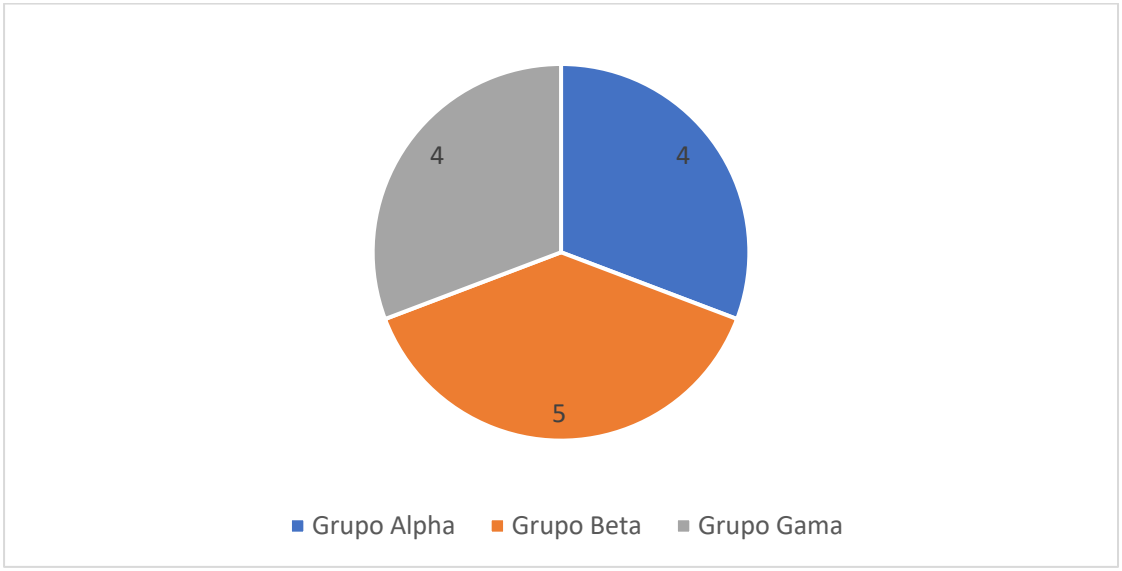

Figura 6 – Gráfico comparativo da quantidade de respondentes do questionário por grupo.

5.4.2.2 Caracterização dos participantes

A experiência dos participantes com desenvolvimento de sistemas (Figura 7) mostrou-se equilibrada entre desenvolvimento no contexto universitário e em contexto comercial para os grupos Alpha e Gama. Já no grupo Beta, a experiência com desenvolvimento de sistemas comerciais predomina entre os integrantes. Apenas um participante não possuía experiência com desenvolvimento, no grupo Alpha, o que foi mitigado por pertencer a um grupo onde os demais integrantes possuem experiência.

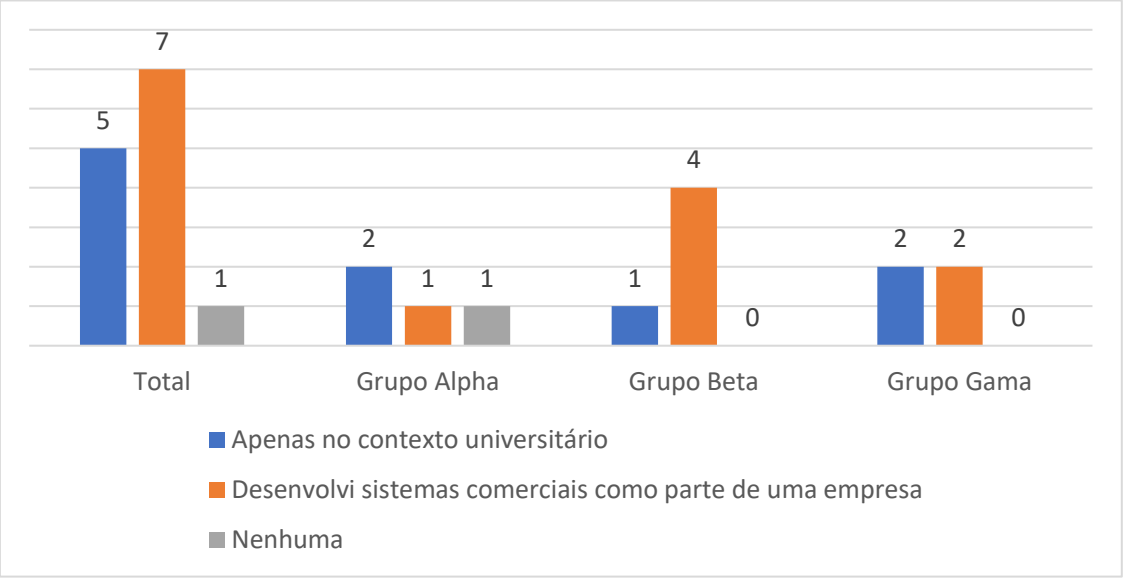

Figura 7 – Gráfico comparativo da experiência dos participantes com desenvolvimento de sistemas.

A experiência com identificação de requisitos (Figura 8), por outro lado, apresentou-se maior no contexto universitário, principalmente no formato de escrita de cenários e histórias de usuários utilizando metodologias ágeis. Nessa pergunta, os participantes poderiam escolher uma ou mais opções de resposta. Apenas um participante não possuía experiência com identificação de requisitos, o que foi mitigado por pertencer a um grupo onde os outros integrantes possuem experiência. Poucos participantes (15%) possuíam experiência com descrição de listas de requisitos funcionais e não funcionais. Como o estudo utilizou esse modelo de organização, isso pode ter influenciado na dificuldade apresentada para registro dos requisitos conjecturais no formato solicitado. Apenas um participante possuía experiência com descrição de casos de uso, e nenhum relatou experiência com a utilização de diagramas UML.

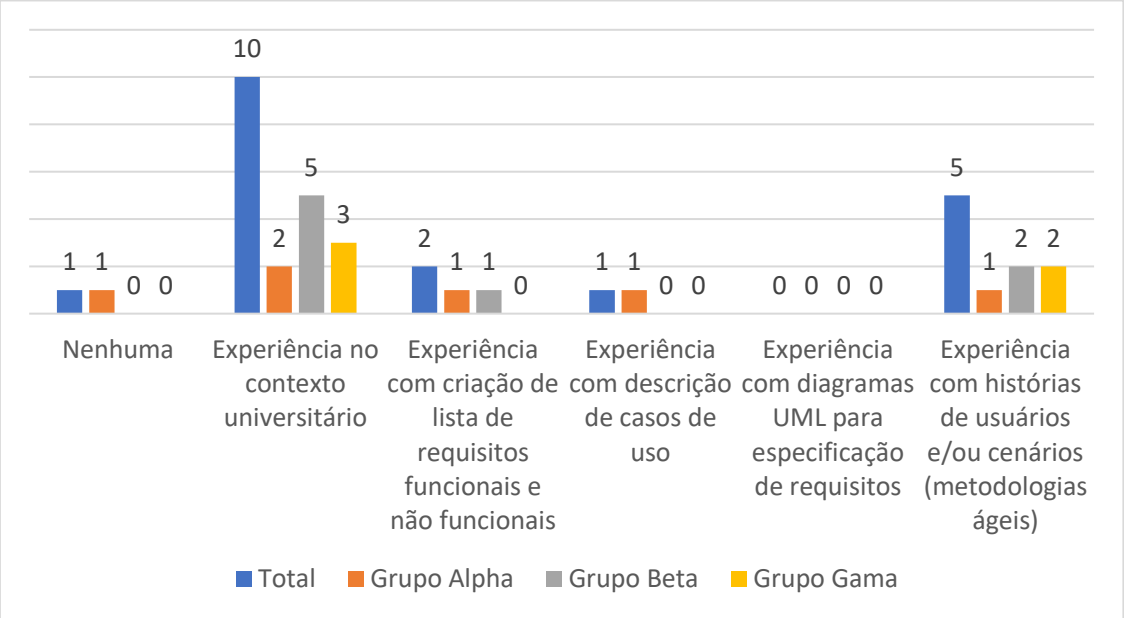

Figura 8 – Gráfico comparativo sobre a experiência com identificação de requisitos

5.4.2.3 Suficiência do FERC

Duas perguntas específicas sobre a suficiência do FERC foram analisadas para compreender a percepção dos participantes sobre os campos do modelo proposto para o instrumento, cada uma com uma parte quantitativa obrigatória e outra de texto livre opcional. Na primeira pergunta, foi solicitado que os participantes utilizassem uma escala de 0 a 10 (VAS) para representar a experiência de utilização do FERC para registrar todas as informações sobre os requisitos e as incertezas identificadas. Nessa escala, o valor zero deveria ser escolhido se não tiver sido possível registrar os requisitos e incertezas e o valor dez se foi possível registrar todos os requisitos e incertezas que os participantes entenderam existir. Para possibilitar a análise das respostas dessa pergunta, elas foram agrupadas em três faixas assim classificadas: detratores (valores 0 a 4), neutros (valores 5 a 7) e promotores (valores 8 a 10), conforme mostrado na Figura 9.

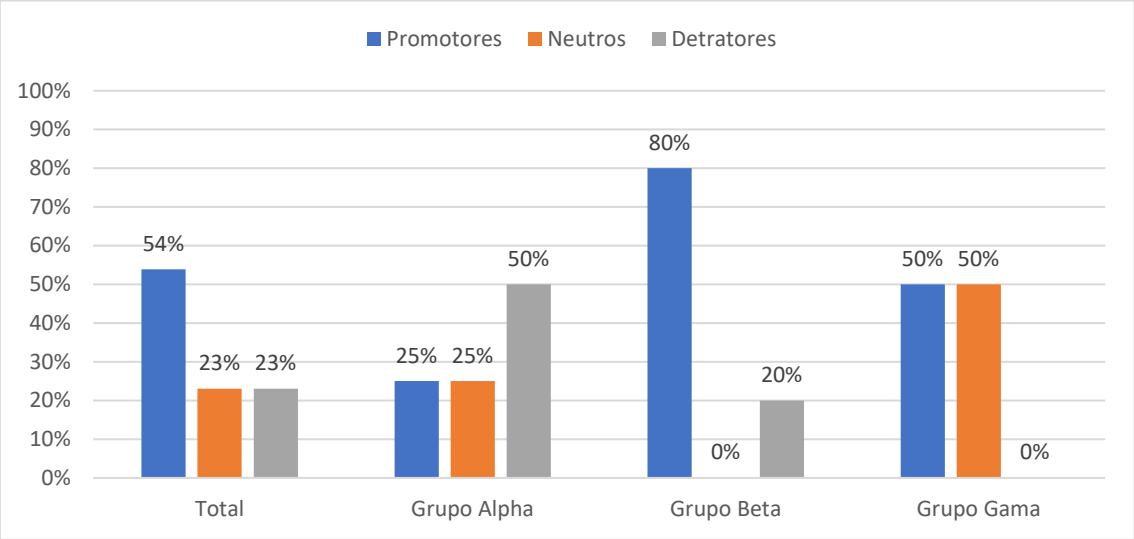

Figura 9 – Respostas classificadas por faixas de valores para a pergunta sobre a possibilidade de registro de todas as informações sobre os requisitos e as incertezas identificadas utilizando o FERC.

Apesar de metade do grupo Alpha ter respondido como detratores, o grupo preencheu corretamente o FERC, incluindo diversas incertezas. Para entender o motivo das respostas, seria preciso realizar uma entrevista com o grupo. A pouca experiência dos participantes com desenvolvimento e identificação de requisitos, entretanto, pode ser um indicador importante para essa compreensão. Já no grupo Beta, a grande maioria respondeu como promotores, o que corresponde ao seu preenchimento do FERC. A experiência da maior parte do grupo com o desenvolvimento de sistemas comerciais pode ter contribuído para essa percepção. O grupo Gama não teve nenhum detrator, e metade dos participantes votou como promotor. No total, 54% dos participantes foram promotores da possibilidade de registro das informações, contra apenas 23% de detratores.

Apenas os grupos Beta e Gama enviaram alguma resposta de texto livre sobre as informações que não foram possíveis de serem registradas (Quadro 31). Enquanto o grupo Beta justificou suas notas promotoras, o comentário do grupo Gama sobre as incertezas que não foram registradas precisaria ser entendido por meio de entrevistas para determinar o motivo que impediu o registro. As entrevistas não puderam ser realizadas por dificuldades logísticas e de tempo.

Quadro 31 – Respostas de texto livre à pergunta "Quais informações você identificou, porém não foi possível registrar?"

Grupo Beta	Todas as incertezas identificadas utilizaram o formato de escrita de requisitos conjecturais. Ajudou bastante. Mas acredito que existem mais incertezas que não foram registradas.
Grupo Beta	Não identifiquei nenhuma que não foi registrada. todas as identificadas por nós foram registradas.
Grupo Beta	Foi possível registrar todos os requisitos e incertezas.
Grupo Gama	Houve várias incertezas quanto à correta disposição (alocação) das bandejas nas prateleiras a cada interação com o sistema que não foram registradas no FERC.

Na segunda pergunta sobre a suficiência, foi solicitado que os participantes utilizassem uma escala de 0 a 10 (VAS) para representar a possibilidade de identificação de todas as informações solicitadas pelo FERC. As respostas também foram agrupadas nas mesmas faixas de valores da pergunta anterior: detratores (valores 0 a 4), neutros (valores 5 a 7) e promotores (valores 8 a 10), conforme mostrado na Figura 10.

Apesar de metade do grupo Alpha ter respondido como detrator, o grupo preencheu todas as informações solicitadas pelo FERC. Para entender o motivo das respostas, seria preciso realizar uma entrevista com o grupo. Porém, novamente a pouca experiência dos participantes com desenvolvimento e identificação de requisitos pode ser um indicador importante para essa compreensão. Já nos grupos Beta e Gama, a grande maioria respondeu como promotor, o que corresponde ao seu preenchimento do FERC e à sua experiência com desenvolvimento e identificação de requisitos. No total, 62% dos participantes consideraram possível a obtenção de todas as informações solicitadas pelo FERC.

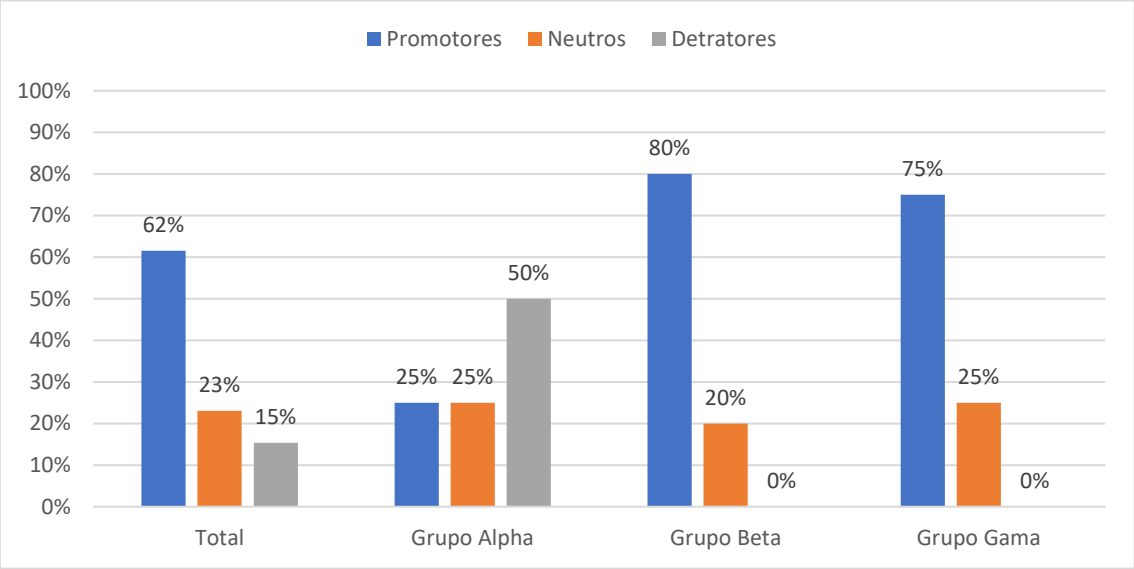

Figura 10 – Respostas classificadas por faixas de valores para a pergunta sobre a possibilidade de preenchimento das informações previstas pelo FERC.

Nas respostas de texto livre sobre as informações que não puderam ser identificadas (Quadro 32), o comentário sobre o preenchimento da seção de aprendizados era esperado, pois essa seção é preenchida ao longo dos ciclos experimentais, que podem não ter ocorrido. Além disso, pelo comentário do grupo Gama, é possível que os objetivos dos enunciados da questão anterior e desta não estivessem claros o suficiente, o que pode ter gerado o preenchimento errado por parte de alguns participantes. Para facilitar o entendimento, os enunciados poderiam ser alterados para, por exemplo, “Em uma escala de 0 a 10, como você considera que foi possível preencher todos os campos solicitados pelo modelo do Formato de Escrita de Requisitos Conjecturais?” e “Quais campos solicitados pelo modelo não foram possíveis de serem identificados?”.

Quadro 32 – Respostas de texto livre à pergunta “Quais informações previstas no modelo não foram possíveis de serem identificadas?”

Grupo Beta	Acredito que a parte do aprendizado foi mais complicado de preencher por às vezes não ter tido de fato um aprendizado.
Grupo Beta	Todas as situações a serem completadas foram identificadas.
Grupo Beta	Foi possível identificar todas as informações previstas pelo modelo do Formato de Escrita de Requisitos Conjecturais
Grupo Gama	Restrições legais, regras de negócio e escopo do projeto.

A partir dessa análise, e dentro do escopo do estudo, é possível concluir que há indícios de que o FERC é percebido como suficiente para registro das informações sobre as incertezas das demandas, sendo possível obter todas as informações solicitadas. Entretanto, mais estudos são necessários para o entendimento das dificuldades de preenchimento informadas sobre tipos específicos de incertezas e nas respostas classificadas como detratoras.

5.4.2.4 Facilidade de uso do FERC

Para compreender a percepção dos participantes sobre a facilidade de uso do FERC, foi analisada uma pergunta com uma parte obrigatória e outra de texto livre opcional. Essa pergunta foi sobre a sobrecarga de esforço para preenchimento do FERC, em comparação com o esforço para registro dos requisitos funcionais e não funcionais. Os participantes deveriam indicar um valor entre zero e dez, sendo zero se não houve esforço adicional e dez se houve esforço adicional excessivo. As respostas a essa pergunta também foram agrupadas em faixas de valores. Porém, a classificação foi diferente das perguntas anteriores, pois nesse caso foi preciso avaliar o nível de esforço, e não a satisfação com o modelo. Portanto, a classificação foi definida da seguinte maneira: pouco esforço (valores 0 a 3), esforço considerável (valores 4 a 6) e muito esforço (valores 7 a 10), conforme mostrado na Figura 11.

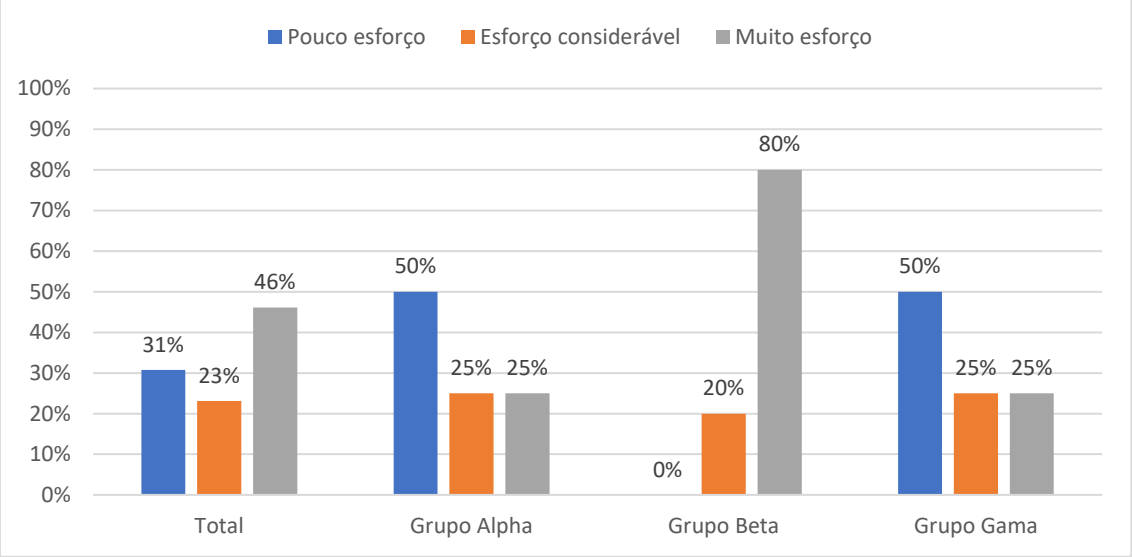

Figura 11 – Respostas classificadas por faixas de valores sobre o esforço adicional para preenchimento do FERC.

Ao contrário dos grupos Alpha e Gama, o grupo Beta considerou que houve esforço adicional excessivo. Esse resultado era esperado, pois o FERC exige que o analista pense em incertezas que talvez não estivessem claras, o que exige um esforço maior. Entretanto, a diferença entre os objetivos funcionais dos módulos trabalhados por cada grupo pode ter contribuído para a diferença entre eles. No total, 50% consideraram que há muito esforço para o registro dos requisitos conjecturais.

Nas respostas de texto livre sobre as partes do modelo que exigiram maior esforço de preenchimento (Quadro 33), os comentários confirmam que há um esforço adicional para preenchimento do FERC devido à necessidade de pensar mais sobre o que ainda não se sabe em relação aos requisitos. Porém, os participantes confirmam que esse esforço é compensatório. Talvez esse esforço possa ser reduzido inserindo uma fase de ideação do projeto antes do registro dos requisitos, com o objetivo de elicitar as incertezas, o que não ocorreu no projeto observado.

Quadro 33 – Respostas de texto livre à pergunta 9 "Qual parte do modelo exigiu maior esforço de preenchimento? Por quê?"

Grupo Alpha	Há um esforço adicional sim - como os requisitos conjecturais são um pouco mais abstratos e conhecemos pouco sobre eles, o domínio sobre eles também é menor do que os outros, fazendo com que a quantidade de esforço mental para pensar no que não sabemos e descrever bem os riscos seja um pouco maior.
Grupo Beta	Os requisitos conjecturais com certeza foram mais desafiadores que os requisitos funcionais e não funcionais. Não apenas pelo fato de ser algo mais novo, mas também pelo fato de que demanda um pensamento maior em relação ao problema para identificar esses requisitos, ou seja, pensar um pouco mais fora da caixa.
Grupo Beta	Esse modelo exige muito mais esforço que os outros documentos, pois devemos pensar em mais situações, mas é um trabalho que compensa e pode ser retrabalhado ao longo do tempo. Vale dizer que o modelo é um pouco confuso na diagramação. Porém a parte que mais gera trabalho é a descrição, pois devemos pensar nas incertezas e tudo mais.
Grupo Beta	A descrição do problema foi uma das partes que mais exigiram esforço, pois precisou de muitos esclarecimentos por parte dos *stakeholders* e contínua reafirmação para entender qual era a dor do cliente.

Grupo Gama	Requisitos Conjecturais. Eles exigem um conhecimento tácito e menos sistemático para o seu correto preenchimento, o que não ocorre no restante do artefato. Neste, são necessárias apenas uma compreensão das necessidades por parte dos interessados no software, diferente daquele.

A partir dessa análise, é possível concluir que há indícios de que o esforço para registro dos requisitos conjecturais com o FERC muitas vezes é maior do que o esforço para registro dos demais requisitos, pois ele exige um trabalho intelectual de identificação de incertezas de características do software que pode não ter sido realizado antes. Porém, esse esforço adicional pode ser compensatório, pois possibilita o registro de informações que, sem esse instrumento, não seriam registradas.

5.5 Evolução da ERC

5.5.1 Evolução do FERC

Um dos aprendizados do estudo de viabilidade foi que os requisitos conjecturais eventualmente estão associados com outros requisitos (funcionais e/ou não funcionais). Quando essa associação existe, é importante que ela seja explicitada, para fornecer maior contexto ao registro do requisito conjectural, de modo a facilitar o entendimento principalmente de atores externos ao projeto. Porém, a versão anterior do FERC não prevê referências, conforme análise na subseção 5.4.1. O acréscimo de uma seção de requisitos referenciados no FERC facilitaria a rastreabilidade dos requisitos conjecturais, relacionando-os com os requisitos funcionais e não-funcionais, conforme mostrado na Quadro 34.

Quadro 34 – Evolução do FERC a partir dos resultados do estudo de viabilidade.

<table>
<tr><th colspan="6">Requisitos Conjecturais</th></tr>
<tr><td>ID</td><td>RC[id]</td><td>Situação</td><td>[Proposto, Aprovado, ou Cancelado]</td><td>Prioridade</td><td>[Prioridade]</td></tr>
<tr><td>Requisitos Associados</td><td colspan="5">[Referências para outros requisitos associados]</td></tr>
<tr><td>Descrição</td><td colspan="5">Espera-se que o sistema de software possua [atributo desejado]
De modo que [necessidade do atributo desejado]
Porém, não sabemos:
● Incerteza In[id]a: [incerteza associada com este requisito]
● Incerteza In[id]b: [incerteza associada com este requisito]
● ...
Já aprendemos que:
● Ap[id]a: [aprendizado sobre as incertezas deste requisito conjectural]
● Ap[id]b: [aprendizado sobre as incertezas deste requisito conjectural]
● ...</td></tr>
<tr><td colspan="6">[O modelo se repete para cada requisito conjectural]</td></tr>
</table>

5.5.2 Criação do Ciclo de Vida do Requisito Conjectural

Os resultados do estudo de viabilidade também mostraram que existe uma falta de clareza entre os conceitos de “requisitos conjecturais” e “suposições de solução”, que resulta na dificuldade de registrar adequadamente as suposições de solução no FERC e na atualização das incertezas, conforme análise na subseção 5.4.1. Para deixar mais clara a diferença entre os conceitos e as possibilidades de evolução do requisito conjectural, de modo a facilitar o entendimento dos conceitos e o preenchimento e atualização do FERC, foi definido o ciclo de vida do requisito conjectural, conforme a Figura 12.

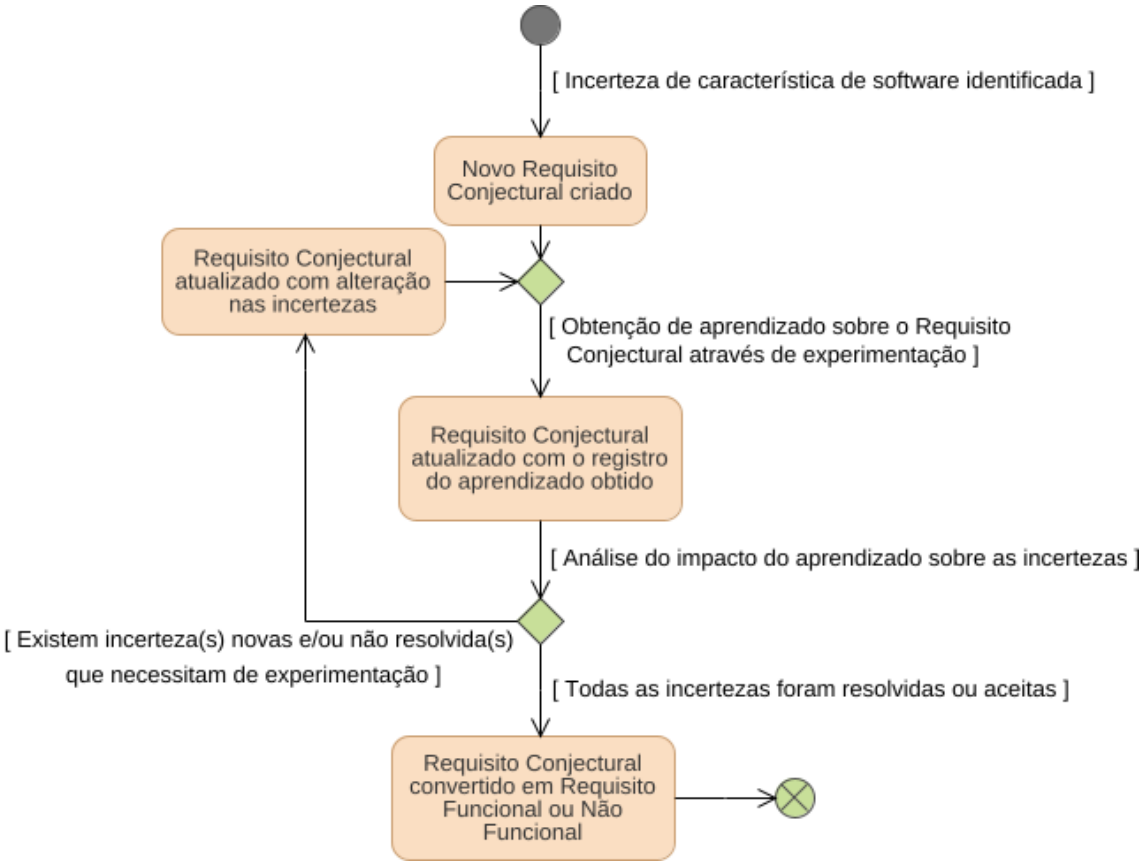

Figura 12 – Ciclo de vida do requisito conjectural.

Um novo requisito conjectural é criado a partir da identificação de uma incerteza sobre uma característica do software. Depois de registrado no FERC, ele aguarda a priorização para um ciclo experimental. Após a priorização, execução da experimentação e obtenção de algum aprendizado sobre esse requisito conjectural, o aprendizado obtido é registrado juntamente com o requisito conjectural.

Após esse registro, o impacto desse aprendizado sobre as incertezas do requisito conjectural é avaliado. Caso as incertezas do requisito conjectural que foram experimentadas não tenham sido resolvidas, e/ou tenham sido identificadas novas incertezas e seja decidido que é necessário realizar uma nova experimentação, as incertezas do requisito conjectural serão atualizadas e ele aguarda priorização para um novo ciclo experimental. Porém, caso as incertezas experimentadas tenham sido resolvidas ou caso seja decidido que não é necessário realizar mais experimentação, as demais incertezas do requisito conjectural são avaliadas. Caso ainda existam incertezas, o requisito conjectural é atualizado com a remoção das incertezas resolvidas e ele aguarda priorização para um novo ciclo experimental.

Caso todas as incertezas do requisito conjectural tenham sido resolvidas ou caso seja decidido que não é mais necessário realizar experimentações sobre esse requisito conjectural, ele torna-se elegível para ser convertido em um requisito funcional, não-funcional ou em uma história de usuário. A qualquer momento do desenvolvimento, entretanto, o requisito conjectural poderá ser descartado por alguma decisão de negócio.

5.5.3 Evolução do Treinamento de Uso

A análise também mostrou que detalhes do FERC, como os identificadores das incertezas e dos aprendizados, não foram preenchidos corretamente, e que a localização adequada das informações não foi respeitada em alguns casos, com dúvidas sendo registradas na seção de aprendizados, por exemplo, conforme mostrado na subseção 5.4.1. A alteração do treinamento de uso do FERC poderia reforçar a importância do correto preenchimento desses campos.

Além disso, com o aumento da sua duração, o treinamento poderia mostrar e explicar o ciclo de vida do requisito conjectural de modo a deixar mais clara a diferença entre os conceitos e mostrando exemplos melhores de registro de incertezas e suposições de solução e da evolução das informações de acordo com os resultados dos ciclos experimentais. A versão final do vídeo de treinamento encontra-se no Apêndice A.

5.6 Limitações do Estudo

As ameaças à validade deste estudo são de dois tipos. Primeiramente, as ameaças internas à validade devem-se a um possível viés de observação do pesquisador sobre a atividade de registro dos requisitos conjecturais e à abordagem qualitativa do questionário para medição dos resultados. Essas ameaças são aceitas e consideradas na análise dos dados.

Uma ameaça externa à validade do estudo é a experiência dos participantes com elicitação de requisitos. Possivelmente, participantes com maior experiência podem ter maior facilidade na elicitação e influenciar o grupo, o que pode mascarar o possível efeito positivo da aplicação dos instrumentos. Para mitigar essa ameaça, foi incluído no questionário a experiência prévia de cada participante com elicitação de requisitos, de modo a identificar os mais experientes na análise dos dados. Assim, percebe-se que a

aceitação do FERC foi maior nos grupos que possuíam maior quantidade de participantes com experiência de desenvolvimento de software na indústria.

O contexto do estudo é limitado, pois foi realizado em sala de aula com alunos de graduação. Assim, seus resultados não devem ser generalizados.

5.7 Conclusão do Capítulo

Este capítulo apresentou o estudo de viabilidade sobre o Formato de Escrita dos Requisitos Conjecturais (FERC), instrumento principal da proposta de Especificação dos Requisitos Conjecturais. A avaliação foi realizada através da observação da utilização do FERC por três grupos de participantes em um contexto acadêmico, porém atuando em um projeto de sistema de software contemporâneo real. Os participantes também responderam um questionário individualmente, e as suas respostas contribuíram para a análise.

O resultado da análise indica que o FERC foi percebido como suficiente para registro dos requisitos conjecturais nesse contexto e que a sua utilização pode exigir um esforço maior na elicitação de requisitos, porém não excessivamente, pois é necessário que os responsáveis pela especificação identifiquem incertezas que sem esse instrumento talvez não fossem identificadas nem registradas. A explicitação dessas informações auxilia na compreensão das incertezas pelos responsáveis, bem como na sua priorização e atualização, sendo parte do objetivo da ERC.

Como lições aprendidas, o FERC foi evoluído com o acréscimo de uma nova seção para registro dos requisitos associados ao requisito conjectural. Além disso, também foi criado o ciclo de vida do requisito conjectural, para facilitar o entendimento sobre os conceitos e sobre os estados pelos quais o requisito conjectural pode transitar, e o treinamento sobre o uso do FERC teve sua duração aumentada com um maior detalhamento sobre os conceitos e mais exemplos, de modo a melhorar o entendimento sobre a correta utilização do instrumento. A versão final da ERC construída com base nos estudos descritos nesta dissertação pode ser vista no Apêndice A.

6 Considerações Finais

Este capítulo discute os principais resultados dos estudos conduzidos para avaliar a proposta da Especificação de Requisitos Conjecturais, bem como as considerações finais deste estudo. Também são apresentadas as contribuições e perspectivas futuras.

6.1 Considerações Finais

A Experimentação Contínua (EC) é um método de desenvolvimento de software que propõe o desenvolvimento e a avaliação de características do software através da contínua definição de hipóteses. Baseado na utilização do software, as hipóteses podem ser aceitas ou rejeitadas. Assim, as características do software não são definidas previamente, mas descobertas como resultado da avaliação de dados de uso real.

A pesquisa sobre EC tem crescido nos últimos anos, bem como a sua utilização tanto por pequenas quanto por grandes organizações (Auer *et al.*, 2021). Diversas técnicas também têm sido propostas com o objetivo de aprimorar a sua eficiência. Porém, essas propostas geralmente se concentram na fase de execução dos experimentos e de análise de resultados. Raros estudos tratam do gerenciamento das incertezas sobre os requisitos a serem desenvolvidos, as quais serão atualizadas ao longo do processo de desenvolvimento e experimentação como resultado dos aprendizados obtidos.

Essa dissertação propôs a categoria de requisitos conjecturais para organizar as incertezas sobre características do software, e apresentou a técnica denominada Especificação de Requisitos Conjecturais (ERC). Ela é composta por dois instrumentos: o Formato de Escrita de Requisitos Conjecturais (FERC) e o Quadro de Experimentação para Suposições de Solução (QESS). O objetivo desejado de um formato de descrição de requisitos que possuem incertezas é atendido pelo FERC, e o objetivo de um formato de descrição dos aprendizados gerados sobre as incertezas por meio de experimentação é atendido pelo QESS.

A utilização da ERC visa apoiar o desenvolvimento de software com experimentação contínua e a tomada de decisão sobre a definição das características do

software enquanto elas ainda possuírem incertezas. O público-alvo da proposta são os responsáveis pela identificação dos requisitos e os tomadores de decisões sobre as características do software.

Uma versão inicial dos instrumentos da ERC foi utilizada para capturar em retrospectiva (prova de conceito) os requisitos conjecturais do projeto OxímetroIoT, desenvolvido na Universidade Federal do Rio de Janeiro. A partir dos aprendizados gerados por essa prova de conceito, os instrumentos foram evoluídos e o FERC foi avaliado por um estudo de viabilidade. Esse estudo foi realizado com estudantes de graduação em Engenharia de Computação e Informação da mesma universidade no desenvolvimento do projeto *daily Huddle*, tendo os seus resultados observados e a experiência avaliada qualitativamente por meio de um questionário. Novamente, o aprendizado gerado foi utilizado para evoluir a proposta da ERC.

Os resultados do estudo de viabilidade indicaram que o FERC foi percebido como suficiente para o registro dos requisitos conjecturais no contexto no qual foi realizado. Além disso, a avaliação das respostas ao questionário indicou que o instrumento pode facilitar a identificação de incertezas que poderiam não ser percebidas com antecedência.

6.2 Contribuições

As contribuições gerais obtidas nesse estudo estão listadas de forma objetiva:

1. Definição dos conceitos de **Requisitos Conjecturais** e **Suposições de Solução** para apoiar a identificação e organização das incertezas sobre características do software e as hipóteses a serem experimentadas.
2. Concepção de um modelo para registro dos requisitos conjecturais (**Formato de Escrita dos Requisitos Conjecturais** – FERC) passível de atualizações com o objetivo de: apoiar a especificação de características do software que possuem incertezas e apoiar a recuperação do conhecimento sobre os requisitos conjecturais.
3. Concepção de um documento para registro dos objetivos e aprendizados obtidos como resultado de um ciclo experimental (**Quadro de Experimentação para Suposições de Solução** – QESS), com o objetivo

de apoiar a atualização dos aprendizados e as incertezas dos requisitos conjecturais.

6.3 Limitações da Pesquisa

As principais limitações desta dissertação são:

1. As conclusões do estudo de viabilidade não podem ser generalizadas, pois o estudo foi realizado em ambiente de sala de aula em um projeto, embora real, tratado em ambiente acadêmico.
2. Os benefícios da aplicação do FERC não podem ser generalizados para outros contextos além do qual foi avaliado, a saber: ambiente acadêmico com utilização por estudantes de graduação em engenharia de computação e informação.
3. O instrumento QESS não foi avaliado pelo estudo de viabilidade, portanto, não há observação sobre seus eventuais benefícios ou oportunidades de melhoria.

6.4 Publicações

Ao longo da realização deste trabalho, duas publicações relacionadas com Experimentação Contínua foram preparadas. Elas foram utilizadas como base para a identificação das lacunas na especificação de requisitos conjecturais nesse contexto:

1. Erthal, V. M., Souza, B. P. de, Santos, P. S. M. dos, Travassos, G. H.: “A Literature Study to Characterize Continuous Experimentation in Software Engineering” (2022). Anais do XXV Congresso Ibero-Americano em Engenharia de Software (pp. 1-15). SBC.
2. Erthal, V. M., Souza, B. P. de, Santos, P. S. M. dos, Travassos, G. H.: "Characterization of Continuous Experimentation in Software Engineering: Expressions, Models, and Strategies." Science of Computer Programming (2023): 102961.

6.5 Perspectivas Futuras

As perspectivas futuras para evolução da proposta de Especificação de Requisitos Conjecturais são as seguintes:

1. Propor um processo de desenvolvimento com experimentação contínua que implemente os conceitos e instrumentos da ERC, realizando estudos experimentais para validação.
2. Propor uma infraestrutura de apoio computacional para os instrumentos FERC e QESS a fim de melhorar a sua aplicabilidade e usabilidade, realizando estudos experimentais com o apoio computacional proposto.

Além disso, tendo em vista (Shull *et al.*, 2001), seria importante avaliar a técnica por meio das seguintes formas:

3. Estudo de observação para avaliação da efetividade dos instrumentos e da ordem de preenchimento.
4. Estudo de caso em um projeto inserido em um contexto real de desenvolvimento de software para observar a interação dos instrumentos com o processo de desenvolvimento e as adaptações necessárias a diferentes contextos.
5. Estudo de caso na indústria para descobrir eventuais interações negativas não previstas dos instrumentos com processos e necessidades da indústria.

Referências

(Aguilar Calderón *et al.*, 2016) Aguilar Calderón, J. A., Garrigós, I., Mazón, J.-N.: "Requirements Engineering in the Development Process of Web Systems: A Systematic Literature Review"; Acta Polytechnica Hungarica, 13(3), 61–80 (2016).

(Aitken, 1969) Aitken, R. C.: "Measurement of feelings using visual analogue scales"; Proc R Soc Med.; 62(10):989-93 (1969).

(Arif *et al.*, 2009) Arif, S., Khan, Q., Gahyyur, S. A. K.: "Requirements engineering processes, tools/technologies, & methodologies"; International Journal of Reviews in Computing, 2(6), 41–56 (2009).

(Auer e Felderer, 2018) Auer, F., Felderer, M.: “Current state of research on continuous experimentation: a systematic mapping study”; 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE (2018).

(Auer *et al.*, 2021) Auer, F., Ros, R., Kaltenbrunner, L., Runeson, P., Felderer, M.: “Controlled experimentation in continuous experimentation: Knowledge and challenges,” Information and Software Technology 134 (2021), 106551.

(Basili *et al.*, 1994) Basili, V. R., Caldiera, G., Rombach, H. D.: "The goal question metric approach"; Encyclopedia of software engineering (1994), 528-532.

(Beck *et al.*, 2021) Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D.: "The agile manifesto"; https://agilemanifesto.org/ (2001), acesso em 30/05/2023

(Blank, 2013) Blank, S.: "Why the Lean Start-Up Changes Everything"; https://hbr.org/2013/05/why-the-lean-start-up-changes-everything (2013), acesso em 04/08/2023.

(Brhel *et al.*, 2015) Brhel, M., Meth, H., Maedche, A., Werder, K.: "Exploring principles of user-centered agile software development: A literature review"; Information and software technology, 61, 163-181 (2015).

(Bosch e Olsson, 2016) Bosch, J., Olsson, H. H.: “Data-driven continuous evolution of smart systems”; 2016 IEEE/ACM 11th International Symposium on Software

Engineering for Adaptive and Self-Managing Systems (SEAMS), IEEE, Austin, USA, pp. 28-34 (2016).

(Bosch e Olsson, 2017) Bosch, J., Olsson, H. H.: “Toward evidence-based organizations: lessons from embedded systems, online games, and the Internet of Things”; IEEE Software 34.5 (2017), 60-66.

(Bosch, 2012) Bosch, J.: “Building products as innovation experiment systems”; International Conference of Software Business, Springer, Berlin, Heidelberg (2012).

(Cole, 2001) Cole, R. E.: "From continuous improvement to continuous innovation"; Quality Management Journal 8.4 (2001), 7-21.

(Crook *et al.*, 2009) Crook, T., Frasca, B., Kohavi, R., Longbotham, R.: “Seven pitfalls to avoid when running controlled experiments on the web,” Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, Paris, France, pp. 1105-1114 (2009).

(Endres e Rombach, 2003) Endres, A., Rombach, D.: "A Handbook of Software and Systems Engineering – Empirical Observations, Laws and Theorie"; Fraunhofer – Pearson/Addison Wesley, (2003). ISBN: 0321154207

(Erthal *et al.*, 2022) Erthal, V. M., Souza, B. P. de, Santos, P. S. M. dos, Travassos, G. H.: “A Literature Study to Characterize Continuous Experimentation in Software Engineering”. Anais do XXV Congresso Ibero-Americano em Engenharia de Software, Córdoba, Argentina, pp. 1-15, SBC (2022).

(Erthal *et al.*, 2023) Erthal, V. M., Souza, B. P. de, Santos, P. S. M. dos, Travassos, G. H.: "Characterization of Continuous Experimentation in Software Engineering: Expressions, Models, and Strategies." Science of Computer Programming: 102961 (2023).

(Esteller-Cucala *et al.*, 2020) Esteller-Cucala, M., Fernandez, V., Villuendas, D.: “Towards data-driven culture in a Spanish automobile manufacturer: A case study”; Journal of Industrial Engineering and Management 13.2 (2020), 228-245.

(Fabijan *et al.*, 2017) Fabijan, A., Dmitriev, P., Olsson, H. H., Bosch, J.: “The evolution of continuous experimentation in software product development: from data to a data-

driven organization at scale"; 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires Argentina, IEEE, pp. 770-780 (2017).

(Fabijan *et al.*, 2018) Fabijan, A., Dmitriev, P., McFarland, C., Vermeer, L., Holmström Olsson, H., Bosch, J.: "Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies"; Journal of Software: Evolution and Process 30.12 (2018), e2113.

(Fagerholm *et al.*, 2014) Fagerholm, F., Guinea, A. S., Mäenpää, H., Münch, J.: "Building blocks for continuous experimentation"; Proceedings of the 1st international workshop on rapid continuous software engineering (2014).

(Fagerholm *et al.*, 2017) Fagerholm, F., Guinea, A. S., Mäenpää, H., Münch, J.: "The RIGHT model for continuous experimentation"; Journal of Systems and Software 123 (2017), 292-305.

(Feitelson *et al.*, 2013) Feitelson, D. G., Frachtenberg, E., Beck, K. L.: "Development and deployment at Facebook"; IEEE Internet Computing 17.4 (2013), 8-17.

(Fitzgerald e Stol, 2017) Fitzgerald, B., Stol, K.J.: "Continuous software engineering: A roadmap and agenda"; Journal of Systems and Software 123 (2017), 176-189.

(Gerostathopoulos *et al.*, 2018) Gerostathopoulos, I., Prehofer, C., Bulej, L., Bureš, T., Horký, V., Tuma, P.: "Cost-aware stage-based experimentation: challenges and emerging results"; 2018 IEEE International Conference on Software Architecture Companion (ICSA-C), Seattle, USA, IEEE, pp. 72-75 (2018).

(Giaimo *et al.*, 2016) Giaimo, F., Yin, H., Berger, C., Crnkovic, I.: "Continuous experimentation on cyber-physical systems: challenges and opportunities"; Proceedings of the scientific workshop proceedings of XP2016, Edinburgh, Scotland, pp. 1-2 (2016).

(Giaimo *et al.*, 2020) Giaimo, F., Andrade, H., Berger, C.: "Continuous experimentation and the cyber-physical systems challenge: An overview of the literature and the industrial perspective"; Journal of Systems and Software 170 (2020), 110781.

(Gomez-Uribe e Hunt, 2015) Gomez-Uribe, C. A., Hunt, N.: "The Netflix recommender system: Algorithms, business value, and innovation"; ACM Transactions on Management Information Systems (TMIS) 6.4 (2015), 1-19.

(Karvonen *et al.*, 2015) Karvonen, T., Lwakatare, L. E., Sauvola, T., Bosch, J., Olsson, H. H., Kuvaja, P., Oivo, M.: “Hitting the Target: Practices and Steps for Moving Towards Innovation Experiment Systems”; Lecture Notes in Business Information Processing (2015).

(Kohavi *et al.*, 2007) Kohavi, R., Henne, R. M., Sommerfield, D.: “Practical guide to controlled experiments on the web: listen to your customers, not to the hippo”; Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, San Jose, USA, pp. 959-967 (2007).

(Kohavi *et al.*, 2009) Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R. M.: “Controlled experiments on the web: survey and practical guide”; Data mining and knowledge discovery 18.1 (2009), 140-181.

(Kohavi *et al.*, 2013) Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., Pohlmann, N.: “Online controlled experiments at large scale”; Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, Chicago, USA, pp. 1168-1176 (2013).

(Lindgren e Münch, 2016) Lindgren, E., Münch, J.: “Raising the odds of success: the current state of experimentation in product development”; Information and Software Technology 77 (2016), 80-91.

(Madampe, 2017) Madampe, M. A. K. G.: "Successful adoption of agile project management in software development industry"; International Journal of Computer Science and Information Technology Research, 5(4), 2327 (2017).

(Marques e da Cunha, 2019) Marques, J., da Cunha, A. M.: "Ares: An agile requirements specification process for regulated environments"; International Journal of Software Engineering and Knowledge Engineering, 29(10), 1403-1438 (2019).

(Mattos *et al.*, 2021) Mattos, D. I., Dakkak, A., Bosch, J., Olsson, H. H.: "The HURRIER process for experimentation in business-to-business mission-critical systems"; Journal of Software: Evolution and Process, e2390 (2021).

(Melegati e Wang, 2019) Melegati, J., Wang, X.: "QUESt: new practices to represent hypotheses in experiment-driven software development"; Proceedings of the 2nd ACM

SIGSOFT International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems, Tallinn, Estonia, pp. 13-18 (2019).

(Melegati e Wang, 2020) Melegati, J., Wang, X.: “Hypotheses Elicitation in Early-Stage Software Startups Based on Cognitive Mapping”; International Conference on Agile Software Development, Springer, Cham, Copenhagen, Denmark, pp. 211-220 (2020).

(Melegati *et al.*, 2019a) Melegati, J., Wang, X., Abrahamsson, P.: “Hypotheses Engineering: first essential steps of experiment-driven software development”; 2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation, and Evolution (RCoSE/DDrEE), IEEE, Montreal, Canada, pp. 16-19 (2019).

(Melegati *et al.*, 2019b) Melegati, J., Chanin, R., Wang, X., Sales, A., Prikladnicki, R.: “Enablers and inhibitors of experimentation in early-stage software startups”; International Conference on Product-Focused Software Process Improvement, Springer, Cham, Barcelona, Spain, pp. 554-569 (2019).

(Melegati *et al.*, 2020) Melegati, J., Guerra, E., Wang, X.: “Understanding Hypotheses Engineering in Software Startups through a Gray Literature Review”; Information and Software Technology (2020), 106465.

(Niculescu *et al.*, 2021) Niculescu, I., Hu, H. M., Gee, C., Chong, C., Dubey, S., Li, P. L.: "Towards inclusive software engineering through A/B testing: a case-study at windows"; 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Madrid, Spain, pp. 180-187 (2021).

(Nuseibeh e Easterbrook, 2000) Nuseibeh, B., Easterbrook, S.: "Requirements engineering: A roadmap"; Proceedings of the Conference on The Future of Software Engineering - ICSE ’00, Limerick, Ireland, pp. 35-46 (2000).

(Olsson and Bosch, 2013) Olsson, H. H., Bosch, J.: “Post-deployment data collection in software-intensive embedded products”; International Conference of Software Business, Springer, Berlin, Heidelberg (2013).

(Olsson e Bosch, 2014) Olsson, H. H., Bosch, J.: “From opinions to data-driven software R&D: A multi-case study on how to close the ‘open loop’ problem”; 2014 40th

EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE, Verona, Italy, pp. 9-16 (2014).

(Olsson e Bosch, 2019) Olsson, H. H., Bosch, J.: “Data-driven development: Challenges in online, embedded and on-premise software”; International Conference on Product-Focused Software Process Improvement, Springer, Cham, Barcelona, Spain, pp. 515-527 (2019).

(Olsson *et al.*, 2017) Olsson, H. H., Bosch, J., Fabijan, A.: “Experimentation that matters: a multi-case study on the challenges with A/B testing”; International Conference of Software Business, Springer, Cham, Essen, Germany, pp. 179-185 (2017).

(O’Reilly, 2013) O'Reilly, B.: "How to implement Hypothesis-Driven Development"; https://barryoreilly.com/explore/blog/how-to-implement-hypothesis-driven-development/ (2013), acesso em 30/05/2023.

(Pandey *et al.*, 2010) Pandey, D., Suman U., Ramani, A. K.: "An effective requirement engineering process model for software development and requirements management." International Conference on Advances in Recent Technologies in Communication and Computing. IEEE, Kottayam, India, pp. 287-291 (2010).

(Pfleeger e Atlee, 2010) Pfleeger, S. L., Atlee, J. M.: "Software engineering: theory and practice"; Pearson Education India (2010).

(Pressman, 2016) Pressman, R. S.: "Software engineering: A practitioner’s approach" (8th ed); McGraw Hill Brasil (2016).

(Rehman *et al.*, 2013) Rehman, T. ur, Khan, M. N. A., Riaz, N.: "Analysis of Requirement Engineering Processes, Tools/Techniques and Methodologies"; International Journal of Information Technology and Computer Science, 5(3), 40–48 (2013).

(Ries, 2011) Ries, E.: "The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses"; Currency (2011).

(Rissanen e Münch, 2015) Rissanen, O., Münch, J.: “Continuous experimentation in the B2B domain: a case study”; 2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering, IEEE, Florence, Italy, pp. 12-18 (2015).

(Sauvola *et al.*, 2015) Sauvola, T., Lwakatare, L. E., Karvonen, T., Kuvaja, P., Olsson, H. H., Bosch, J., Oivo, M.: “Towards customer-centric software development: a multiple-case study”; 2015 41st Euromicro Conference on Software Engineering and Advanced Applications, IEEE, Madeira, Portugal, pp. 9-17 (2015).

(Schermann *et al.*, 2018) Schermann, G., Cito, J., Leitner, P.: “Continuous experimentation: challenges, implementation techniques, and current research”; Ieee Software 35.2 (2018), 26-31.

(Schwaber e Sutherland, 2011) Schwaber, K., Sutherland, J.: "The scrum guide"; Scrum Alliance (2011).

(Shull *et al.*, 2001) Shull, F., Carver, J., Travassos, G.H.: "An Empirical Methodology for Introducing Software Processes", Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering, Vienna, Austria, pp. 288-296, 2001.

(Sommerville, 2020) Sommerville, I.: "Engineering software products"; Vol. 355, London: Pearson (2020).

(Sullivan, 2021) Sullivan, C.: "Hypothesis Kit V4"; https://optimiseordie.medium.com/hypothesis-kit-v4-4a1441f77ddc (2021), acesso em 30/05/2023.

(Sutcliffe e Sawyer, 2013) Sutcliffe, A., Sawyer, P.: "Requirements elicitation: Towards the unknown unknowns"; 2013 21st IEEE International Requirements Engineering Conference (RE), IEEE, Rio de Janeiro, Brazil, pp. 92-104 (2013).

(Valente, 2020) Valente, M. T.: "Engenharia de software moderna, Princípios e Práticas para Desenvolvimento de Software com Produtividade" (2020).

(Vegendla *et al.*, 2018) Vegendla, A., Duc, A. N., Gao, S., Sindre, G.: "A Systematic Mapping Study on Requirements Engineering in Software Ecosystems"; Journal of Information Technology Research, 11(1), 4:1-4:21 (2018).

(Weigel, 2022) Weigel, E.: "Hypothesis template"; https://erindoesthings.com/index.php/2022/08/04/experimentation-hypothesis-template/ (2022), acesso em 30/05/2023.

APÊNDICE A - Versão Final da Especificação de Requisitos Conjecturais

1. Formato de Escrita dos Requisitos Conjecturais (FERC)

Quadro 35 – Versão final do FERC.

Requisitos Conjecturais					
ID	*RC[id]*	**Situação**	*[Proposto, Aprovado, ou Cancelado]*	**Prioridade**	*[Prioridade]*
Requisitos Associados	*[Referências para outros requisitos associados]*				
Descrição	**Espera-se que o sistema de software possua** *[atributo desejado]* **De modo que** *[necessidade do atributo desejado]* **Porém, não sabemos:** ● **Incerteza In***[id]***a:** *[incerteza associada com este requisito]* ● **Incerteza In***[id]***b:** *[incerteza associada com este requisito]* ● ... **Já aprendemos que:** ● **Ap***[id]***a:** *[aprendizado sobre as incertezas deste requisito conjectural]* ● **Ap***[id]***b:** *[aprendizado sobre as incertezas deste requisito conjectural]* ● ...				
[O modelo se repete para cada requisito conjectural]					

2. Quadro de Experimentação para Suposições de Solução (QESS)

Quadro 36 – Versão final do QESS.

Ciclo Experimental CE*[id-ce]*
Expectativa do Ciclo Experimental
Esperamos que *[descrição da suposição de solução]* **Resulte na atualização das incertezas sobre** *[incertezas que serão avaliadas]* **Como resultado de** *[descrição da observação e análise que resultará na atualização das incertezas]*
Requisitos Conjecturais do Ciclo Experimental
● RC*[id]*: Incerteza In*[id]* (versão *[n]*) ● RC*[id]*: Incerteza In*[id]* (versão *[n]*) ● …
Resultados do Ciclo Experimental

- **Incerteza In*[id][x]*:** *[descrição do aprendizado dos resultados da suposição de solução executada nesta iteração para essa incerteza]*
- **Incerteza In*[id][y]*:** *[descrição do aprendizado dos resultados da suposição de solução executada nesta iteração para essa incerteza]*
- ...

3. Vídeo de Treinamento sobre a ERC

O vídeo de treinamento produzido sobre a ERC e seus instrumentos pode ser acessado através do seguinte endereço eletrônico: https://zenodo.org/record/8368316/files/Treinamento%20Final%20ERC.mp4?download=1.

APÊNDICE B - Materiais Referentes ao Estudo de Viabilidade

Os materiais disponibilizados para os grupos a fim de contextualizar o estudo de viabilidade apresentado no Capítulo 5 podem ser encontrados nos seguintes endereços eletrônicos:

Apresentação do projeto *daily Huddle*	https://zenodo.org/record/8367625/files/01%20-%20Apresenta%C3%A7%C3%A3o%20Huddle.pdf?download=1
Descrição inicial da terceira iteração de desenvolvimento	https://zenodo.org/record/8367625/files/02%20-%20PROJETO%20HUDDLE%20-%20Descri%C3%A7%C3%A3o%20inicial.pdf?download=1
Cenário do Dashboard	https://zenodo.org/record/8367625/files/03%20-%20Cen%C3%A1rio%20do%20Dashboard.pdf?download=1
Cenário do Subsistema de Equipamentos	https://zenodo.org/record/8367625/files/04%20-%20Cen%C3%A1rio%20do%20Subsistema%20de%20Equipamentos.pdf?download=1
Cenário do Controle de Materiais	https://zenodo.org/record/8367625/files/05%20-%20Cen%C3%A1rio%20do%20Controle%20de%20Materiais.pdf?download=1
Treinamento sobre requisitos conjecturais e FERC	https://zenodo.org/record/8368316/files/Treinamento%20FERC.mp4?download=1

Semelhantemente, os materiais produzidos como resultado do estudo de viabilidade podem ser encontrados nos seguintes endereços eletrônicos:

Protocolo para Estudo de Viabilidade	https://zenodo.org/record/8368183/files/Protocolo%20para%20Estudo%20de%20Viabilidade%20do%20Formato%20de%20Escrita%20dos%20Requisitos%20Conjecturais.pdf?download=1
Documento de Requisitos Completo	https://zenodo.org/record/8367625/files/06%20-%20Documento%20de%20Requisitos%20Completo.pdf?download=1
Resultados do Questionário	https://zenodo.org/record/8367625/files/07%20-%20Resultados%20do%20Question%C3%A1rio.xlsx?download=1