

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

WILLIAM SIMÃO DE DEUS

**A CARACTERIZAÇÃO DAS MICROTASKS APLICADAS AO
DESENVOLVIMENTO DE SOFTWARE CROWDSOURCING**

DISSERTAÇÃO DE MESTRADO

CORNÉLIO PROCÓPIO

2018

WILLIAM SIMÃO DE DEUS

**A CARACTERIZAÇÃO DAS MICROTASKS APLICADAS AO
DESENVOLVIMENTO DE SOFTWARE CROWDSOURCING**

Dissertação De Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR como requisito parcial para obtenção do grau de “Mestre em Informática” – Área de Concentração: Computação Aplicada.

Orientador: Prof. Dr. Alexandre L’Erario

CORNÉLIO PROCÓPIO

2018

Dados Internacionais de Catalogação na Publicação

D486 Deus, William Simão de

A caracterização das microtasks aplicadas ao desenvolvimento de software crowdsourcing / William Simão de Deus. – 2018.
130 f. : il. color. ; 31 cm.

Orientador: Alexandre L'erário.
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. Cornélio Procópio, 2018.
Bibliografia: p. 120-127.

1. Software – Desenvolvimento. 2. Colaboração acadêmico-industrial. 3. Computação humana. 4. Informática – Dissertações. I. L'erário, Alexandre, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD (22. ed.) 004

Biblioteca da UTFPR - Câmpus Cornélio Procópio

Bibliotecários/Documentalistas responsáveis:
Simone Fidêncio de Oliveira Guerra – CRB-9/1276
Romeu Righetti de Araujo – CRB-9/1676



Título da Dissertação Nº 42:

“A CARACTERIZAÇÃO DAS MICROTASKS APLICADAS AO DESENVOLVIMENTO DE SOFTWARE CROWDSOURCING”.

por

William Simão de Deus

Orientador: **Prof. Dr. Alexandre L'Erário**

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM INFORMÁTICA – Área de Concentração: Computação Aplicada, pelo Programa de Pós-Graduação em Informática – PPGI – da Universidade Tecnológica Federal do Paraná – UTFPR – Câmpus Cornélio Procópio, às 15h do dia 19 de fevereiro de 2018. O trabalho foi _____ pela Banca Examinadora, composta pelos professores:

Prof. Dr. Alexandre L'Erário
(Presidente – UTFPR-CP)

Prof. Dr. André Leme Fleury
(USP-SP)

Prof. Dr. Willian Massami Watanabe
(UTFPR-PR)

Visto da coordenação:

Danilo Sipoli Sanches
Coordenador do Programa de Pós-Graduação em Informática
UTFPR Câmpus Cornélio Procópio

A Folha de Aprovação assinada encontra-se na Coordenação do Programa.

Para a minha irmã Adeline e meus pais Valmor e Edelzina

AGRADECIMENTOS

O desenvolvimento deste trabalho, além de numerosas reflexões, também revelou diversas pessoas talentosas e generosas, que contribuíram de alguma forma durante o seu decorrer. Assim, gostaria de externar alguns reconhecimentos.

A condução deste projeto só foi possível graças ao meu orientador, professor Alexandre L'Erario, que direcionou todo o trabalho e contribuiu de forma incomensurável por meio de suas revisões, considerações e correções sempre pertinentes.

O indispensável auxílio prestado pela Renata Marques Barros, Hellen Christine Seródio Thomazinho e Heydi Miura Machado durante a leitura, apresentação e tradução dos artigos merece profunda gratidão.

Agradeço aos professores membros das bancas de qualificação e defesa - André Takeshi Endo, Willian Massami Watanabe e André Leme Fleury - pela leitura e as valorosas observações, críticas e sugestões.

Algumas pessoas colaboraram para este trabalho de forma indireta. Neste sentido, gostaria de prestar reconhecimento ao pessoal da Forlogic Software, em especial ao Jackson Alexandre, que tornou possível a minha realocação. Aproveito e estendo estes agradecimentos aos colegas que conheci durante todo este percurso, sobretudo aos colegas do tempo de república, pelas tantas conversas disruptivas.

Agradeço imensamente a Felipe Soares da Silva, pela grande cooperação na reta final deste percurso.

Minha irmã, Adeline Simão de Deus, despendeu grande parte de seu tempo para a leitura e aperfeiçoamento deste trabalho. Sempre recorri ao seu auxílio e fui prontamente atendido. Assim, gostaria de destacar imensamente a sua dedicação.

Meus pais, Edelzina da Aparecida Baptista de Deus e Valmor Simão de Deus, representam o guia das minhas ações. Ambos me ensinaram a importância da educação na vida de qualquer pessoa, além de seus sábios ensinamentos, por tudo isso sou extremamente grato.

Essas últimas linhas de agradecimento gostaria de dedicar à Deus, por me conceder clareza de pensamento em todos os momentos necessários desta caminhada.

[...] vou deitar ao papel as reminiscências que me vierem vindo. Deste modo, viverei o que vivi, e assentarei a mão para alguma obra de maior tomo.

Machado de Assis, *Dom Casmurro* (1899).

RESUMO

DEUS, William Simão de. A CARACTERIZAÇÃO DAS MICROTASKS APLICADAS AO DESENVOLVIMENTO DE SOFTWARE CROWDSOURCING. 131 f. Dissertação De Mestrado – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Contexto: o desenvolvimento de software é transformado conforme o avanço das tecnologias e as particularidades que cada projeto pode apresentar. Atualmente, existe uma tendência do setor produtivo pela adoção do *crowdsourcing* (CS) na criação de produtos e projetos de software, sendo denominado como desenvolvimento de software CS. Essa tendência aplica diversos conceitos do desenvolvimento distribuído e do desenvolvimento *open source*. Apesar disso, as suas atividades, conhecidas como “*microtasks*”, ainda representam a causa de muita contradição e confusão dentro da literatura. **Objetivo:** o objetivo deste trabalho foi investigar a caracterização das *microtasks* apoiado em quatro pilares: o uso, as características, os contrastes e a complexidade das *microtasks*. **Método:** os métodos e procedimentos adotados na condução deste estudo representaram uma abordagem híbrida de validação. Inicialmente, foi conduzido um estudo de caso para investigar o uso das *microtasks* no desenvolvimento de software CS. Com base na experiência fornecida pelo caso analisado, a literatura foi consultada para adicionar insumos sobre as características de uma *microtasks* e comparar seus contrastes com as atividades de software. Finalmente, foi adotada a experimentação controlada para modelar e validar uma abordagem de aferição de complexidade das *microtasks*. **Resultados:** os resultados demonstraram a amplitude de uso das *microtasks* em diferentes cenários e etapas do ciclo de vida de um projeto CS; uma abrangente taxonomia sobre as características e estados que uma *microtask* pode possuir; a liberdade de desenvolvimento que as *microtasks* fornecem em relação as atividades de software executadas no desenvolvimento distribuído; e uma abordagem capaz de mensurar o esforço de execução de uma *microtask*. **Conclusão:** com a condução desta pesquisa, foi possível identificar que as *microtasks* representam uma tendência do desenvolvimento de software CS, fornecendo contribuições empíricas sobre a sua aplicação e execução, além de contribuições teóricas, sobre a sua estrutura e comportamento.

Palavras-chave: Microtasks, Crowdsourcing, Desenvolvimento de Software

ABSTRACT

DEUS, William Simão de. MICROTASKS CHARACTERIZATION APPLIED TO CROWD-SOURCING SOFTWARE DEVELOPMENT. 131 f. Dissertação De Mestrado – Programa de Pós-graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Background: software development has been transformed with the progress of technologies and the particularities that each project can present. Currently, there is a tendency of the productive sector for the adoption of crowdsourcing (CS) in the creation of products and software projects being denominated as CS software development. This trend applies several concepts of distributed and open source development. Despite this, its activities known as “microtasks”, still represent the cause of much contradiction and confusion within the literature. **Objective:** the aim of this work was to investigate the characterization of microtasks based on four pillars: the use, characteristics, contrasts and complexity of microtasks. **Method:** the methods and procedures adopted in conducting this study represented a hybrid validation approach. Initially, a case study was conducted to investigate the use of microtasks in CS software development. Based on the experience provided by the analyzed case, the literature was consulted in order to add inputs on the characteristics of a microtask and to compare their contrasts with the software activities. Finally, controlled experimentation was used to model and validate a microtask complexity assessment approach. **Results:** the results demonstrated the breadth of use of microtasks in different scenarios and steps of the life cycle of a CS project; a comprehensive taxonomy on the characteristics and states that a microtask may possess; the freedom of development that microtasks provide in relation to software activities performed in distributed development; and an approach capable of measuring the execution effort of a microtask. **Conclusion:** with the conduction of this research, it was possible to identify that microtasks represent a trend of CS software development, providing empirical contributions on its application and execution as well as theoretical contributions on its structure and behavior.

Keywords: Microtasks, Crowdsourcing, Software Development

LISTA DE FIGURAS

FIGURA 1	– O funcionamento do <i>crowdsourcing</i>	20
FIGURA 2	– Os modelos de desenvolvimento de software na visão de diferentes autores	21
FIGURA 3	– O processo de criação de <i>microtask</i>	24
FIGURA 4	– O processo de buscas do mapeamento sistemático	28
FIGURA 5	– Estudos analisados	29
FIGURA 6	– Relação de publicações por ano.	30
FIGURA 7	– Domínios das <i>microtasks</i> reveladas pelo mapeamento sistemático	34
FIGURA 8	– Métodos e procedimentos de validação	36
FIGURA 9	– Fontes publicadoras	37
FIGURA 10	– Ilustração do processo de desenvolvimento tradicional e o <i>crowdsourcing</i>	54
FIGURA 11	– O tipo de uso das <i>microtasks</i>	56
FIGURA 12	– A porcentagem e o uso das <i>microtasks</i> no ciclo de desenvolvimento	57
FIGURA 13	– As plataformas usadas nas <i>microtasks</i>	59
FIGURA 14	– As tecnologias presentes nas <i>microtasks</i>	60
FIGURA 15	– Os mecanismos para auxiliar a execução de <i>microtasks</i>	61
FIGURA 16	– O índice de finalização e falha na execução de <i>microtasks</i>	62
FIGURA 17	– O tamanho dos <i>crowds</i> de forma detalhada	63
FIGURA 18	– A influência das submissões nas <i>microtasks</i>	64
FIGURA 19	– A influência da premiação nas <i>microtasks</i>	64
FIGURA 20	– A influência dos dias disponíveis nas <i>microtasks</i>	65
FIGURA 21	– O total de <i>microtasks</i> dos projetos de software <i>crowdsourcing</i>	66
FIGURA 22	– O total de tecnologias presentes nas <i>microtasks</i>	66
FIGURA 23	– As características das <i>microtasks</i>	73
FIGURA 24	– Metamodelo sobre características das <i>microtasks</i>	74
FIGURA 25	– Comparação dos modelos de desenvolvimento de software <i>crowdsourcing</i> e distribuído	79
FIGURA 26	– Fluxograma de conversão das características e estados das <i>microtasks</i> ..	97
FIGURA 27	– Montagem dos robôs <i>Lego Mindstorms</i> grupo “A” (acima) e grupo “B” (abaixo)	100
FIGURA 28	– Exemplo de digrama gerado pelo <i>crowd</i>	101
FIGURA 29	– Estrutura do repositório gerado pelo <i>crowd</i>	101
FIGURA 30	– A detecção de <i>outliers</i> das amostras	110
FIGURA 31	– A dispersão das amostras de ordenação e de interesse que apresentaram correlação forte com o índice de execução	112
FIGURA 32	– A dispersão das amostras de especialidade e de complexidade que apresentaram correlação extremamente forte com o índice de execução	112

LISTA DE TABELAS

TABELA 1	– Bases digitais selecionadas	27
TABELA 2	– Termos selecionados	27
TABELA 3	– Montagem final da <i>string</i> de pesquisa	27
TABELA 4	– Critérios de inclusão e exclusão	29
TABELA 5	– Relação de autores analisados	37
TABELA 6	– Relação de autores analisados	37
TABELA 7	– Comparação entre os trabalhos similares	38
TABELA 8	– <i>Template</i> resumido para conduzir um estudo de caso na Engenharia de Software	42
TABELA 9	– <i>Design</i> do estudo de caso	43
TABELA 10	– Protocolo resumido para conduzir um experimento controlado na Engenharia de Software	46
TABELA 11	– Protocolo do primeiro teste experimental	47
TABELA 12	– Protocolo do segundo teste experimental	49
TABELA 13	– Protocolo do experimento controlado	51
TABELA 14	– Visão geral dos métodos, procedimentos, contribuições e divulgação dos resultados	52
TABELA 15	– Finalidade e Categoria das <i>microtasks</i>	57
TABELA 16	– Taxonomia das características das <i>microtasks</i>	70
TABELA 17	– A conversão das características das <i>microtasks</i>	96
TABELA 18	– Exemplo de página gerado pelo <i>crowd</i>	102
TABELA 19	– Complexidade das <i>microtasks</i> utilizadas no experimento controlado ...	104
TABELA 20	– Frequência da complexidade (C_o) das <i>microtasks</i> do experimento controlado	104
TABELA 21	– Frequência do prazo das <i>microtasks</i> do experimento controlado	105
TABELA 22	– Nível de conhecimento dos participantes do experimento controlado ...	105
TABELA 23	– Frequência do nível conhecimento dos participantes sobre modelos de desenvolvimento de software	105
TABELA 24	– Frequência do nível conhecimento dos participantes sobre as tecnologias das <i>microtasks</i>	105
TABELA 25	– Frequência dos aspectos que guiaram os participantes a selecionarem uma <i>microtask</i>	107
TABELA 26	– Amostras utilizadas no teste de hipótese do experimento controlado ...	109
TABELA 27	– Nome das fontes	129
TABELA 28	– <i>Microtasks</i> usadas no experimento controlado	130

LISTA DE SIGLAS

API	<i>Application Programming Interface</i>
BPMN	<i>Business Process Model and Notation</i>
C&K	<i>Chidamber e Kemerer</i>
COCOMO	<i>Constructive Cost Model</i>
CS	<i>Crowdsourcing</i>
CSS	<i>Cascading Style Sheets</i>
DDS	Desenvolvimento Distribuído de Software
HTML	<i>Hypertext Markup Language</i>
MS	Mapeamento Sistemático
PICO	<i>Population, Intervention, Comparison, and Outcomes</i>
QP	Questão de Pesquisa
SQL	<i>Structured Query Language</i>

SUMÁRIO

1 INTRODUÇÃO	14
1.1 MOTIVAÇÃO	15
1.2 OBJETIVOS	16
1.3 ORGANIZAÇÃO DO TEXTO	17
2 REVISÃO BIBLIOGRÁFICA	19
2.1 FUNDAMENTAÇÃO TEÓRICA	19
2.1.1 Desenvolvimento de Software <i>Crowdsourcing</i>	21
2.1.2 <i>Microtasks</i>	22
2.2 MAPEAMENTO SISTEMÁTICO	25
2.2.1 Questões de Pesquisa	25
2.2.2 Buscas	26
2.2.3 Seleção de Critérios	28
2.2.4 Resultados	29
2.3 TRABALHOS RELACIONADOS	38
2.4 CONSIDERAÇÕES FINAIS	39
3 MÉTODOS E PROCEDIMENTOS	41
3.1 ESTUDO DE CASO	41
3.1.1 <i>Design</i>	42
3.1.2 Preparação e Coleta	44
3.1.3 Análise	45
3.2 EXPERIMENTAÇÃO CONTROLADA	45
3.2.1 Primeiro Teste Experimental	47
3.2.2 Segundo Teste Experimental	48
3.2.3 Experimento Controlado	50
3.3 CONSIDERAÇÕES FINAIS	52
4 RESULTADOS	54
4.1 USO	54
4.1.1 A Aplicação das <i>Microtasks</i> no Desenvolvimento de Software <i>Crowdsourcing</i>	56
4.1.2 Como a Configuração do <i>Crowdsourcing</i> Afeta as <i>Microtasks</i>	62
4.1.3 Resumo e Discussões	66
4.2 CARACTERÍSTICAS	69
4.2.1 Taxonomia	69
4.2.2 Metamodelo de uma <i>Microtask</i>	74
4.2.3 Resumo e Discussões	76
4.3 CONTRASTES	78
4.3.1 Definição Adotada	78
4.3.2 Seleção do Modelo de Desenvolvimento e Definição dos Papéis	79
4.3.3 Os Contrastes Identificados	80
4.3.4 Resumo e Discussões	88
4.4 COMPLEXIDADE	90
4.4.1 Métricas Existentes	91

4.4.2	Abordagem Teórica	92
4.4.3	Abordagem Prática	93
4.4.4	Conversão	94
4.4.5	Validação Empírica	99
4.4.5.1	Primeiro teste experimental	99
4.4.5.2	Segundo teste experimental	100
4.4.5.3	Experimento controlado	103
4.4.6	Visão Geral	103
4.4.7	Teste de Hipótese	107
4.4.8	Resumo e Discussões	113
4.5	CONSIDERAÇÕES FINAIS	114
4.5.1	Ameaças à Validade	115
5	CONCLUSÕES	117
5.1	CONTRIBUIÇÕES	118
5.2	LIMITAÇÕES	119
5.3	LACUNAS E TRABALHOS FUTUROS	120
	REFERÊNCIAS	121
	Apêndice A – NOME DAS FONTES RECUPERADAS NO MAPEAMENTO SISTÊMÁTICO	129
	Apêndice B – RELAÇÃO COMPLETA DAS MICROTASKS UTILIZADAS NO EXPERIMENTO CONTROLADO	130

1 INTRODUÇÃO

O desenvolvimento de software é frequentemente transformado pelo avanço das tecnologias e as necessidades de cada projeto, isso ocorre porque a forma de desenvolver um software no passado pode não ser tão satisfatória e eficiente hoje em dia (GUPTA; GOUTTAM, 2017). Atualmente, existe uma emergente tendência em adotar o *crowdsourcing* (CS) para transformar e otimizar o desenvolvimento dos produtos ou projetos de software. Assim, como enfatizaram Saremi et al. (2017), diversos pesquisadores e profissionais da área estão voltando a sua atenção para compreender o fenômeno.

O CS representa um modelo estratégico que busca a divisão de pequenas porções de trabalho para realizar entregas rápidas e paralelas (TUNIO et al., 2017). Basicamente, o CS terceiriza tais porções para um grupo amplo e anônimo de participantes, buscando utilizar a inteligência e a contribuição ao mesmo tempo que procura diminuir custos (STOL et al., 2017).

O funcionamento do CS evidenciou a necessidade de diferentes papéis, para isso, Hosseini et al. (2014) recorreram a definição de quatro pilares nomeados como: o *crowdsourcer*, líder responsável do projeto; o *crowd*, grupo de participantes; a plataforma, sistema responsável por unir *crowd* e *crowdsourcer*; e as atividades, tarefas a serem executadas. Na prática, o *crowdsourcer* define uma atividade e sua premiação. Após isso, a atividade é cadastrada em uma plataforma *web*, responsável por atrair possíveis participantes.

Rapidamente os conceitos do CS foram acoplados ao desenvolvimento de software. Como escreveram Mao et al. (2017), atividades de diversos tipos, como extração de requisitos, *design*, implementação e teste são executadas por meio da aplicação do CS. O fenômeno tem guiado diversas pesquisas, e proporcionado uma nova concepção para a produção de software, denominado “desenvolvimento de software CS” (TUNIO et al., 2017). A literatura não é homogênea quanto a denominação e uso de uma terminologia comum, existindo variações como “desenvolvimento *crowd*” (LATOZA; HOEK, 2015) ou “CS software” (ZHAO et al., 2015). Entretanto, para minimizar erros de compreensão, sempre que necessário, este trabalho adotou a denominação de “desenvolvimento de software CS”.

Existem diferentes benefícios em utilizar o desenvolvimento de software CS, Moodley et al. (2017) lembraram que o responsável por definir o valor da premiação de uma atividade é o *crowdsourcer*, e com isso vantagens econômicas podem ser evidenciadas. Como complemento, Saremi et al. (2017) destacaram que o *crowdsourcer* também pode buscar um conjunto amplo de participantes, alcançando variados níveis de criatividade e expertise. As vantagens também são concentradas nos participantes do *crowd* que se propõem a executar uma atividade. Assim, os participantes possuem liberdade para a seleção de trabalho e podem ser contratados somente de forma eletrônica, dispensando trâmites burocráticos. Por isso, na percepção do *crowd*, Machado et al. (2017) citaram a independência que os participantes possuem no desenvolvimento de software CS e a possibilidade de buscar uma renda extra com atividades de curta duração.

Todavia, a literatura sobre o desenvolvimento de software CS concentra diversas contradições. É o caso do que acontece com os estudos de LaToza e Hoek (2016), Naik (2016) e Li et al. (2015), em que os autores discordam ao definir, classificar e elucidar como ocorre o desenvolvimento de software CS. Pode-se verificar que grande parte da contradição presente na literatura decorre em virtude da dificuldade e escassez de estudos sobre as atividades. Em síntese, diversos autores possuem opiniões similares sobre a definição dos demais papéis do CS (*crowd*, *crowdsourcer* e plataforma *online*) (MAO et al., 2017; LATOZA; HOEK, 2016; LI et al., 2015). No entanto, a compreensão e as consequências das atividades representam uma ampla lacuna literária (DENG et al., 2016).

1.1 MOTIVAÇÃO

As atividades representam um importante elo do desenvolvimento de software CS. Todavia, como alertou Stol e Fitzgerald (2014a), a literatura atesta a ausência de estudos focados nesse tema. Isso pode ser verificado pela amplitude de terminologias distintas que são adotadas para representá-las, como “*microtasks*”, “*tasks*”, “*CS tasks*”, “*complex tasks*” (WINKLER et al., 2017; HOSSEINI et al., 2014; MAO et al., 2013), etc. Tendo em vista esse cenário, para evitar equívocos, este trabalho adotou o termo “*microtasks*” para representar as atividades do desenvolvimento de software CS.

Na atual conjectura sobre o desenvolvimento de software, deve-se destacar que o setor produtivo vem dedicando grande atenção ao uso de *microtasks*. Diversas plataformas estão sendo criadas para suportar a crescente demanda de produção baseada nesses conceitos (TUNIO et al., 2017; DWARAKANATH et al., 2015). Porém, ao mesmo tempo, percebe-se que a falta de estudos concentrados no tema cria um ambiente confuso sobre a organização e estruturação de uma *microtask* (GADIRAJU et al., 2017; TRANQUILLINI et al., 2015).

Na prática, as *microtasks* podem possuir diferentes componentes e finalidades, e com isso serem realizadas em prazo de horas até se estenderem em prazo de semanas, conforme apontaram LaToza e Hoek (2016). Os autores também afirmaram que existem outros aspectos que demonstram alta variação nas *microtasks*, como a sua interdependência e o conhecimento que um participante precisa possuir para executá-la. Contudo, os componentes de uma *microtask* ainda não foram completamente sumarizados, e os trabalhos que buscaram sistematizar dados iniciais, necessitam de atualização devido ao rápido crescimento da área (HOSSEINI et al., 2014).

Existe também uma dificuldade em comparar o comportamento das *microtasks* sob o ponto de vista de uma atividade de software. Enquanto Stol et al. (2017), LaToza e Hoek (2016) classificaram as *microtasks* como atividades curtas e de rápida duração aplicadas no desenvolvimento de software CS, Murray-Rust et al. (2015) buscaram demonstrar que as *microtasks* representam um novo paradigma de desenvolvimento, paralelo e independente ao CS. Porém, grande parte das investigações disponíveis na literatura apenas analisaram aspectos do CS com outros modelos de desenvolvimento, como o distribuído ou *open source*, ao invés de comparar efetivamente as *microtasks* (NAIK, 2016; LI et al., 2015).

Por fim, as considerações de Deng et al. (2016) destacaram como as *microtasks* não representam uma abordagem frequente na literatura sobre desenvolvimento de software CS. Assim, como foi observado por Mao et al. (2017), ainda hoje residem dificuldades em compreender a sua granularidade e o que de fato é uma *microtask* simples, uma *microtask* complexa, e qual o intervalo de variação entre elas. Logo, torna-se inviável a realização de tarefas básicas, como métricas ou abordagens capazes aferir a sua complexidade (LATOZA; HOEK, 2016).

1.2 OBJETIVOS

Tendo em vista o cenário atual, o foco central deste trabalho foi conduzir uma investigação sobre as *microtasks*. Para isso, as lacunas do tema foram analisadas e os seguintes objetivos foram extraídos:

- *Uso das microtasks*: como apresentado anteriormente, as *microtasks* estão sendo amplamente aplicadas no setor produtivo, apesar disso, ainda faltam estudos concentrados no tema. Em vista desse cenário, o objetivo primário foi conduzir uma análise nas *microtasks* para identificar seu uso e aplicação no desenvolvimento de software CS.
- *Características das microtasks*: conforme exposto, as *microtasks* podem possuir diferentes componentes e finalidades. Com isso, o objetivo foi identificar o conjunto de carac-

terísticas comuns nas *microtasks*, ou seja, as características que todas - ou grande parte - das *microtasks* podem possuir.

- *Contrastes das microtasks*: não compreender a real natureza das *microtasks* gera dificuldades e discussões na literatura. Para contornar as controvérsias, o objetivo foi comparar as diferenças que as *microtasks* possuem frente a atividades do desenvolvimento de software.
- *Complexidade das microtasks*: foi apresentado que a dificuldade em compreender a granularidade das *microtasks* impede a condução de tarefas básicas. Em vista desse cenário, o objetivo foi gerar uma abordagem capaz de prever e aferir a complexidade de uma *microtask*.

Para alcançar os objetivos expostos, este trabalho adotou quatro ciclos de pesquisa. Assim, quando um objetivo era alcançado, automaticamente um ciclo era finalizado. Logo, a sua conclusão fornecia insumos para o início de um novo ciclo da pesquisa para alcançar o próximo objetivo. Ademais, para a condução dos quatro ciclos desta pesquisa foi adotada uma abordagem híbrida de validação mesclando diferentes métodos e procedimentos.

Os objetivos podem ser separados e abstraídos de forma relativamente independente. Porém, a sua união simboliza o esforço desta investigação sobre a caracterização das *microtasks*. Por isso, a condução deste trabalho no sentido mais pleno representa uma investigação sobre o uso, as características, os contrastes e a complexidade das *microtasks* no desenvolvimento de software CS.

1.3 ORGANIZAÇÃO DO TEXTO

Em união, os próximos capítulos deste trabalho buscam investigar e contribuir com a literatura sobre *microtasks* focalizando os objetivos anteriormente apresentados. E para isso, o restante do conteúdo encontra-se organizado com a seguinte estrutura:

- O capítulo 2 sintetiza uma revisão sobre os temas abordados. Com isso, o conteúdo foi organizado por meio da fundamentação sobre o CS e *microtasks*, a condução do mapeamento sistemático (MS), e os trabalhos sinérgicos ao objetivo desta pesquisa.
- O capítulo 3 busca posicionar o leitor junto aos métodos e procedimentos empregados para conduzir a investigação deste trabalho. Com isso, são descritos os protocolos e as operacionalizações que foram empregadas.

- O capítulo 4 descreve o conjunto de resultados obtidos com a execução deste trabalho. Com isso, o capítulo encontra-se estruturado em porções dedicadas ao uso, as características, o contraste e a complexidade de uma *microtask*.
- O capítulo 5 conclui este trabalho apresentando uma síntese geral das contribuições realizadas. O capítulo também descreve o conjunto de limitações, lacunas e possíveis caminhos futuros.

Por fim, quando possível, deve-se destacar que foi utilizada a ferramenta *Wayback Machine*¹ do *Internet Archive* para referenciar endereços eletrônicos. Tal ferramenta possibilita a apresentação da cópia de uma página salva em seus servidores caso o endereço referenciado torne-se inválido.

¹Disponível em: <https://web.archive.org/> acesso em 14 nov. 2017

2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta a fundamentação teórica dos assuntos delineados neste trabalho. Dessa forma, a Seção 2.1 apresenta uma revisão dos aspectos relacionados ao CS, sua aplicação no desenvolvimento de software, e uma sumarização sobre as *microtasks*. A Seção 2.2 introduz o protocolo, análises e resultados do MS conduzido. A Seção 2.3 investiga um conjunto de trabalhos que fornecem resultados sinérgicos ao desta pesquisa. Finalmente, a Seção 2.4 sintetiza as considerações finais do capítulo.

2.1 FUNDAMENTAÇÃO TEÓRICA

O termo CS foi originalmente utilizado¹ por Howe (2006b) para definir um novo modelo de negócios que diversas empresas estavam aderindo no desenvolvimento de seus produtos. Em sua investigação, Howe (2006b) percebeu que as empresas estavam utilizando mecanismos para criar uma rede de trabalhadores anônimos para desenvolverem atividades preestabelecidas e receberem uma retribuição menor que empregados tradicionais.

Apesar do termo CS ter sido utilizado pela primeira vez no ano de 2006, a literatura apresenta diversos exemplos de práticas similares executadas em anos anteriores. Um dos exemplos mais bem estabelecidos refere-se à “questão da longitude” ou “problema da longitude” que estendeu-se entre o final do ano 1400 até o início do ano 1700. De acordo com Burton e Nicholas (2017) e Stock (2014), o problema era derivado de um desafio matemático de navegação para calcular a posição das embarcações no oceano. Sem a capacidade de determinar coordenadas precisas, as viagens tornavam-se perigosas e desastrosas, até que o parlamento inglês resolveu intervir e criou um substancial prêmio monetário para qualquer cidadão que resolvesse o enigma. No ano de 1714 um relojoeiro inglês apresentou uma solução criando o primeiro cronômetro marinho. Ou seja, um problema interno e restrito a navegação marinha foi terceirizado para um conjunto amplo e anônimo de pessoas, e a solução final foi proposta por um trabalhador que tinha experiência no conserto de relógios.

¹Howe publicou um artigo na revista WIRED utilizando o termo *crowdsourcing*, e logo após, realizou uma postagem em seu *blog* pessoal dividindo a criação do termo com Mark Robinson (HOWE, 2006a)

Mais de 100 anos após a solução para o desafio de longitude, uma prática muito similar ao CS foi adotada para a primeira publicação do Dicionário Oxford de Língua Inglesa (RAYMOND; TOMPA, 1987). Sem premiação, cerca de 800 voluntários catalogaram palavras para criar o primeiro fascículo do dicionário. Dessa forma, uma multidão ampla e anônima contribuiu para a publicação do dicionário executando pequenas porções de trabalho.

Apesar dos exemplos referentes ao “desafio da longitude” e a publicação da primeira versão do dicionário Oxford, existem diferenças entre essas práticas e ao CS. Como dito anteriormente, de acordo com Howe (2006b), o CS terceiriza o trabalho para uma multidão ampla e anônima, entretanto, a multidão recebe uma pequena retribuição. No primeiro exemplo, o parlamento inglês ofereceu uma grande quantia monetária ao vencedor, enquanto no segundo exemplo a participação foi voluntária. Em vista desse cenário, surge uma necessidade em compreender o real significado do CS devido ao alto número de exemplos, definições e percepções sobre CS na literatura (HOSSEINI et al., 2014). Para tratar essa questão, este estudo adotou a definição original dada por Howe (2006b) e complementada por Howe (2006a), em que o CS pode ser resumido como o ato de uma companhia ou instituição terceirizar uma função performada por empregados para uma rede anônima de pessoas na forma de chamada aberta. Sendo que o trabalho não é gratuito, mas custa menos que empregados tradicionais. Para facilitar a compreensão do CS, a Figura 1 ilustra o funcionamento do CS.

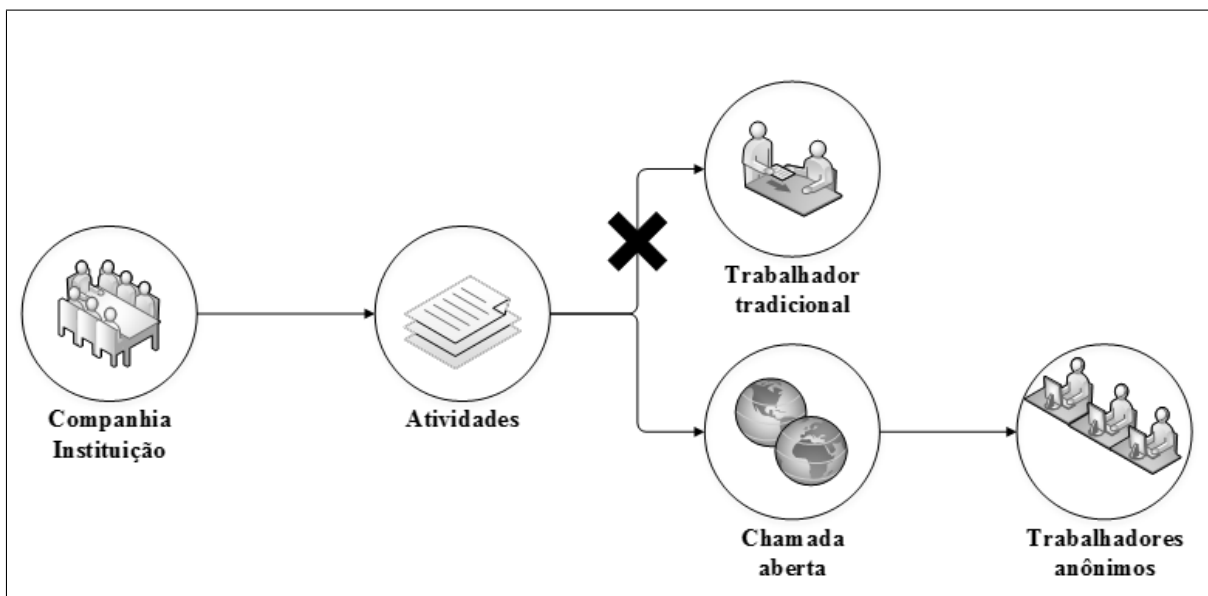


Figura 1: O funcionamento do *crowdsourcing*

Fonte: Autoria própria

Rapidamente o CS se alastrou por áreas do conhecimento humano, revelando resultados satisfatórios em diversas aplicações. Uma das áreas emergentes atualmente refere-se ao CS

na Engenharia de Software, que encontra grande apoio da academia e da indústria de desenvolvimento de software (DWARAKANATH et al., 2015).

Segundo Mao et al. (2017) um dos primeiros estudos a abordarem a aplicação do CS na Engenharia de Software ocorreu no ano de 2008. Desde então, vem ocorrendo um crescimento cumulativo de publicações focando no uso e na aplicação do CS em domínios diversos. Dessa forma, se faz necessário classificar como o CS torna-se desenvolvimento de software CS.

2.1.1 DESENVOLVIMENTO DE SOFTWARE *CROWDSOURCING*

A literatura demonstra heterogeneidade quando se busca ordenar o desenvolvimento de software CS perante outros modelos de desenvolvimento. Como foi discutido no início deste trabalho, as diferentes visões dos autores colidem entre si. Naik (2016) buscou sistematizar o CS como um modelo paralelo ao desenvolvimento distribuído de software (DDS) e ao *open source*. Assim, todos os modelos possuem, em grande parte, os mesmos desafios. Ao mesmo tempo, Li et al. (2015) definiram o desenvolvimento de software CS apenas como uma generalização do *open source*. Logo, os desafios do CS seriam herdados do *open source* e maximizados devido a generalização. Em contrapartida, LaToza e Hoek (2016) contrariaram essa argumentação e definiram que o *open source* que foi generalizado a partir do CS. Todas essas argumentações apresentadas são ilustradas na Figura 2.

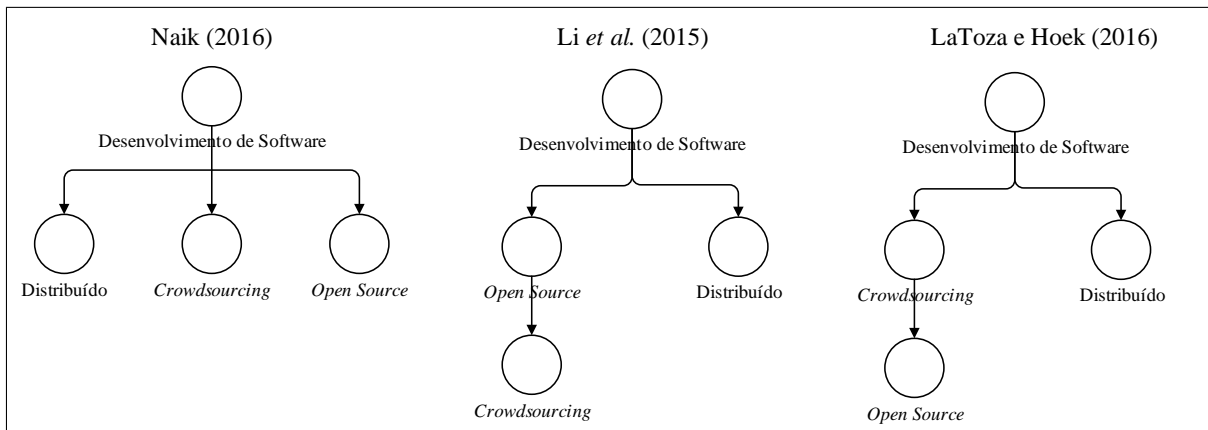


Figura 2: Os modelos de desenvolvimento de software na visão de diferentes autores

Fonte: Adaptado de Naik (2016), Li et al. (2015) e LaToza e Hoek (2016)

Em síntese, o CS representa uma tendência no desenvolvimento de software. Contudo, os autores divergem com suas opiniões em busca de classificar o uso do CS perante outros modelos de desenvolvimento. Apesar disso, existem alguns conceitos que podem ser abstraídos:

- O DDS é uma abordagem em que uma empresa ou organização divide um projeto, ou uma parte dele, e envia sua execução para uma entidade externa (NAIK, 2016). De acordo com Li et al. (2015), o desenvolvimento de software CS herdou como principal característica do DDS a dispersão global entre os *stakeholders* e os desafios que essa dispersão pode causar (assincronismo, barreiras linguísticas, culturais, etc). Os autores Li et al. (2015) também apresentaram que no DDS a coordenação entre os *stakeholders* ocorre por meio de ferramentas; a abertura ao projeto é limitada entre as partes envolvidas e existe um conjunto de contratos físicos que estabelecem prazos, custos e entregas. No desenvolvimento de software CS, a coordenação de um projeto é praticamente inexistente devido ao alto número de participantes anônimos; o contrato de trabalho é unicamente online, não havendo multas; e o custo das *microtasks* é definido apenas pelo *crowdsourcer*.
- No desenvolvimento *open source* o projeto é normalmente criado por um esforço colaborativo em que todos os participantes refinam o sistema e sua programação, compartilhando as mudanças com a comunidade (NAIK, 2016). O *open source* apresenta muita similaridade com o desenvolvimento de software CS, como a influência da dispersão global; a baixa coordenação do projeto; e a abertura a comunidade anônima (LI et al., 2015). Entretanto, enquanto no CS o trabalho é classificado como uma atividade com custo monetário, no *open source* o trabalho é voluntário e os participantes colaboram por altruísmo. Ademais, no *open source* toda a comunidade tem direito de utilizar, personalizar e distribuir um projeto, diferente do desenvolvimento de software CS.

Deve-se destacar que a literatura não demonstra com clareza como o desenvolvimento de software CS pode ser definido. Essa questão foi previamente resumida por Mao et al. (2017), e segundo os autores, existem diferentes percepções que mesclam definições próprias, a definição original dada por Howe (2006b), entre outras fontes. Similarmente, isso também ocorre com as *microtasks*.

2.1.2 MICROTASKS

Tendo em vista que a literatura sobre o desenvolvimento de software CS ainda é permeada de contradições e lacunas de definição, esse cenário torna-se mais intenso quando se põem em perspectiva as *microtasks*. Como foi brevemente resumido na Introdução deste trabalho, atualmente, existe uma escassez de estudos focados na temática de *microtasks* e diversos termos são adotados para representá-las. Bem, considerando esse panorama, optou-se por abstrair os principais conceitos que são amplamente difundidos e conhecidos sobre as *microtasks*. Em termos mais objetivos, como foi delineado por LaToza et al. (2014), pode-se dizer que as

microtasks descendem de um conceito focado na minimização do trabalho, representando um processo de separar grandes projetos em atividades pequenas, relativamente independentes, e que podem ser distribuídas. Todavia, diversas questões emergem sobre esse ponto de vista. Ainda não existe um consenso sobre qual a carga de complexidade, tamanho da porção de trabalho ou tempo necessário para se executar uma *microtask*. Um argumento defendido por diferentes autores reflete que uma *microtask* ocorre somente quando o esforço necessário para a sua execução representa a menor peça de trabalho possível (KITTUR et al., 2011). Porém, reside um risco grande em tal visão, principalmente pela dificuldade de atomizar determinados tipos de atividades.

Apesar da dificuldade em definir o que de fato são as *microtasks*, diversos setores estão adotando e se beneficiando da utilização dos seus conceitos de atomização (CONSUELO, 2016). Um dos principais meios responsáveis por potencializar o conceito de execução de *microtasks* foi a Amazon Mechanical Turk², em que são encontradas *microtasks* como a tradução de documentos, classificação de fotos e vídeos, entre outros. Assim, para o seu funcionamento, diversos trabalhadores *online* selecionam, executam e submetem soluções diversificadas para as *microtasks* disponíveis na plataforma. Com isso são evidenciados diferentes benefícios, como lembraram LaToza e Hoek (2016), como o acesso à soluções rápidas, com custo reduzido e alto grau de inovação.

Devido a portabilidade de uso, atualmente, as *microtasks* estão sendo adotadas no desenvolvimento de software CS para viabilizar a execução de tarefas em diferentes etapas do ciclo de vida de um produto ou projeto (STOL et al., 2017). Todavia, a sua utilização ainda representa um conceito muito complexo de ser definido e classificado. Esse panorama é complexado, principalmente, pelas diversas etapas que um projeto de software possui, pela indefinição sobre a granularidade das *microtasks* e a falta de mecanismos de gerenciamento ou controle (GADI-RAJU et al., 2017). Assim, diversos sintomas emergem dentro da literatura, como a contradição entre autores. Nesse sentido, pode-se destacar que para LaToza et al. (2015) as *microtasks* receberam tal nomenclatura por serem atividades que exigem pouco esforço e serem executadas rapidamente pelo *crowd*. Entretanto, Li et al. (2016) alertaram que cada projeto de software CS possui um conjunto de particularidades que limita uma generalização de modo a concluir que toda *microtask* é de fato simples e de rápida execução. Finalmente, para Zanatta et al. (2017), as *microtasks* podem ser complexas ou simples, dependendo da sua estruturação e objetivo final.

Apesar da definição formalizada por Zanatta et al. (2017) representar um visão am-

²Também conhecido como MKTurk ou somente AMT, representa uma plataforma online responsável de execução de *microtasks* que requerem o uso da inteligência humana. Disponível em: <https://www.mturk.com/> acesso em 22 fev. 2018

pla sobre *microtasks* e de certo modo estabelecer uma ligação entre a definição de LaToza et al. (2015) e Li et al. (2016), os autores introduzem o termo “*complex tasks*” para referir-se às *microtasks* com maior complexidade, entretanto não apresentam o que seria de fato uma “*complex tasks*”. E, similarmente, diversos autores adotam uma terminologia distinta para se referir às *microtasks* como a adoção do termo “*tasks*” por Winkler et al. (2017) ou “*CS tasks*” por Hosseini et al. (2014). Assim sendo, nenhum dos estudos estabeleceram uma terminologia comum ou uma forma de identificar qual a diferença dessas atividades, apesar de todos focarem no mesmo objeto de estudo.

Parte dos desafios referentes às *microtasks* se devem pela sua característica dinâmica, e de certa forma, sem fiscalização ou controle. Conforme investigado por Tranquillini et al. (2015), faltam recursos para classificar e compreender uma *microtask*. Assim, as “*microtask*” podem possuir diferentes configurações e níveis de complexidade, e ainda assim serem consideradas como “*microtask*”. Para facilitar a compreensão sobre o que é uma *microtask*, a Figura 3 simula seu processo de criação.

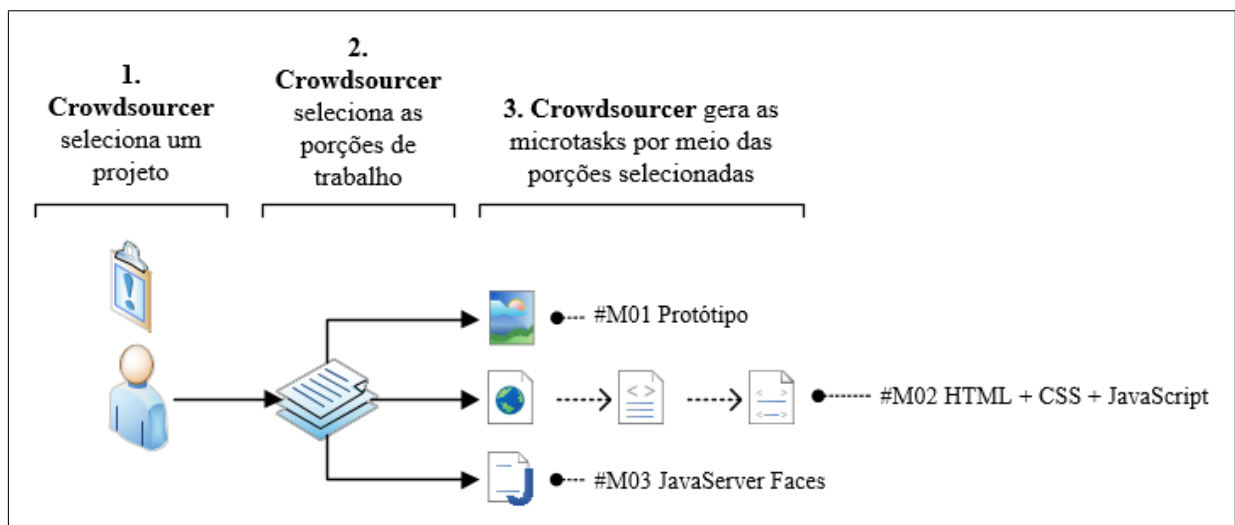


Figura 3: O processo de criação de *microtask*

Fonte: Autoria própria

No exemplo apresentado, o *crowdsourcer* selecionou um projeto e dividiu em diferentes porções. No caso ilustrado, a *microtask* #M01 foi direcionada a prototipação de uma página web, a *microtask* #M02 foi direcionada a estruturação da página web anteriormente prototipada e a *microtask* #M03 foi direcionada a inserção da tecnologia JavaServer Faces à página anteriormente estruturada. No exemplo apresentado as *microtasks* concentram diferentes níveis de expertises, de cargas de trabalho e de complexidade. Com isso, a *microtask* #M01 exigiu apenas a criação de um protótipo. A *microtask* #M02 foi baseada no protótipo anterior e no uso de

três tecnologias distintas. Finalmente, a *microtask* #M03 encerra a página web adicionando a tecnologia referente à linguagem de programação definida.

Apesar da Figura 3 ilustrar uma estruturação lógica de *microtasks*, o papel que decide a granularidade, a complexidade e o valor de uma *microtasks* é o *crowdsourcer*. Dessa forma, o *crowdsourcer* poderia, a exemplo, unificar as *microtasks* #M02 e #M03 em uma única. Em virtude disso, muitos desafios são originários como a definição da granularidade (WEIDEMA et al., 2016), do valor (MAO et al., 2013) e da complexidade de uma *microtask* (LATOZA; HOEK, 2016).

Devido as contradições da literatura e à natureza dinâmica das *microtasks* foi necessário estabelecer uma definição formal sobre o que representa uma *microtask*. E para isso, foi conduzido um MS sobre o tema, buscando compreender e analisar o atual estado da arte. Para a execução do MS foram investigados os domínios de aplicação, as contradições presentes na literatura e a divulgação dos trabalhos referentes às *microtasks* no desenvolvimento de software CS.

2.2 MAPEAMENTO SISTEMÁTICO

O processo de execução do MS deste estudo foi realizado conforme as especificações de Petersen et al. (2015) sobre a condução de estudos sistemáticos na área de Engenharia de Software. O processo completo utiliza um conjunto de etapas para produzir como produto final um MS da literatura. As etapas foram sintetizadas na seguinte ordem: elaboração das questões de pesquisa; procedimento de buscas a serem executadas; definições dos critérios de seleção e exclusão; e por fim, apresentação dos resultados obtidos. A execução dessas etapas são apresentadas a seguir.

2.2.1 QUESTÕES DE PESQUISA

O objetivo do MS está diretamente relacionado com a questão de pesquisa (QP)³ analisada. Nesse caso, o objetivo do MS deste estudo foi verificar os domínios, contradições e divulgações sobre as *microtasks* no desenvolvimento de software CS. Dessa forma, foram postuladas cinco QPs:

- **QP₁**: *Quais os domínios referentes às microtasks foram analisados?*
- **QP₂**: *Quais são as contradições encontradas na literatura referente as microtasks?*

³Quando necessário adotar o plural “questões de pesquisa” a sigla utilizada foi (QPs)

- **QP₃**: *Quais foram os métodos e procedimentos utilizados para a validação?*
- **QP₄**: *Quais foram os autores que publicaram estudos sobre microtasks?*
- **QP₅**: *Quais fontes científicas publicaram estudos sobre microtasks?*

2.2.2 BUSCAS

As buscas do MS foram performadas em dois estágios. No estágio inicial realizou-se a preparação: seleções das bases, seleção de termos e formulação da *string* de busca. No estágio final realizou-se a execução: buscas com base no título, no resumo, nos termos de indexação, na lista de referências e nas citações obtidas. Para auxiliar a condução do estágio inicial adotou-se o método PICO (do inglês *Population, Intervention, Comparison, and Outcomes* traduzido para *População, Intervenção, Comparação e Saída*), usado para identificar os termos chaves e formular a *string* de pesquisa (PETERSEN et al., 2015).

- **População**: A população refere-se ao domínio analisado. Nesse caso, a população adotada foram estudos pertencentes à área de Engenharia de Software com ênfase no desenvolvimento de software CS.
- **Intervenção**: A intervenção refere-se aos métodos, ferramentas, tecnologias ou procedimentos que deverão ser catalogados. Nesse contexto, a intervenção do MS foi identificar quais os métodos e procedimentos estão sendo adotados nas publicações sobre *microtasks*.
- **Comparação**: A comparação refere-se ao modo como as publicações do MS serão analisadas. Dessa forma, utilizando as QPs, a comparação ocorreu por meio dos domínios e das contradições presentes na literatura sobre as *microtasks* no desenvolvimento de software.
- **Saídas**: A saída é o produto final esperado com a conclusão do MS. Assim, a saída deste MS foi categorizar e classificar as publicações sobre *microtasks* fornecendo soluções para as QPs do MS.

Após a definição do método PICO, o processo de seleção das bases digitais foi simplificado. Assim, as bases foram selecionadas conforme o critério da “População” e do apoio fornecido pela entidade educacional da Universidade Tecnológica Federal do Paraná - Campus Cornélio Procópio, e são apresentadas na Tabela 1.

Tabela 1: Bases digitais selecionadas

Sigla	Nome	Endereço
IEEE Xplore	IEEE Xplore Digital Library	http://ieeexplore.ieee.org
ACM DL	ACM Digital Library	http://dl.acm.org/
ScienceDirect	Science Direct Library	http://www.sciencedirect.com/

Fonte: Autoria própria.

Para identificar os termos chaves foram adotados os critérios PICO e estudos que analisaram domínios semelhantes ao MS. O primeiro estudo refere-se ao trabalho “*A Survey of the Use of Crowdsourcing in Software Engineering*” realizado por Mao et al. (2017) para analisar a aplicação do CS na Engenharia de Software. Apesar do título revelar um *survey*, os autores conduziram um MS e enviaram formulários de pesquisa aos autores dos estudos selecionados para auxiliar a sistematização de dados. O segundo estudo refere-se ao trabalho “*The Four Pillars of Crowdsourcing: A Reference Model*” conduzido por Hosseini et al. (2014) que expôs o resultado de um MS referente ao CS e identificou as *microtasks* como uma das principais bases. Para auxiliar a montagem da *string* de pesquisa os termos selecionados foram divididos em três áreas: *microtasks*, *crowdsourcing* e *software*, como mostra a Tabela 2.

Tabela 2: Termos selecionados

	<i>Microtasks</i>	<i>Crowdsourcing</i>	<i>Software</i>
<i>microtask</i>	<i>microtasks</i>	<i>crowdsourcing</i>	
<i>“micro task”</i>	<i>“micro tasks”</i>	<i>“crowd sourcing”</i>	<i>software</i>
<i>task</i>	<i>tasks</i>	<i>“crowd sourced”</i>	
<i>activity</i>	<i>activities</i>	<i>crowd</i>	

Fonte: Autoria própria.

Com a seleção dos termos foi realizada a montagem da *string* de pesquisa. Desse modo, os termos foram unidos com os operadores lógicos “*OR*” e “*AND*”. O operador “*OR*” foi adotado para unir termos de uma mesma área enquanto o operador “*AND*” foi usado para unir os termos de áreas diferentes. A montagem concluída da *string* de pesquisa é encontrada na Tabela 3.

Tabela 3: Montagem final da *string* de pesquisa

<i>String de pesquisa</i>
<i>((microtask OR microtasks OR “micro task” OR “micro tasks” OR task OR tasks OR activity OR activities) AND (crowdsourcing OR “crowd sourcing” OR crowdsourced OR crowd)) AND (software)</i>

Fonte: Autoria própria.

A *string* de pesquisa foi aplicada nas bases digitais por meio de três consultas: com base no título, com base no resumo e com base nos termos de indexação. Cabe salientar que a *string* foi adaptada conforme a necessidade de cada base digital durante as consultas. Além

disso, também houve buscas por meio do uso da técnica conhecida como *snowballing*. O *snowballing* foi executado de dois modos: *backward* (com base na lista de referências) e *forward* (com base nas citações).

As consultas foram realizadas para selecionar as publicações iniciais. Após a inclusão de uma publicação, suas referências e citações obtidas eram analisadas. Caso uma nova publicação fosse adicionada durante o *snowballing* a iteração era repetida até não existir nenhuma nova publicação a ser adicionada no MS. Para otimizar o processo, uma base com o nome das publicações foi alimentada durante o procedimento de buscas. Caso alguma publicação fosse duplicada, a cópia era sumariamente removida do MS. O processo completo é apresentado na Figura 4.

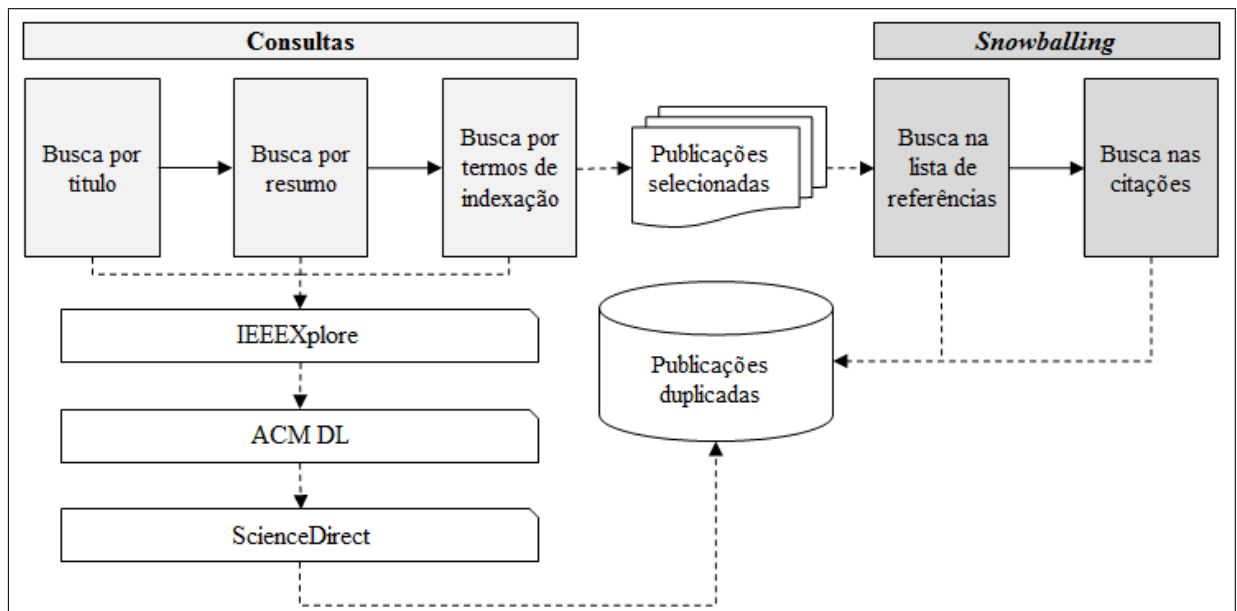


Figura 4: O processo de buscas do mapeamento sistemático

Fonte: Autoria própria

2.2.3 SELEÇÃO DE CRITÉRIOS

Os critérios foram adicionados ao MS para remover publicações que não estivessem diretamente alinhadas com as QPs e o objetivo do MS. Para a criação dos critérios adotou-se o método PICO em conjunto com os trabalhos utilizados para seleção de termos da *string* de pesquisa.

Foram utilizados dois critérios de inclusão relacionados à linguagem e ao tipo da publicação. Os critérios de remoção foram organizados em quatro categorias e pertencem ao

tipo da publicação e sua relação entre o objetivo do MS. Os critérios adotados são apresentados na Tabela 4.

Tabela 4: Critérios de inclusão e exclusão

Tipo	Sigla	Descrição
Inclusão	CI1	Publicações escritas em inglês e português
Inclusão	CI2	Capítulos de livros, artigos e relatórios técnicos
Exclusão	CE1	Tutorias ou apresentações
Exclusão	CE2	Não estão diretamente relacionado ao <i>crowdsourcing</i>
Exclusão	CE3	Não estão diretamente relacionado as <i>microtasks</i>
Exclusão	CE4	Não estão diretamente relacionado ao desenvolvimento de software

Fonte: Autoria própria.

2.2.4 RESULTADOS

O processo de execução do MS retornou um conjunto de publicações a serem analisadas. As consultas realizadas resultaram na seleção de 39 publicações. O *snowballing* foi executado durante 5 iterações, sempre adicionando ao menos uma nova publicação para o MS, entretanto na 6ª iteração nenhuma publicação foi adicionada e o processo foi finalizado. Ao final, por meio do *snowballing* foram adicionadas 12 publicações ao MS. O processo completo da execução das buscas e o total de estudos retornados por tipo de busca e iteração é apresentado na Figura 5.

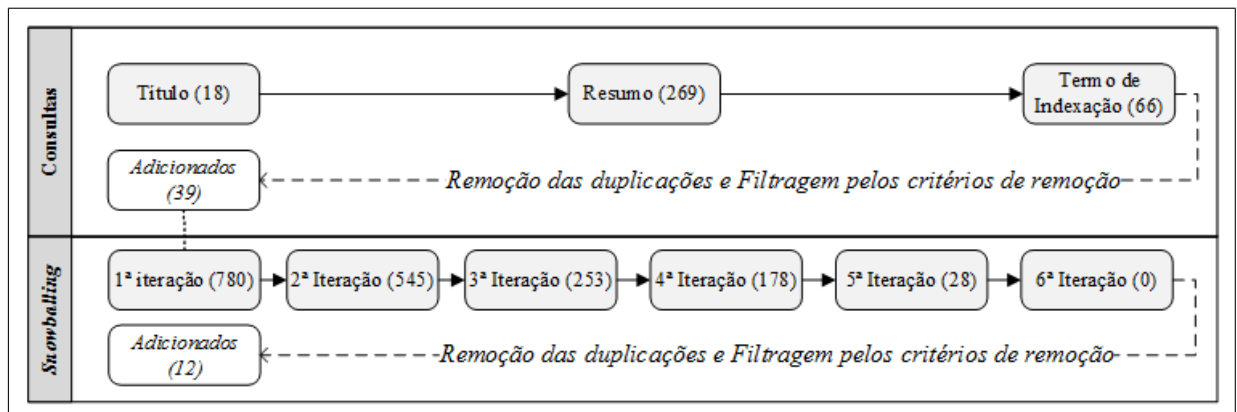


Figura 5: Estudos analisados

Fonte: Autoria própria

A primeira publicação indexada remete ao ano de 2009, os dois anos subsequentes não retornaram nenhum estudo. Entre 2012 e 2013 foram retornadas, respectivamente, 5 e 6 publicações. O crescimento exponencial das publicações ocorreu a partir do ano de 2014, em que foram analisadas 4 publicações, alcançando 10 publicações no ano de 2015 e

16 publicações no ano de 2016. Finalmente, em 2017 – ano de execução do MS – foram retornadas 9 publicações. A relação das publicações por ano é apresentada na Figura 6.

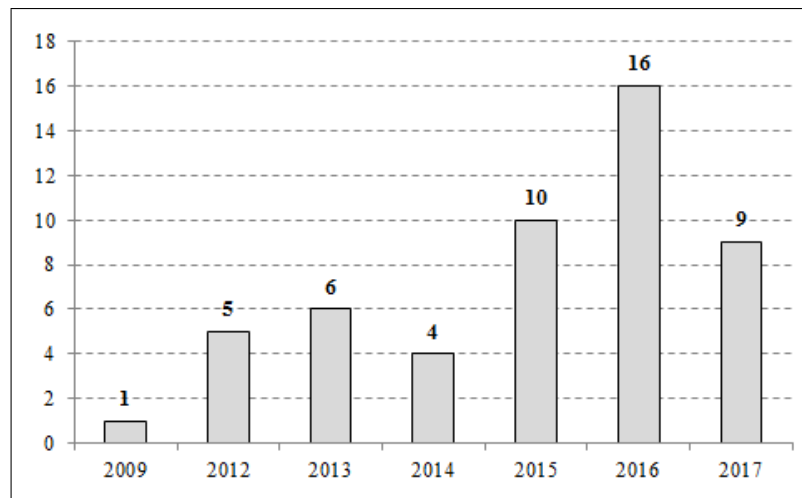


Figura 6: Relação de publicações por ano.

Fonte: Autoria própria.

Nota: O ano de 2017 registrou menos publicações que nos dois anos anteriores por ser o ano de execução do MS.

Após a análise inicial dos estudos, realizou-se a extração de dados como método de responder as QPs delineadas na subseção 2.2.1 (páginas 25 e 26) deste estudo.

QP₁: *Quais os domínios referentes às microtasks foram analisados?*

Para auxiliar a classificação dos domínios analisados foi utilizado o padrão proposto pelo *Institute of Electrical and Electronic Engineers* sobre desenvolvimento de software⁴. Em síntese, os estudos foram classificados em cinco domínios distintos, e cada domínio foi estruturado por um conjunto de subdomínios. A classificação final é apresentada a seguir.

1) Gerenciamento: diversos estudos focaram nas ações fundamentais ou no monitoramento de *microtasks*. Para classificá-los foi necessário adotar três subdomínios: *Inicialização*, ações essenciais e primárias; *Documentação*, recursos que auxiliam a execução; e *controle*, ações de monitoramento.

1.1) Iniciação: as ações primárias identificadas no MS se referem a interdependência e definição do valor de uma *microtask*. Nesse sentido, Stol e Fitzgerald (2014b) buscaram evidenciar como as *microtasks* são mais eficientes quando não existem interdependências. Enquanto que os autores Mao et al. (2013) apresentaram um modelo estatístico para realizar o cálculo do valor de uma *microtask*. No estudo de

⁴Disponível em <https://web.archive.org/web/20180112213104/http://ieeexplore.ieee.org/document/741936/> acesso em 18 dez. 2017

Afridi (2017) foi apresentada uma investigação para auxiliar os líderes a definirem o valor de uma *microtask*. No estudo de Wang et al. (2013) buscou-se sistematizar uma forma para definir o valor de uma *microtask* com base na qualidade do trabalho final, ajustando o valor de acordo com a contribuição de cada trabalhador.

1.2) Documentação: Zanatta et al. (2017) discutiram como a ausência da documentação das *microtasks* torna a sua execução complexa, e por vezes, inadequada.

1.3) Monitoramento/Controle: de acordo com Deus et al. (2017a) e Ali et al. (2016), a leitura, acompanhamento, atualização e remoção das *microtasks* representam um dos principais impactos no monitoramento do projeto. Os autores LaToza et al. (2015) implementaram um mecanismo de perguntas e respostas para a execução de *microtasks* com foco em analisar o seu monitoramento. Enquanto que o estudo conduzido por Hetmank (2013) revelou como a alocação da *microtasks* pode influenciar o monitoramento do projeto. Tajedin e Nevo (2014) propôs um novo arranjo para o CS, levando em consideração a utilização de um papel intermediário entre o *crowd* e as *microtasks*.

2) Pré-desenvolvimento: múltiplos estudos focaram nos conceitos que antecedem o desenvolvimento das *microtasks*. Nesse sentido, quatro subdomínios foram identificados, sendo eles: *Alocação*, aspectos que devem ser considerados antes, durante e após selecionar os participantes e a plataforma CS; *Refinamento*, atomização da *microtasks*; *Características*, relação das particularidades da *microtask*; e *Complexidade*, dificuldade de execução.

2.1) Alocação: na perspectiva do *crowd*, de acordo com Machado et al. (2016), Li et al. (2016), Saremi e Yang (2015a) e Cleland-Huang et al. (2012), a alocação deve ser realizada considerando a habilidade necessária, o tempo disponível, o valor final e a descrição de uma *microtask*. Enquanto que na perspectiva da plataforma CS, a alocação foi investigada pelo estudo de Dubey et al. (2016). Além disso, o estudo Bessai e Charoy (2016) propôs um *framework* para auxiliar a tarefa de alocação.

Diversos estudos buscaram analisar o comportamento dos participantes após a alocação das *microtasks*. Com isso, Saremi et al. (2017), Yang et al. (2016) e Yang e Saremi (2015) investigaram o comportamento que leva um participante selecionar determinada *microtask*. Enquanto que o estudo de Alelyani e Yang (2016) apenas analisou como o prazo, o valor e o tipo de das *microtasks* induzem um colaborador a selecioná-la.

Ademais, após a alocação, também houveram trabalhos que buscaram estimular a

participação do *crowd* por meio de recomendação. Os autores Mao et al. (2015) introduziram um *framework* para recomendar automaticamente desenvolvedores conforme cada *microtask*. O estudo de Zhu et al. (2015) discutiu como extrair critérios das *microtasks* e recomendá-los ao *crowd*. Os autores Zhao et al. (2015) apresentaram um modelo de recomendação com base nas competências, similaridades e capacidade dos participantes. O estudo de Leano et al. (2016) apresentou mecanismos para identificar as habilidades necessárias de participantes especialistas ou inexperientes para o desenvolvimento da *microtasks*. E o estudo de Morris et al. (2012) buscou explorar como a execução das *microtasks* podem estar relacionadas aos participantes.

2.2) *Refinamento*: os autores Weidema et al. (2016) apresentaram uma discussão sobre a exploração de *microtasks* e os potenciais benefícios e riscos de sua redução. O estudo de Stol e Fitzgerald (2014a) endereçou uma questão sobre como reduzir as *microtasks* de modo que o projeto não sofra nenhum problema. Os autores Schiller e Ernst (2012) automatizaram a rotina de refinamento de *microtasks* por meio de uma ferramenta de decomposição. O estudo de LaToza e Hoek (2015) apresentou uma agenda de pesquisa, focando em tendências de análises futuras, destacando o impacto do refinamento das *microtasks*.

2.3) *Características*: uma lista das características encontradas nas *microtasks* foi apresentada no estudo de Hosseini et al. (2014). Além disso, os autores apresentaram uma taxonomia sobre o CS.

2.4) *Complexidade*: a complexidade de uma *microtask* foi tratada apenas como agenda de pesquisa. O estudo de Vukovic (2009) apresentou possíveis tópicos de aprofundamento de *microtasks* focando no uso de *microtasks* simples e complexas. O estudo de LaToza e Hoek (2016) focou na eficiência de *microtasks* como uma área incipiente do ponto de vista de complexidade e decomposição.

3) Desenvolvimento: uma parte dos estudos analisados focou, efetivamente, na execução das *microtasks*. E para classificá-los, os subdomínios utilizados foram: *Análise*, relacionado aos requisitos; *Design*, direcionado aos protótipos; *Implementação*, execução por uma linguagem de programação; *Testes*, avaliação; e *Integração*, união das *microtasks*.

3.1) *Análise*: A aplicação de *microtasks* em requisitos de software CS foi analisada por Bhatia et al. (2016). Os autores uma abordagem híbrida para auxiliar a extração das políticas de privacidade presentes em cada requisito de um projeto de software com o uso de *microtasks* e da linguagem natural.

3.2) *Design*: O estudo de Zhao e Hoek (2015) descreveu aspectos que podem auxiliar na criação de *microtasks* durante a etapa de *design*.

3.3) *Implementação*: O estudo de LaToza et al. (2014) apresentou uma abordagem para construir um software aplicando pequenas porções de trabalhos nas *microtasks*. Similarmente, o estudo de LaToza et al. (2013) discutiu a etapa de implementação via *microtasks* no desenvolvimento de software CS.

3.4) *Testes*: Uma forma de otimizar a escrita e execução de testes utilizando conceitos da *gamification* e de *microtasks* foi discutida por Fraser (2017). Os estudos de Wang et al. (2016), Feng et al. (2015) e Feng et al. (2016) discutiram a aplicação das *microtasks* no teste de software, com ênfase em verificar se os participantes executaram corretamente as rotinas de testes. Enquanto que o estudo Winkler et al. (2017) apresentou um modelo para separar a inspeção de *microtasks* em diferentes porções. O estudo de Jiang e Matsubara (2012) propôs uma abordagem para melhorar a eficiência das *microtasks* com ênfase na correção e detecção de erros.

3.5) *Integração*: Tranquillini et al. (2015) propôs uma abordagem para realizar a integração das *microtasks*. Os autores se basearam na utilização de um processo próprio para projetar e depois integrar as *microtasks*.

4) Integral: alguns autores destacaram aspectos que estão relacionados durante toda a execução de uma *microtask*, porém são evidenciados após a sua conclusão⁵. Nesse sentido, somente um subdomínio foi identificado: *Avaliação*, que busca prever a qualidade de execução de uma *microtask*.

4.1) *Avaliação*: para garantir a qualidade de uma *microtask*, o estudo de Faisal (2017) propôs utilizar a base histórica da plataforma, investigando informações sobre as *microtasks* finalizadas. O estudo Allahbakhsh et al. (2013) apresentou um *framework* para definir as várias dimensões da qualidade do desenvolvimento de software CS e das *microtask*. Enquanto que Baba e Kashima (2013) propuseram um método para estimar a qualidade estatística de execução das *microtasks*.

5) Outros: os demais estudos que não foram classificados em nenhum agrupamento anterior foram organizados em dois subdomínios distintos: *Uso*, estudos que revelam como as *microtasks* estão sendo aplicadas; e *Ensino/Aprendizado*, estudos focados no uso de *microtasks* em ambientes acadêmicos.

⁵Deve-se destacar que teste e avaliação são conceitos distintos. O subdomínio de testes classificou os estudos focados na execução das *microtasks*, enquanto que o subdomínio de avaliação sistematizou os estudos que investigavam a análise após a execução das *microtasks*.

5.1) *Uso*: os estudos de Deus et al. (2017b) e Saremi e Yang (2015b) apresentaram análises sobre a aplicação de *microtasks* no desenvolvimento de software CS. Enquanto que o estudo de Mao et al. (2017) analisou a área de Engenharia de Software, demonstrando resultados sobre a aplicação de *microtasks* no desenvolvimento de software CS.

5.2) *Ensino/Aprendizado*: O foco do estudo L'Erario et al. (2017) foi apresentar uma comparação entre o ensino e aprendizado de CS e DDS. Nesse aspecto, os autores destacaram dados referentes à distinção entre as atividades DDS e *microtasks* para os discentes e docentes. Enquanto que o estudo de Deus et al. (2016) focou em aspectos sobre a simulação acadêmica do CS. Para isso, os autores demonstraram diversos aspectos necessários, e entre eles, a necessidade de utilização das *microtasks*.

Para auxiliar a sistematização das informações, a Figura 7 apresenta uma síntese dos domínios e subdomínios analisados das *microtasks*.

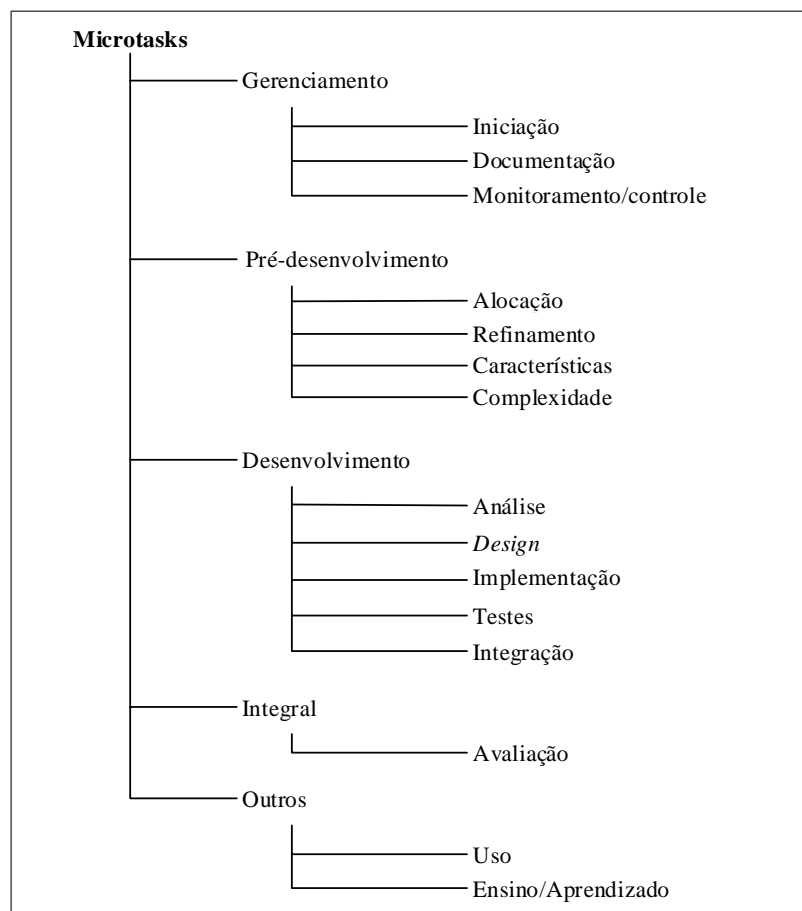


Figura 7: Domínios das *microtasks* reveladas pelo mapeamento sistemático

Fonte: Autoria própria

QP₂: Quais são as contradições encontradas na literatura referente as microtasks?

No que diz respeito às contradições detectadas na literatura foram identificados três tópicos sobre: i) simplicidade/complexidade, ii) natureza das *microtasks*, e iii) terminologia empregada.

- **Microtasks são simples ou são complexas?** Os estudos de Weidema et al. (2016), LaToza e Hoek (2015), Stol e Fitzgerald (2014a) e Schiller e Ernst (2012) apresentaram considerações sobre a atomização das *microtasks* e o fato disso refletir em atividades curtas, interdependentes, com rápido desenvolvimento, de fácil execução pelo *crowd*. Entretanto, na visão de Li et al. (2016), as *microtasks* não podem ser generalizadas para este cenário de simplicidade. Isso decorre devido ao caráter único de cada projeto e a percepção dos participantes na execução das *microtasks*.
- **Microtasks são atividades CS ou representam um novo domínio para atividades de software?** O estudo de Cleland-Huang et al. (2012) utilizou conceitos das *microtasks* serem executadas por grande grupos de pessoas e propôs que a sua utilização pode auxiliar outros setores da Engenharia de Software. Assim, seu uso não estaria restrito ao desenvolvimento de software CS. Entretanto, o estudo Hosseini et al. (2014) gerou uma taxonomia sobre o CS que classificou as *microtasks* como um dos pilares do CS. Ademais, Mao et al. (2013) demonstraram que as métricas tradicionais das atividades de software não possuem resultados satisfatórios nas *microtasks*. Dificultando assim, sua aplicação em outros setores da Engenharia de Software.
- **Microtasks, Tasks, Tasks CS ou Complex Tasks?** Essa é a principal questão que pode fomentar discussões e falhas no entendimento de uma *microtask*. Durante a análise do MS percebeu-se que alguns autores utilizaram o termo “*tasks*” (BESSAI; CHAROY, 2016; WINKLER et al., 2017; FENG et al., 2016). Enquanto os autores Hosseini et al. (2014) aplicaram o termo “*tasks CS*”. Os autores Mao et al. (2013) e Zanatta et al. (2017) adotaram dois termos distintos “*micro tasks*” e “*complex tasks*” para tratar, respectivamente, de atividades rápidas e com maior duração. Os demais estudos adotaram o termo utilizado neste trabalho “*microtasks*” ou alguma variação como “*micro-task*”.

QP₃: Quais foram os métodos e procedimentos utilizados para a validação?

Foram identificadas abordagens distintas para validação dos estudos analisados no MS. Das três abordagens mais usadas, duas pertencem aos modelos empíricos e uma ao modelo teórico. A condução de experimentos foi a abordagem mais utilizada, seguida pela execução

de estudo de caso, e em terceiro lugar, as publicações dedicadas aos modelos teóricos, ou seja, sem nenhum tipo de validação empírica. As publicações de opinião (autores convidados a descreverem sobre a área) e a condução de MS obtiveram o mesmo índice de validação. Por fim, se encontram as abordagens híbridas, que utilizaram MS + *survey* e estudo de caso + *survey*. A porcentagem dos métodos e procedimentos de validação são apresentados na Figura 8.

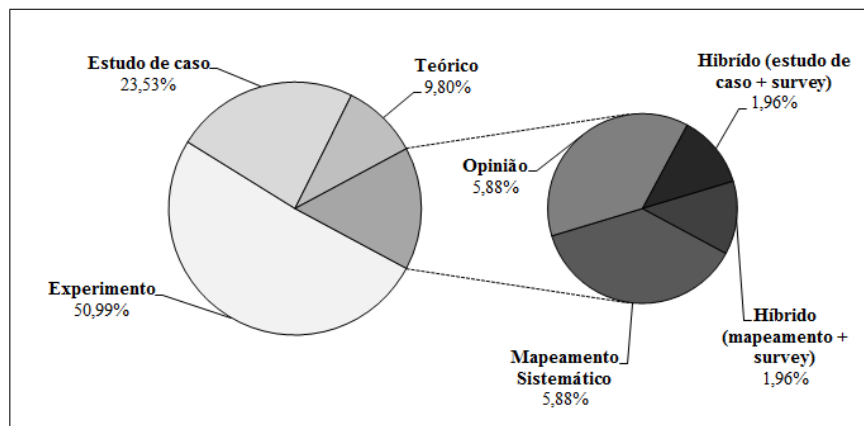


Figura 8: Métodos e procedimentos de validação

Fonte: Autoria própria

QP₄: *Quais foram os autores que publicaram estudos sobre microtasks?*

O autor que mais publicou estudos analisados neste MS sobre *microtasks* foi Y. Yang, pesquisador que possuiu o total de 8 artigos analisados. A. van der Hoek foi o segundo pesquisador mais analisado, com 7 artigos. Os autores A. L'Erario, T. D. LaToza e W. S de Deus obtiveram 4 publicações analisadas por este MS. Os demais autores tiveram 3 publicações ou menos. A relação completa de autores analisados é encontrada na Tabela 5.

QP₅: *Quais fontes científicas publicaram estudos sobre microtasks?*

Em relação às fontes analisadas, foram identificadas publicações em eventos e periódicos científicos. Os eventos foram organizados como conferências, *workshops* ou simpósios. Os periódicos foram organizados em revistas ou jornais científicos.

Os eventos foram as fontes que concentraram as três maiores porcentagens de publicação enquanto que os periódicos ocuparam a quarta e a quinta posição. O índice de porcentagem por fonte publicadora é apresentada na Tabela 6, enquanto que a dispersão gráfica das fontes publicadoras são encontradas na Figura 9. No Apêndice A deste estudo encontra-se a lista com o nome a sigla das fontes publicadoras.

Com a condução do MS foram revelados estudos presentes na literatura com objetivos similares aos desta pesquisa. Dessa forma, os trabalhos referentes ao uso, características,

Tabela 5: Relação de autores analisados

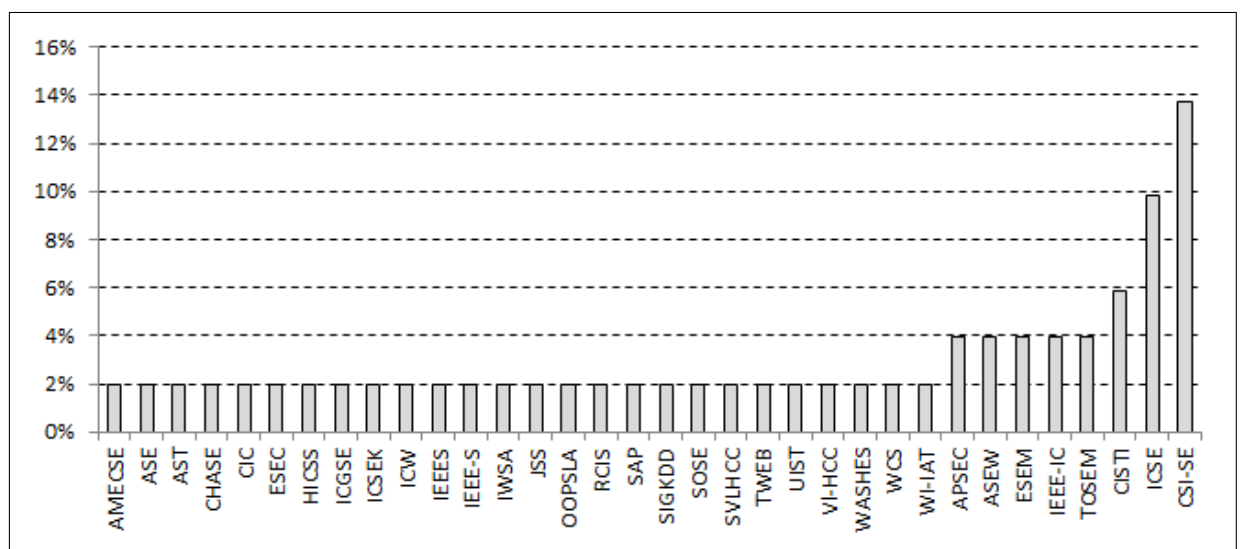
Quantia de Publicação	Autores
1 Publicação	A. Czauderna, A. D. Lecce, A. Dubey, A. Dwarakanath, A. Ignjatovic, A. Kass, A. L. Zanatta, A. Sarma, B. Benatallah, B. Xu, C. López, C. M. Adriano, D. Messinger, D. Nevo, D. Poshyvanyk, D. Winkler, E. Bertino, E. Carmel, E. Keenan, E. M. Gerber, E. Moritz, E. R.Q. Weidema, E. S. Nasr, F. Casati, F. Charoy, F. Daniel, F. Hu, F. Meneguzzi, F. Ricci, F. Spanghero, G. Carneiro, G. Fraser, G. Leach, G. Virdi, H. G. Afridi, H. Jiang, H. Kashima, H. R. Motahari-Nezhad, H. Tajedin, H. Zhong, I. Steinmacher, J. Cleland-Huang, J. H. Hayes, J. Taylor, J. Wang, J. Zhu, K. Abhinav, K. Bessai, K. Phalp, L. Capra, L. Hetmank, L. Machado, M. Allahbakhsh, M. D. Ernst, M. Dontcheva, M. Gethers, M. Gheith, M. Hosseini, M. I. Faisal, M. Kalinowski, M. Li, M. R. Karim, M. S. Kuriakose, M. Sabou, M. Vukovic, M. Zhao, N. Li, P. Kucherbaev, Q. Cui, R. Ali, R. H. C. Palácios, R. Leano, R. M. Barros, R. R. Morris, S. Biffl, S. Dustdar, S. Matsubara, S. Nayebaziz, S. Petrovic, S. Taneja, S. Tranquillini, S. Wang, S. Zhao, T. Alelyani, T. Ali, T. LaToza, T. W. Schiller, W. Godoy, W. Li, W. Mo, Y. Baba, Y. Chen, Y. Shin, Z. Wang.
2 Publicações	B. Fitzgerald, C. Fang, C. R. B. de Souza, F. Schaub, G. Ruhe, J. A. Jones, J. Bhatia, K. Stol, L. S. Machado, Q. Wang, R. Prikładnicki, R. Saremi, T. D. Breaux, Y. Feng, Y. Jia, Z. Chen.
3 Publicações	B. Shen, J. A. Fabri, K. Mao, M. Harman, R. L. Saremi, W. B. Towne.
4 Publicações	A. L'Erario, T. D. LaToza, W. S. de Deus.
7 Publicações	A. van der Hoek.
8 Publicações	Y. Yang.

Fonte: Autoria própria.

Tabela 6: Relação de autores analisados

Tipo da fonte	Fonte	Frequência (%)
Evento	Conferências	49%
Evento	Workshops	22%
Evento	Simpósios	15%
Periódico	Jornais	10%
Periódico	Revistas	4%

Fonte: Autoria própria.

**Figura 9: Fontes publicadoras**

Fonte: Autoria própria

comparações ou complexidade de *microtasks* foram analisados com o intuito de verificar seus resultados e limitações. Assim, a seguir ocorre a comparação dos trabalhos similares.

2.3 TRABALHOS RELACIONADOS

Os trabalhos similares identificados no MS são apresentados na Tabela 7. Os dados foram sintetizados conforme o subdomínio da *microtask*, a identificação dos autores e as limitações de cada abordagem.

Tabela 7: Comparação entre os trabalhos similares

Subdomínio	Autores	Limitações
<i>Uso</i>	Saremi e Yang (2015b)	Conjunto de dados pequeno, curto período de análise e restrição dos tipos de <i>microtasks</i>
	Mao et al. (2017)	Ausência de termos e processo de busca
<i>Características</i>	Hosseini et al. (2014)	Ampliar a lista de característica para acompanhar a rápida evolução da área
<i>Complexidade</i>	Vukovic (2009)	Não houve comprovação empírica e ano de publicação defasado
	LaToza e Hoek (2016)	Não houve comprovação empírica
<i>Contraste</i>	Não foi detectado nenhum trabalho similar	

Fonte: Autoria própria.

Tendo em vista o uso de *microtasks* foram detectados dois trabalhos. O primeiro trabalho foi desenvolvido por Saremi e Yang (2015b) a partir de um conjunto de dados de uma plataforma de software CS. Os autores reportaram resultados sobre as áreas de aplicação, a falta de padronização e a capacidade de paralelismo das *microtasks*. Entretanto, os resultados gerados dificilmente podem ser generalizados devido a limitação do conjunto de dados (apenas 4908 *microtasks*), período de análise (*microtasks* cadastradas entre janeiro/2014 à fevereiro/2014) e tipos de *microtasks* (apenas 6). O segundo trabalho foi desenvolvido por Mao et al. (2017) e demonstrou a aplicação do CS na Engenharia de Software. Os autores utilizaram uma abordagem híbrida com MS e *survey* para analisar os domínios, as *microtasks*, as aplicações, as plataformas e os *stakeholders* do CS. Contudo, durante a construção da *string* de busca, os autores desprezaram termos importantes, como “*crowd sourcing*” e “*software*”. Assim, importantes trabalhos podem ter sido desconsiderados. Além disso, os autores apenas executaram a busca *snowballing* nas referências dos estudos analisados, descartando as citações obtidas.

No que diz respeito as análises sobre as características das *microtasks* Hosseini et al. (2014) apresentaram um modelo de referência sobre CS em que classificou as *microtasks* com base em oito características: i) forma de execução; ii) forma de terceirização; iii) modularidade; iv) complexidade; v) solvabilidade; vi) automação; vii) orientação e viii) tipo de contribuição.

Apesar das características apresentadas, o estudo se restringe a uma sistematização do conhecimento. Os próprios autores apresentaram em suas considerações que a lista de características deve ser ampliada por meio de novas pesquisas para acompanhar o rápido desenvolvimento da área.

Sob o ponto de vista da complexidade das *microtasks*, foram detectados apenas trabalhos focados na necessidade de gerar mecanismos para esse fim. Assim sendo, esses trabalhos representaram agendas de pesquisa para direcionar análises em setores carentes da literatura. De acordo com as considerações expostas por Vukovic (2009), faltavam mecanismos para definir e suportar o trabalho nas atividades simples e complexas. Todavia, o ano de publicação do trabalho estava defasado. Porém, percebe-se que a limitação exposta ainda representa uma lacuna da literatura, pois segundo LaToza e Hoek (2016), existe uma necessidade em focar novas análises para aferir e investigar a complexidade das *microtasks*. Todavia, essas considerações foram baseadas apenas no conhecimento, experiência e opinião dos autores sobre as *microtasks*.

Finalmente, não houveram trabalhos recuperados no MS que realizassem comparações dos contrastes *microtasks* CS com as atividades de software. Ademais, como foi discutido na **QP₂** (página 35 deste trabalho) foi detectada uma contradição na literatura do mesmo sentido. Sendo assim, acredita-se que esses sejam fortes sinalizadores de que a área apresenta grande incipiência de análises, fornecendo um cenário fecundo para as contribuições realizadas por este trabalho.

2.4 CONSIDERAÇÕES FINAIS

A escassez de estudos em torno do tema “*microtask*” revelou as diversas lacunas presentes na literatura. De fato, os diferentes sintomas percebidos – como as contradições entre os autores ou a falta de uma terminologia comum – destacam a necessidade da condução de novos estudos analisando o tema. Logo, emerge a necessidade e relevância deste trabalho.

Para evitar discussões pouco fecundas em volta do assunto “*microtasks*” este trabalho buscou por meios próprios compreendê-las e defini-las. Assim, com a experiência fornecida pelo MS, pode-se considerar que as *microtasks* surgiram como um paradigma focado na minimização do trabalho, evidenciando entregas curtas e rápidas, encontrando aplicação em diversos setores da sociedade. Todavia, garantir a granularidade sempre reduzida de tais *microtasks* ainda representa uma tarefa imprecisa e dispendiosa e tema de muita controvérsia. Ademais, além dessa descrição, cabe destacar que as *microtasks* investigadas por este trabalho dizem respeito as atividades executadas no desenvolvimento de software CS.

Além disso, os resultados fornecidos pelo MS demonstraram diversos achados. Inicialmente, é adequado evidenciar a identificação de uma agenda de pesquisa baseada na complexidade e granularidade sobre *microtasks*. Tais tópicos representam lacunas e podem guiar a condução de novos trabalhos analisando os assuntos. Existem também às contradições e a ausência de padronização de termos sobre *microtasks*. Nesse sentido, existe a possibilidade da disseminação de novos estudos para reduzir tais conflitos.

Por fim, deve-se salientar que os resultados apresentados neste capítulo foram obtidos exclusivamente por meio da condução do MS. Porém, o MS também foi usado de maneira conjunta com outros métodos e procedimentos de pesquisa que são apresentados no capítulo 3 deste trabalho.

3 MÉTODOS E PROCEDIMENTOS

Para conduzir esta pesquisa foi adotada uma abordagem híbrida de validação. Dessa forma, o principal objetivo deste capítulo é apresentar os métodos e procedimentos adotados. Para isso, o capítulo encontra-se organizado em três seções: a Seção 3.1 apresenta a condução do estudo de caso. A Seção 3.2 demonstra a aplicação da experimentação controlada. Finalmente, a Seção 3.3 sintetiza as considerações finais.

Este capítulo é uma síntese dos métodos e procedimentos adotados nos artigos publicados por Deus et al. (2016) no evento “*I Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software*”; Deus et al. (2017b) no evento “*12th Iberian Conference on Information Systems and Technologies*”; Deus et al. (2017d) no evento “*47th Frontiers in Education*” e Deus et al. (2017c) e no periódico “*Journal on Advances in Theoretical and Applied Informatics*”.

3.1 ESTUDO DE CASO

O estudo de caso é um método de pesquisa empírica que busca coletar informações detalhadas sobre um objeto de estudo por meio de seu próprio contexto (WOHLIN et al., 2000). A execução de um estudo de caso ocorre em diversas situações, todavia como alerta Yin (2015), o estudo de caso é comumente utilizado em análises de ciências sociais, políticas, organizacionais e situações correlacionadas. Isso ocorre pela necessidade de compreender os fenômenos nesses setores da sociedade.

Por conseguinte, é considerável comparar a necessidade e a motivação de aplicar estudos de caso na Engenharia de Software. De acordo com Runeson e Høst (2008), a aplicabilidade de estudos de casos surge devido à necessidade de investigar como o desenvolvimento, a operação e a manutenção de software são conduzidos por diferentes indivíduos, grupos e organizações.

O ponto de partida para conduzir um estudo de caso é a observação e compreensão do

fenômeno a ser analisado. Como foi destacado por Yin (2015) existem mecanismos que podem auxiliar essa tarefa, como a seleção da estratégia e o planejamento da pesquisa. Contudo, como alertado pelo próprio autor, não existe nenhum catálogo abrangente que possa apoiar a execução de todos, ou da maioria, dos estudos de caso devido as particularidades de cada pesquisa. Em vista desse cenário, Wohlin et al. (2000), Runeson e Høst (2008) destacam as importantes etapas a serem realizadas para conduzir um estudo de caso na Engenharia de Software, como mostra a Tabela 8.

Tabela 8: *Template* resumido para conduzir um estudo de caso na Engenharia de Software

Etapa	Descrição
1ª Etapa	<p>1. Design</p> <p><i>1.1 Definição dos objetivos:</i> deve ser exposto os principais aspectos que caracterizam o estudo de caso. Nesse sentido, pode-se adotar os seguintes elementos propostos por Robson (2002) sobre i) o objetivo, ii) o caso, iii) a teoria, iv) as questões analisadas, v) os métodos e vi) a estratégia.</p> <p><i>1.2 Protocolo do estudo de caso:</i> estruturar um documento para auxiliar a condução do estudo de caso e as possíveis tomadas de decisão durante a execução do estudo de caso.</p>
2ª Etapa	<p>2. Preparação e Coleta</p> <p><i>2.1 Nível da coleta de dados:</i> de acordo com Lethbridge et al. (2005) os dados podem ser coletados em três níveis. No primeiro nível ocorre contato direto e a coleta em tempo real; no segundo nível o contato é indireto e a coleta, geralmente, torna-se atemporal; no terceiro nível ocorre uma análise independente em dados disponíveis.</p> <p><i>2.2 Coleta de dados:</i> após a definição do nível de coleta de dados ocorre, efetivamente, o processo de coletar dados do estudo de caso.</p>
3ª Etapa	<p>3. Análise</p> <p><i>3.1 Análise quantitativa:</i> ocorre quando o estudo de caso deseja derivar conclusões dos dados.</p> <p><i>3.2 Análise qualitativa:</i> ocorre quando o estudo de caso utiliza análises como estatística descritiva, análise de correlação, desenvolvimento de modelos e teste de hipótese.</p> <p><i>3.3 Ameaças à validade:</i> demonstrar a incerteza sobre os resultados captados pela condução do estudo de caso.</p>

Fonte: Adaptado de Wohlin et al. (2000), Runeson e Høst (2008).

Cabe destacar que de acordo Wohlin et al. (2000), Runeson e Høst (2008) grande parte das etapas são essenciais para a execução de um estudo de caso. Todavia, abre-se uma única exceção para as etapas de análise, respectivamente as etapas 3.1 e 3.2, que somente são executadas conforme a natureza de cada estudo de caso.

O estudo de caso foi selecionado para iniciar esta pesquisa sobre a caracterização das *microtasks* no desenvolvimento de software CS. A seguir, tem-se a execução das etapas descritas na Tabela 8.

3.1.1 DESIGN

Inicialmente foi realizada a definição dos objetivos do estudo de caso. Conforme sugerido por Robson (2002), os elementos foram identificados e tabulados. Ademais, o protocolo

do estudo de caso também foi sintetizado e seus principais elementos foram identificados. A apresentação do *design* do estudo de caso é encontrada na Tabela 9.

Tabela 9: Design do estudo de caso

Etapa	Conteúdo
1.1 Definição dos objetivos	<p><i>O objetivo</i> O objetivo do estudo de caso foi identificar como as <i>microtasks</i> estão sendo utilizadas no desenvolvimento de software CS.</p>
	<p><i>O caso</i> O caso estudado trata-se de seleção de conjunto formado por <i>microtasks</i> reais extraídas de uma plataforma de desenvolvimento de software CS.</p>
	<p><i>A teoria</i> Foram adotados os trabalhos de Suganthy e Chithralekha (2016), Naik (2016), Hosseini et al. (2014) e LaToza e Hoek (2015) que revelam diferentes lacunas sobre o uso de <i>microtasks</i> no desenvolvimento de software CS.</p>
	<p><i>As questões analisadas</i> De acordo com os autores anteriormente consultados, existem diferentes vertentes a serem exploradas sobre o uso de <i>microtasks</i> para o desenvolvimento de software CS. Com isso, para construir cada questão de pesquisa (QP) do estudo de caso, foram selecionadas as vertentes apontadas pelos autores:</p> <ul style="list-style-type: none"> • QP₁: <i>Qual a aplicação das microtasks no desenvolvimento de software crowdsourcing?</i> <ul style="list-style-type: none"> – QP_{1.1}: <i>Qual a sua aplicação no ciclo de desenvolvimento?</i> – QP_{1.2}: <i>Quais as principais plataformas e tecnologias aplicadas?</i> – QP_{1.3}: <i>Qual o índice de sucesso e falha?</i> • QP₂: <i>Como a configuração do crowdsourcing afeta as microtasks?</i> <ul style="list-style-type: none"> – QP_{2.1}: <i>Qual a influência do tamanho do crowd?</i> – QP_{2.2}: <i>Qual a complexidade dos projetos?</i>
	<p><i>Os métodos</i> Para realizar a coleta de dados foi considerada uma extração automatizada, devido ao grande fluxo de informação.</p>
	<p><i>A estratégia</i> Foram consideradas apenas plataformas dedicadas ao desenvolvimento de software CS com algum mecanismo capaz de automatizar a extração de dados.</p>
1.2 Protocolo do estudo de caso	<p><i>Background</i> Os trabalhos apresentados no elemento “a teoria” da etapa 1.1.</p>
	<p><i>Plano</i> Estudo de caso único</p>
	<p><i>Seleção</i> Os critérios de seleção foram baseados nos elementos “os métodos” e “a estratégia” da etapa 1.1. Assim, cabe destacar que o estudo de caso a ser realizado focou apenas em plataformas de desenvolvimento CS com mecanismos automatizados para extração de dados.</p>
	<p><i>Coleção de dados</i> Dados reais capazes de fornecer alguma solução as QPs apresentadas no elemento “as questões analisadas” da etapa 1.1.</p>

Fonte: Autoria própria.

3.1.2 PREPARAÇÃO E COLETA

O nível de coleta de dados para o estudo de caso foi classificado como terceiro nível, representando uma análise de artefatos que estão disponíveis e, por vezes, foram compilados e utilizados. Como argumentaram Runeson e Høst (2008), tradicionalmente, utilizam-se documentos, relatórios e dados oriundos de arquivos e registros para a condução de estudos de caso de terceiro nível. Assim, os dados a serem analisados foram originários de *microtasks* executadas. Além disso, nenhum tipo de entrevista, vídeo ou gravação de áudio foi utilizado. Também não houve reuniões com profissionais ou entusiastas da área para solucionar as QPs.

Após a identificação do nível de coleta de dados, foi realizada efetivamente a coleção de dados para o estudo de caso. Como previamente determinado na etapa 1.2 do “protocolo do estudo de caso”, a seleção dos dados deveriam ocorrer com base em dados reais de uma plataforma de software CS. Além disso, a plataforma deveria fornecer algum mecanismo para automatizar a extração de dados. Em vista desses elementos, foi selecionada uma plataforma dedicada ao desenvolvimento de software CS integrada com uma *application programming interface* (API) que oferece a extração de *microtasks* cadastradas na plataforma.

A plataforma selecionada para conduzir o estudo de caso representa um dos principais mercados de desenvolvimento de software CS. Atualmente, a plataforma contabiliza mais de 1.000.000 de participantes cadastrados. Isso representa uma ampla força de trabalho para a formação de *crowds* e execução de *microtasks*. De acordo com dados fornecidos, a plataforma registra diariamente cerca de 20 novas *microtasks*. Todas as *microtasks* possuem um valor definido pelo *crowdsourcer* em moeda americana. Desse modo, existem *microtasks* simples que chegam a valer poucos dólares, até projetos com desafios complexos, formado por múltiplas *microtasks*, que alcançam alta premiação.

Todavia, apesar de utilizar a moeda americana, a plataforma não restringe a sua utilização nessas fronteiras. Atualmente, encontram-se *microtasks* oriundas de diversas partes do mundo exigindo diferentes expertises para a sua execução. Devido ao barateamento de projeto, força do *crowd* e velocidade de finalização, diversas empresas de porte multinacional estão migrando seus projetos, ou partes deles, para a plataforma. Além disso, muitas parcerias emergiram nesse sentido, em busca de solucionar desafios do desenvolvimento de software por meio do CS.

Apesar do esforço em mensurar a parcela de mercado representado pela plataforma, não foram identificadas fontes confiáveis devido à dinamicidade do mercado ocasionada pela criação de novas plataformas. Ademais, optou-se por manter o anonimato da plataforma em

questão durante a execução do estudo de caso, e dessa forma, nomes e endereços eletrônicos foram omitidos. A plataforma possui uma API para extração de dados das *microtasks* que torna o procedimento completamente automatizado. Contudo, a API limita o período máximo de extração, e com isso, somente foram captadas as *microtasks* cadastradas na plataforma entre as datas 31/12/2011 até 15/12/2016. Ao todo foram recuperadas 14485 *microtasks*, formando um conjunto amplo de dados para as análises das QPs.

Os dados extraídos da API foram gerados em texto puro, e por isso, foram convertidos e tabulados em planilhas eletrônicas. Para a condução do estudo de caso foram aplicadas diferentes análises conforme a natureza de cada QP. Dessa forma, realizaram-se análises do tipo manual, computacional e estatística.

3.1.3 ANÁLISE

Para conduzir a análise do estudo de caso foi utilizada a estatística descritiva. Dessa forma, foram adotados recursos capazes de fornecer as respostas as QPs do estudo de caso. Foram adotados elementos para ilustração das informações e a visualização gráfica. A análise completa do estudo de caso encontra-se na Seção 4.1 presente neste trabalho. Além disso, a experiência fornecida pelo caso analisado também forneceu insumos para os resultados apresentados nas Seções 4.2 e 4.3 deste trabalho.

Deve-se destacar que foram identificados diferentes tipos de ameaças durante a execução do estudo de caso. Com isso, a subseção 4.5.1 deste estudo busca apresentar os principais aspectos que podem ameaçar a confiabilidade dos resultados obtidos.

3.2 EXPERIMENTAÇÃO CONTROLADA

A experimentação controlada é um método empírico de pesquisa amplamente utilizado e difundido na Engenharia de Software. Como argumentou Basili (2006), a sua aplicação vem evoluindo constantemente, buscando a sua expansão e estruturação conforme o avanço das eras e a necessidade de cada pesquisa.

Apesar da ampla adoção de métodos experimentais em diversos cenários, existem pesquisadores que demonstram diferentes argumentos para rejeitar a sua aplicação na Engenharia de software, conforme advertiu Juristo e Moreno (2010). Os autores afirmam que os argumentos propostos na verdade representam falácias que podem ser facilmente refutadas devido à amplitude e qualidade dos resultados obtidos em diversos trabalhos acadêmicos. Todavia, como apresentado por Wohlin et al. (2000), conduzir um experimento controlado não é uma

tarefa simples, exigindo um rigoroso processo que envolve a preparação, a condução e a análise de dados.

Para iniciar um experimento controlado deve-se primeiramente estruturar o protocolo experimental. Esse documento serve como um guia para auxiliar a execução do experimento e validar a pesquisa. Para sintetizar os seus elementos, a Tabela 10 apresenta a estrutura de um protocolo experimental.

Tabela 10: Protocolo resumido para conduzir um experimento controlado na Engenharia de Software

Etapa	Descrição
1ª Etapa	<p>1. Escopo</p> <p>1.1 <i>Definição dos objetivos</i>: deve-se definir os principais aspectos para a execução do experimento controlado.</p>
2ª Etapa	<p>2. Planejamento</p> <p>2.1 <i>Seleção de contexto</i>: pode ser caracterizado de acordo com as dimensões que o experimento controlado será executado, como o ambiente utilizado e os participantes.</p> <p>2.2 <i>Formulação de hipóteses</i>: as hipóteses devem ser formuladas. O teste de hipótese ocorre usando os dados gerados por meio do uso da estatística.</p> <p>2.3 <i>Seleção das variáveis</i>: as variáveis são separadas em independentes e dependentes. As variáveis independentes podem ser modificadas e controladas durante o experimento, enquanto que as variáveis dependentes são, geralmente, derivadas diretamente da hipótese e não podem ser manipuladas.</p> <p>2.4 <i>Instrumentalização</i>: identificação dos objetos, guias ou instrumentos de medição que deverão ser utilizados no experimento.</p>
3ª Etapa	<p>3. Operação</p> <p>3.1 <i>Preparação</i>: deve-se selecionar e informar os participantes do experimento. Ademais, as ferramentas e os materiais disponíveis também devem ser preparados.</p> <p>3.2 <i>Execução</i>: ocorre efetivamente a execução do experimento por parte dos participantes.</p> <p>3.3 <i>Validação dos dados</i>: verificar se os participantes compreenderam o objetivo experimental e executaram de forma correta.</p>
4ª Etapa	<p>4. Análise e Interpretação</p> <p>4.1 <i>Estatística descritiva</i>: apresentação e processamento numérico do conjunto de dados gerado pelo experimento.</p> <p>4.2 <i>Redução do conjunto de dados</i> remover os dados inválidos, geralmente gerados por participantes que não compreenderam ou não executaram de forma consciente a experimentação.</p> <p>4.3 <i>Teste de hipótese</i>: verificar se é possível rejeitar a hipótese nula (H_0).</p>

Fonte: Adaptado de Wohlin et al. (2000).

Antes de realizar a execução de um experimento controlado é recomendada a execução de ensaios, geralmente conhecidos como testes experimentais (WOHLIN et al., 2000). Esses testes são utilizados para aferir o protocolo experimental, buscando identificar erros e corrigir falhas.

A experimentação controlada foi selecionada para finalizar a investigação sobre a caracterização das *microtasks* no desenvolvimento de software CS. Assim, foram realizados dois testes experimentais para fornecer experiências e mitigar erros antes da realização do experimento controlado. A seguir, é exposta a execução dos testes bem como do experimento, que foram realizados conforme o protocolo exposto na Tabela 10.

3.2.1 PRIMEIRO TESTE EXPERIMENTAL

O objetivo do primeiro teste experimental foi identificar se o CS poderia ser simulado em um ambiente acadêmico. Devido a isso, uma sala de aula de aula foi configurada e os participantes foram arranjados em um modelo que demonstrava uma simulação reduzida de um ambiente CS. Aspectos como a comunicação por ferramentas e distanciamento entre *crowd* e *crowdsourcer* foram evidenciados. A Tabela 11 apresenta uma síntese das etapas e elementos utilizados no primeiro teste experimental.

Tabela 11: Protocolo do primeiro teste experimental

Etapa	Conteúdo
Esco- po	<i>O objetivo</i> Identificar se o CS pode ser simulado em um ambiente acadêmico.
	<i>Seleção de contexto</i> Totalmente acadêmico, com estudantes de graduação dos cursos de Análise e Desenvolvimento de Sistemas e Engenharia da Computação da Universidade Tecnológica Federal do Paraná - Campus Cornélio Procopio.
Planejamento	<i>Formulação das hipótese</i> H_0 : O CS não poderá ser simulado em um ambiente acadêmico. H_1 : O CS poderá ser simulado em um ambiente acadêmico.
	<i>Seleção das variáveis</i> Foram selecionadas duas variáveis independentes: vi_1) total de participantes e vi_2) tempo utilizado. Também foram selecionadas duas variáveis dependentes: vd_1) distância entre <i>crowd</i> e <i>crowdsourcer</i> e vd_2) ferramenta de comunicação.
	<i>Instrumentalização</i> Foi selecionado um guia eletrônico capaz de auxiliar os participantes na execução do experimento.
Operação	<i>Preparação</i> Os participantes foram consultados e alertados para a execução do teste experimental. O guia eletrônico também foi oferecido aos participantes, bem como acesso à ferramenta de comunicação.
	<i>Execução</i> Os participantes possuíram cerca de 1h e 30min para a execução do experimento.
	<i>Validação de dados</i> Os produtos gerados pelos participantes foram coletados e as informações necessárias foram extraídas.
Análise e Interpretação	<i>Estatística descritiva</i> Foram utilizados recursos de visualização gráfica para apresentação dos dados.
	<i>Redução do conjunto de dados</i> Dados gerados por participantes que não executaram de maneira consciente o teste experimental foram removidos.
	<i>Teste de hipótese</i> Verificar se foi possível simular o CS no ambiente acadêmico.

Fonte: Autoria própria.

Para conduzir o primeiro teste experimental foram utilizados papéis e características tradicionalmente encontradas em projetos CS (SAREMI; YANG, 2015a; STOL; FITZGERALD, 2014a) e (LI et al., 2015). Foram definidos três papéis nomeados *crowd*, *crowdsourcer* e *microtasks*. Além desses, também foram definidos aspectos como a comunicação por ferramentas

entre *crowd* e *crowdsourcer*, dispersão dos participantes e condução de *feedbacks*.

Para efetuar o procedimento de avaliação do teste experimental foi necessário utilizar uma iniciativa CS. Com isso, recorreu-se ao levantamento realizado por Aris e Din (2016) que evidenciou uma lista de características e iniciativas que podem ser utilizadas para o uso e a simulação do CS. Devido ao ambiente acadêmico e as restrições temporais optou-se pelo uso dos Robôs *Lego Mindstorms*. Os robôs *Mindstorms* possuem um conjunto de peças que podem ser combinadas para a montagem de diversos modelos. Dessa forma, as peças podem ser distribuídas entre os participantes a fim de simularem um projeto CS. Ademais, os robôs também possuem um guia eletrônico que apresenta o código, a descrição e as dimensões de cada peça.

O primeiro teste experimental foi organizado para verificar se o *crowd* poderia realizar a montagem de um robô *Lego Mindstorms* por meio de CS. Para isso, *crowd* e *crowdsourcer* deveriam realizar três *microtasks*: (i) o *crowdsourcer* deveria coordenar a montagem e solicitar ao *crowd* a próxima parte do robô a ser montada; (ii) o *crowd* deveria procurar no guia qual peça se encaixaria ao pedido do *crowdsourcer* e enviar o código da peça; (iii) o *crowdsourcer* deveria enviar o *feedback* da montagem. Os participantes possuíam acesso ao guia, porém apenas o *crowdsourcer* possuía acesso as peças do kit. *Crowd* e *crowdsourcer* ficaram fisicamente distantes e a comunicação e os *feedbacks* ocorreram por meio de um sistema que registrava a comunicação.

3.2.2 SEGUNDO TESTE EXPERIMENTAL

O objetivo do segundo teste experimental foi identificar a possibilidade de execução de *microtasks* em um projeto de software CS simulado em um ambiente acadêmico. Com isso, um projeto de software foi configurado e os participantes foram alocados de forma distribuída. O segundo teste experimental pode ser considerado um amadurecimento do procedimento experimental. No primeiro teste, o CS foi simulado com os participantes no mesmo ambiente, não foi utilizado nenhum projeto de software, e as *microtasks* eram fixas. Com isso, no segundo teste todos esses aspectos foram otimizados para a realidade do desenvolvimento de software CS. A descrição completa do protocolo do segundo teste experimental é encontrado na Tabela 12.

No segundo teste experimental os participantes deveriam desenvolver um software por meio das *microtasks*. Sendo assim, os participantes receberam a seguinte orientação: “o produto a ser desenvolvido trata-se de um ambiente de integração entre clientes e prestadores de serviços. Um cliente pode buscar um profissional/empresa que presta um determinado serviço, negociar preço, forma de pagamento, etc”. Com essa orientação, os participantes deveriam

elaborar os requisitos, a documentação e o desenvolvimento do projeto via CS. Inicialmente, cada participante deveria elaborar um requisito do projeto. Após a elaboração, o *crowdsourcer* analisava se o requisito poderia ser aprovado ou reprovado. Em caso de aprovação o requisito era liberado na plataforma para ser documentado por outro participante. Após a conclusão da documentação o *crowdsourcer* analisava e verifica se a documentação do requisito poderia ser aprovada ou reprovada. Somente após a aprovação da documentação o requisito era liberado para desenvolvimento.

Tabela 12: Protocolo do segundo teste experimental

Etapa	Conteúdo
Escopo	<i>O objetivo</i> Identificar a possibilidade de executar <i>microtasks</i> de um projeto de software CS em um ambiente acadêmico.
	<i>Seleção de contexto</i> Similarmente ao primeiro teste experimental, também foi utilizado um contexto acadêmico, com estudantes de graduação dos cursos de Análise e Desenvolvimento de Sistemas e Engenharia da Computação da Universidade Tecnológica Federal do Paraná - Campus Cornélio Procopio.
Planejamento	<i>Formulação das hipótese</i> H_0 : Não será possível executar <i>microtasks</i> de um projeto de software CS em um ambiente acadêmico. H_1 : Será possível executar <i>microtasks</i> de um projeto de software CS em um ambiente acadêmico.
	<i>Seleção das variáveis</i> Foram selecionadas duas variáveis independentes: vi_1) total de <i>microtasks</i> submetidas e vi_2) índice de <i>microtasks</i> aprovadas. Também foram selecionadas duas variáveis dependentes: vd_1) dispersão física e vd_2) granularidade das <i>microtasks</i> .
	<i>Instrumentalização</i> Foi selecionado um repositório comum para o projeto CS em que todos os participantes possuíam acesso. Além disso também foi utilizada uma ferramenta de comunicação.
Operação	<i>Preparação</i> Os participantes foram consultados e alertados para a execução do teste experimental. O repositório do projeto foi apresentado e os participantes foram ambientados com a ferramenta de comunicação.
	<i>Execução</i> Os participantes possuíam cerca de 30 dias para a execução do experimento.
	<i>Validação de dados</i> As <i>microtasks</i> geradas e executadas pelos participantes foram coletadas no final do teste experimental
Análise e interpretação	<i>Estatística descritiva</i> Foram utilizados recursos de visualização gráfica para apresentação dos dados.
	<i>Redução do conjunto de dados</i> Similarmente ao primeiro teste experimental, os dados gerados por participantes que não executaram de maneira consciente o teste experimental foram removidos.
	<i>Teste de hipótese</i> Verificar se foi possível simular e executar as <i>microtasks</i> de um projeto de software CS em um ambiente acadêmico.

Fonte: Autoria própria.

Por tratar-se de um sistema desenvolvido por CS os requisitos foram fiscalizados conforme a sua granularidade. Requisitos considerados complexos, que demandariam emprego de muito tempo, esforço ou integração externa de desenvolvimento eram sumariamente con-

siderados reprovados pelo *crowdsourcer*, que recebeu ajuda de um orquestrador auxiliar. Os requisitos com complexidade moderada foram decompostos em dois ou mais requisitos com o intuito de realizar uma simplificação. Por isso, os requisitos elicitados finais tornaram-se, em grande parte, simples e foram considerados *microtasks*.

3.2.3 EXPERIMENTO CONTROLADO

O objetivo da execução do experimento controlado foi propor uma abordagem para aferir a complexidade de uma *microtask* e verificar a sua eficácia. No experimento controlado as experiências obtidas pelos dois testes experimentais foram fundamentais durante a fase de elaboração e execução. Inicialmente, percebeu-se que a granularidade das *microtasks* afeta a duração do projeto e onera o *crowdsourcer*. Por isso, as *microtasks* foram definidas utilizando dados reais de uma plataforma CS, ao invés de deixar o *crowd* definir a *microtask* e depois fiscalizar a sua granularidade (como conduzido no segundo teste experimental). Além disso, percebeu-se também a necessidade de mensurar a experiência dos participantes por meio de um questionário, ato não realizado no primeiro teste experimental.

No experimento controlado os participantes representaram alunos de mestrado do Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – Campus Cornélio Procópio. Portanto, as experiências eram mescladas em contextos acadêmicos e profissionais, visto que diversos mestrandos possuem experiências no setor produtivo de software. Os participantes estavam fisicamente e temporalmente distribuídos durante a execução experimental, com isso, foi minimizada a incidência de contato direto. Ademais, as apresentações do experimento foram gravadas e disponibilizadas eletronicamente, minimizando a comunicação pessoal. Devido à distribuição dos participantes, foi utilizada uma ferramenta de comunicação em que todos os participantes foram convidados a acessarem e apresentarem possíveis dúvidas durante a execução experimental.

Para fornecer maior confiabilidade na execução, as *microtasks* utilizadas no experimento controlado foram inspiradas em dados de *microtasks* reais. Por isso, a distribuição, a comunicação e as *microtasks* do experimento apresentavam alta similaridade com a realidade do desenvolvimento de software CS.

A dinâmica de funcionamento do experimento controlado foi a seguinte: os participantes foram convidados a integrarem o projeto e submeterem contribuições. Cada participante devia selecionar uma *microtask*, executá-la e submetê-la. Ao selecionar uma *microtask*, o participante deveria descrever qual motivo o guiou para a seleção. Não havia um número mínimo ou máximo de contribuições, dessa forma os participantes eram independentes na esco-

lha, execução e submissão. Entretanto, foi adicionado um mecanismo para evitar que diversos participantes se concentrassem em uma única *microtask*. Assim, quando alguma *microtask* apresentava um índice concentrado de registro, ela era travada e não aceitava registros de novos participantes. O protocolo completo do experimento controlado é encontrado na Tabela 13.

Tabela 13: Protocolo do experimento controlado

Etapa	Conteúdo
Esco- po	<i>O objetivo</i> Propor e validar uma abordagem capaz de aferir a complexidade das <i>microtasks</i> .
	<i>Seleção de contexto</i> Contexto híbrido, mesclando profissionais da academia com profissionais do setor produtivo. Além disso, os participantes possuíam diferentes experiências e conhecimentos, contribuindo com a simulação do CS.
	<i>Formulação das hipótese</i> H_0 : O índice de execução de uma <i>microtask</i> não poderá ser aferido com base na abordagem proposta para mensurar a complexidade de uma <i>microtask</i> . H_1 : O índice de execução de uma <i>microtask</i> poderá ser aferido com base na abordagem proposta para mensurar a complexidade de uma <i>microtask</i> .
	<i>Seleção das variáveis</i> Foram selecionadas duas variáveis independentes: vi_1) liberdade de seleção de <i>microtask</i> e vi_2) tempo gasto para finalizar a <i>microtask</i> . Foram selecionadas três variáveis dependentes: vd_1) características das <i>microtasks</i> , vd_2) índice de registro e vd_3) índice de submissão.
Planejamento	<i>Instrumentalização</i> Inicialmente, o projeto de software do segundo teste experimental foi refinado e alocado em um repositório. Após isso, foram captadas informações de <i>microtasks</i> reais e cadastradas no repositório. Foram gerados guias e apresentações que foram armazenados no repositório. Por fim, foi selecionada uma ferramenta de comunicação.
	<i>Preparação</i> Os participantes foram consultados para a participação voluntária do experimento controlado. Logo após, os participantes foram introduzidos com o experimento por meio de uma apresentação <i>online</i> em que foi exposto o repositório, o projeto, as <i>microtasks</i> e os guias de auxílio. Além disso, os participantes também foram convidados ao acesso da ferramenta de comunicação.
	<i>Execução</i> Os participantes possuíam cerca de 10 dias para a execução do experimento.
Operação	<i>Validação de dados</i> Todas as <i>microtasks</i> selecionadas e executadas foram analisadas. Assim, foi verificado a modificação a nível de código ou submissão de artefatos.
	<i>Estatística descritiva</i> Foram utilizados recursos de visualização gráfica para apresentação dos dados.
Análise e Interpretação	<i>Redução do conjunto de dados</i> Os participantes que não compreenderam ou não conduziram o experimento de forma consciente foram removidos.
	<i>Teste de hipótese</i> Foi utilizada um teste estatístico para realizar o teste de hipótese.

Fonte: Autoria própria.

Similarmente ao estudo de caso, as ameaças à validade do experimento controlado são apresentadas na subseção 4.5.1 deste estudo. Os resultados captados pela execução do experimento são discutidos na Seção 4.4 deste trabalho.

3.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a abordagem híbrida adotada para a validação desta pesquisa. Os métodos e procedimentos aplicados correspondem ao uso estudo de caso e da experimentação controlada. Todavia, para aumentar a qualidade final da investigação, as considerações reveladas no MS também foram aplicadas em determinados cenários. Para simplificar a compreensão da abordagem utilizada, a Tabela 14 apresenta uma síntese geral dos métodos e procedimentos adotados.

Tabela 14: Visão geral dos métodos, procedimentos, contribuições e divulgação dos resultados

Caracterização	Descrição
<ul style="list-style-type: none"> • Uso <ul style="list-style-type: none"> ○ <i>Métodos e Procedimentos</i> <ul style="list-style-type: none"> – Explicação – Operacionalização ○ <i>Contribuições</i> <ul style="list-style-type: none"> – Resultados 	<p>Estudo de Caso.</p> <p>Seção 3.1.</p> <p>Subseções 3.1.1, 3.1.2 e 3.1.3.</p> <p>Aplicação das <i>microtasks</i> e a influência que a configuração do CS exerce.</p> <p>Seção 4.1.</p>
<ul style="list-style-type: none"> • Características <ul style="list-style-type: none"> ○ <i>Métodos e Procedimentos</i> <ul style="list-style-type: none"> – Explicação – Operacionalização ○ <i>Contribuições</i> <ul style="list-style-type: none"> – Resultados 	<p>Mapeamento Sistemático/Estudo de Caso.</p> <p>Seções 2.2/3.1</p> <p>Subseções 2.2.1, 2.2.2 e 2.2.3/3.1.1, 3.1.2 e 3.1.3.</p> <p>Características e estados das <i>microtask</i>. Metamodelo de uma <i>microtask</i>.</p> <p>Seção 4.2.</p>
<ul style="list-style-type: none"> • Contraste <ul style="list-style-type: none"> ○ <i>Métodos e Procedimentos</i> <ul style="list-style-type: none"> – Explicação – Operacionalização ○ <i>Contribuições</i> <ul style="list-style-type: none"> – Resultados 	<p>Mapeamento Sistemático/Estudo de Caso.</p> <p>Seções 2.2/3.1</p> <p>Subseções 2.2.1, 2.2.2 e 2.2.3/3.1.1, 3.1.2 e 3.1.3.</p> <p>Originalidade da investigação. Os contrastes entre o líder, o time e o processo.</p> <p>Seção 4.3</p>
<ul style="list-style-type: none"> • Complexidade <ul style="list-style-type: none"> ○ <i>Métodos e Procedimentos</i> <ul style="list-style-type: none"> – Explicação – Operacionalização ○ <i>Contribuições</i> <ul style="list-style-type: none"> – Resultados 	<p>Experimentação controlada.</p> <p>Seção 3.2</p> <p>Subseções 3.2.1, 3.2.2 e 3.2.3</p> <p>Aferir a complexidade. Fatores que podem interferir a execução de uma <i>microtask</i>.</p> <p>Seção 4.4.</p>
<ul style="list-style-type: none"> • Divulgação <ul style="list-style-type: none"> ○ <i>Artigos publicados</i> ○ <i>Sob avaliação</i> ○ <i>Contribuições</i> 	<p>Deus et al. (2016), Deus et al. (2017b), Deus et al. (2017c) e Deus et al. (2017d).</p> <p>(ES01*), (ES02*) e (ES03**)</p> <p>L’Erario et al. (2017), Deus et al. (2017a)</p>

Fonte: Autoria própria.

Nota: (*) Estudos que estão sob avaliação em periódicos. Cabe destacar que o estudo (ES01) foi avaliado por pares e requisitado um conjunto de considerações antes de efetivar a publicação. As considerações foram aplicadas e agora aguarda-se a decisão final dos editores. (**) O artigo foi aceito na Conferência Latinoamericana de Informática, porém foi direcionado para a publicação em um simpósio interno do evento ao invés da trilha principal. Com isso, os autores decidiram realizar uma nova tentativa de publicação em outro evento a ser definido.

O motivo da adoção de uma abordagem híbrida para validar este trabalho pode trazer a luz amplos questionamentos. Assim, para evitar possíveis dúvidas desse sentido, deve-se

advertir que a pesquisa desenvolvida nesse trabalho levou em consideração, principalmente, a incipiência da literatura sobre o tema analisado, como foi endereçada no capítulo 2 deste trabalho. Logo, foi necessário adotar um processo de investigação em um conjunto sequencial de quatro ciclos. Após a conclusão de um ciclo, eram fornecidos insumos para a execução do próximo ciclo. Assim sendo, a seguinte dinâmica foi adotada:

- No primeiro ciclo, tendo em vista a escassez de estudos e os sintomas da literatura, foi realizado um estudo de caso para identificar como as *microtasks* estão sendo utilizadas no desenvolvimento de software CS.
- O foco do segundo ciclo foi embasado na dificuldade em compreender e definir uma *microtask*. Assim, com a experiência adquirida pelo caso analisado em união as visões dos autores investigados no MS foi gerada uma taxonomia sobre as *microtasks*. Com isso foi possível apresentar as características que todas, ou grande parte das *microtasks*, possuem em comum;
- Na execução do terceiro ciclo foi levado em consideração as lacunas da literatura sobre os contrastes das *microtasks* em relação as atividades que são tradicionalmente executadas no desenvolvimento de software. Assim, similarmente ao ciclo anterior, munido da experiência do caso analisado com as considerações reveladas no MS, foi realizada uma comparação e evidenciada as diferenças de comportamento entre as *microtasks* e tais atividades;
- Por fim, no quarto ciclo, considerando a experiência fornecida pelos ciclos anteriores e a ausência de métricas para aferir a complexidade de uma *microtask*, foi gerada uma abordagem capaz de mensurar o esforço necessário. E para a criação e validação da abordagem proposta foi adotado o uso da experimentação controlada.

Em resumo, este estudo adotou uma abordagem híbrida para a sua condução. Tal estratégia foi necessária devido ao atual estado da arte da literatura sobre *microtasks*, ambientada por contradições entre autores, escassez de estudos e dificuldades de compreensão do tema. Por fim, deve-se ressaltar que a abordagem foi executada em diferentes ciclos, e cada ciclo representou uma contribuição em torno das *microtasks*.

4 RESULTADOS

Os resultados expostos neste capítulo representam uma síntese sobre a caracterização das *microtasks* aplicadas ao desenvolvimento de software CS. À vista disso, o capítulo encontra-se organizado em cinco seções: a Seção 4.1 introduz o uso de *microtasks*, a Seção 4.2 apresenta as características das *microtasks*, a Seção 4.3 investiga o contraste das *microtasks* CS com as atividades DDS, a Seção 4.4 demonstra uma abordagem para mensurar a complexidade das *microtasks*, e a Seção 4.5 concentra os principais pontos de destaque do capítulo.

4.1 USO

Para investigar o uso das *microtasks* foi conduzido um estudo de caso, e o caso analisado pertence a uma plataforma de desenvolvimento de software CS via *microtasks*. O processo de desenvolvimento da plataforma utiliza etapas distintas e complementares ao desenvolvimento tradicional de software, como ilustra a Figura 10.

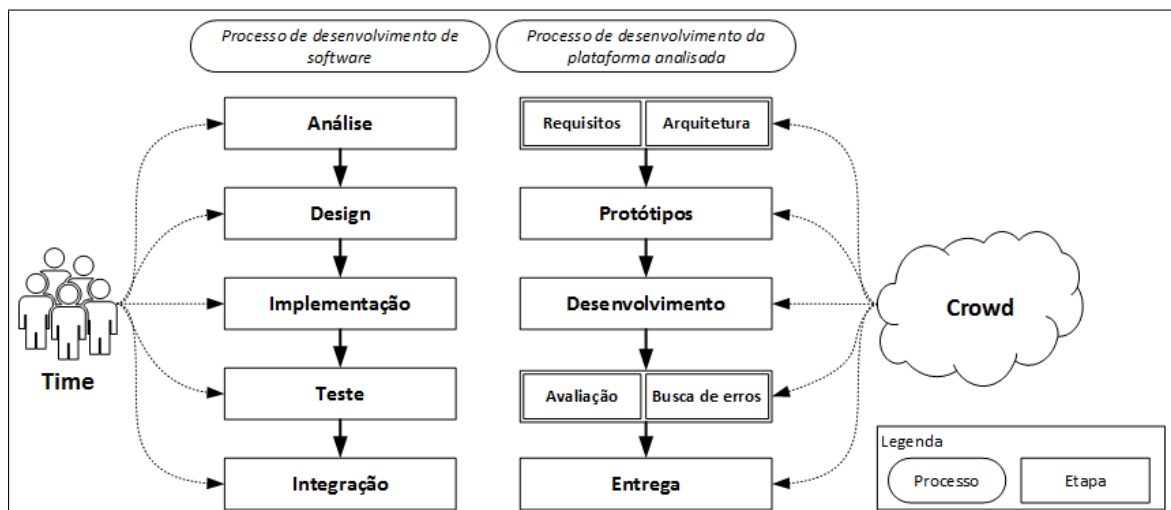


Figura 10: Ilustração do processo de desenvolvimento tradicional e o *crowdsourcing*

Fonte: Autoria Própria

Em síntese, cada etapa do desenvolvimento tradicional possui uma etapa correspon-

dente na plataforma analisada. Contudo, as etapas análogas de Análise e Teste foram divididas em duas etapas distintas. De modo geral, a Análise acaba sendo classificada em “Requisitos” para projetos que serão modelados e “Arquitetura” para projetos já existentes. O Teste é separado em “Avaliação” quando se deseja apenas identificar erros ou “Busca de erros” quando se pretende identificá-los e corrigi-los. As demais etapas do processo da plataforma modificam apenas a terminologia, porém a finalidade permanece similar ao processo tradicional.

Além disso, na execução das *microtasks*, os participantes são livres para navegarem por cada etapa do processo e selecionarem as *microtasks*. Os participantes que submeterem as melhores soluções, ou as mais rápidas, recebem uma recompensa financeira. Se um participante selecionar uma *microtask* e não submeter nenhuma solução não existe nenhum tipo de multa ou penalidade.

É necessário esclarecer que cada etapa do processo de desenvolvimento da plataforma possui um conjunto de classificações para organizar e facilitar a distribuição das *microtasks*. Porém, apesar disso, diversas *microtasks* são cadastradas de maneira errônea. Isso ocorre, em grande parte, devido à falta de mecanismos para gerenciar e controlar o cadastro e execução de *microtasks*. Ademais, não existe controle sobre a granularidade das *microtask*, assim, são encontradas tarefas simples até tarefas com alto grau de complexidade.

Tendo em vista todo o processo de desenvolvimento, bem como o funcionamento da plataforma, o caso analisado demonstrou alta aderência ao objetivo proposto: compreender como as *microtasks* estão sendo utilizadas no desenvolvimento de software CS. Ademais, o caso analisado pertence a uma das principais plataformas de desenvolvimento CS, possuindo constante cadastro de novas *microtasks*; amplo grupo de participantes; e mecanismos para automatizar a extração de dados.

Para conduzir o estudo de caso foram extraídos dados de 14485 *microtasks* cadastradas na plataforma. Os dados representaram uma importante gama de informação, cobrindo um período de praticamente meia década de desenvolvimento de software CS. Diversas particularidades foram extraídas, como as informações sobre a duração, premiação, descrição, tecnologias e status das *microtasks*. O processo de extração foi automatizado, assim foi gerado um arquivo em texto puro que logo após foi convertido e tabulado para uma planilha eletrônica. Todavia, apesar de grande parte dos dados disponíveis estarem previamente padronizados pela própria plataforma, algumas análises necessitaram de maior atenção. Nesse sentido, deve-se destacar que para identificar a aplicação no ciclo de vida dos projetos de software foi necessário conduzir uma análise manual em um conjunto restrito de *microtasks* para compreender e sistematizar o restante do conjunto de dados. Ademais, análises que necessitavam da interpretação

de informação (como a leitura da descrição das *microtasks*) foram realizadas a partir de um algoritmo de leitura e ordenação de palavras. Finalmente, as demais análises foram realizadas de maneira estatística. Para simplificar a compreensão dos resultados obtidos, o restante dessa seção do trabalho traz luz a cada QP do estudo de caso e as considerações finais sobre o uso de *microtasks*.

4.1.1 A APLICAÇÃO DAS *MICROTASKS* NO DESENVOLVIMENTO DE SOFTWARE *CROWDSOURCING*

A plataforma utilizada para extração de dados possibilita a execução de *microtasks* por meio de duas formas: colaborativa e competitiva. As *microtasks* colaborativas são executadas pelo *crowd* por meio da participação massiva, dessa forma diversas soluções derivadas de vários participantes são utilizadas em uma *microtask*. Um exemplo bem estabelecido de *microtask* colaborativa refere-se à busca por erros em um sistema. Enquanto isso, nas *microtasks* competitivas ocorre uma disputa interna no *crowd* para que um participante apresente uma solução no menor tempo possível. Todavia, existem casos que as *microtasks* competitivas permitem um número maior de soluções, variando conforme a decisão do *crowdsourcer*. Um exemplo bem estabelecido de *microtask* competitiva é a criação de *layouts* ou prototipação de *interfaces*. Percebe-se que as *microtasks* colaborativas ocupam o maior índice de uso, como mostra a Figura 11.

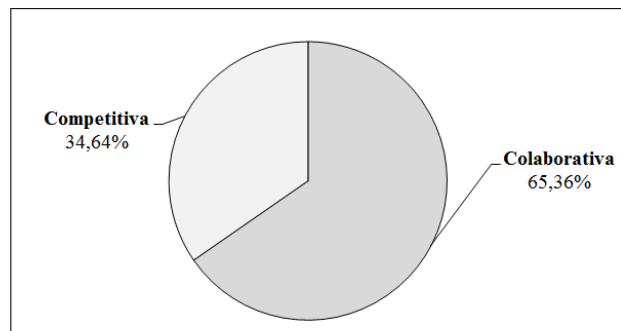


Figura 11: O tipo de uso das *microtasks*

Fonte: Autoria Própria

Após identificar o modo de aplicação das *microtasks* no desenvolvimento de software CS foram analisados os domínios de utilização no que diz respeito ao ciclo de desenvolvimento de um projeto de software.

QP_{1.1} *Qual a sua aplicação no ciclo de desenvolvimento?*

Como foi endereçada no Figura 10 (página 54), a plataforma utilizada no estudo de

caso possui diferentes etapas para execução no processo de desenvolvimento CS. Cada etapa possui um conjunto de categorias para identificar e classificar as *microtasks*. Todavia, existem duas categorias nomeadas “*Desktop*” e “*Primeiro a finalizar*” que podem conter *microtasks* de qualquer etapa do ciclo de desenvolvimento, enquanto que as demais categorias são endereçadas à uma única etapa. A relação das categorias das *microtasks* é encontrada na Tabela 15.

Tabela 15: Finalidade e Categoria das *microtasks*

Etapa	Categoria	Finalidade
Requisitos	1. Ideia	<i>Crowdsourcer</i> possui uma ideia de projeto que precisa ser aperfeiçoada
Requisitos	2. Especificação	<i>Crowdsourcer</i> possui ideia estruturada e precisa extrair os requisitos
Requisitos	3. Conceitualização	<i>Crowdsourcer</i> possui os requisitos mas precisa da documentação (diagramas, casos de uso, etc)
Arquitetura	4. Arquitetura	<i>Crowdsourcer</i> possui um projeto funcionando ou estruturado, porém precisa remodelá-lo
Protótipo	5. <i>Design</i>	<i>Crowdsourcer</i> precisa de ajuda em atividades gerais relacionadas ao design
Protótipo	6. Criação de conteúdo	<i>Crowdsourcer</i> precisa “recheiar” o projeto, seja adicionando conteúdo à páginas ou reescrevendo/reformatando
Protótipo	7. Protótipo	<i>Crowdsourcer</i> precisa de protótipos, <i>mock ups</i> , etc
Desenvolvimento	8. Desenvolvimento	<i>Crowdsourcer</i> precisa de ajuda em atividades gerais relacionadas ao desenvolvimento (banco de dados, <i>web</i> , servidor, etc)
Desenvolvimento	9. Código	<i>Crowdsourcer</i> precisa de ajuda em atividades focadas na codificação do projeto
Avaliação	10. Em conjunto	<i>Crowdsourcer</i> necessita de testes em diversas partes do projeto
Avaliação	11. Em cenário	<i>Crowdsourcer</i> necessita de testes específicos utilizando cenários
Busca de erros	12. Busca de erros	<i>Crowdsourcer</i> necessita que o projeto seja testado a procura de erros
Entrega	13. Montagem	<i>Crowdsourcer</i> necessita unir diversas partes do projeto
	14. Desktop	<i>Microtasks</i> para sistemas <i>desktop</i>
	15. Primeiro a finalizar	<i>Microtasks</i> dedicadas aos desafios que exigem soluções rápidas

Fonte: Autoria própria.

Nota: Para identificar a finalidade de cada categoria, um grupo restrito de *microtasks* foi analisado.

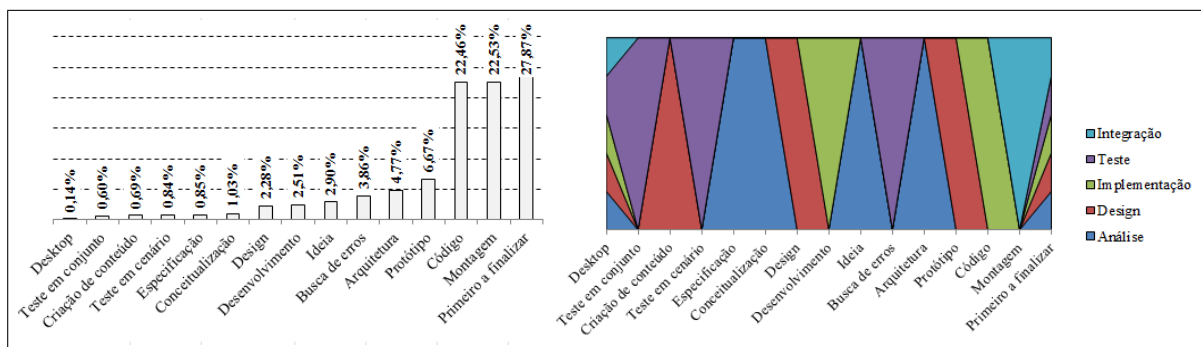


Figura 12: A porcentagem e o uso das *microtasks* no ciclo de desenvolvimento

Fonte: Autoria Própria

A Figura 12 exibiu as diferentes porcentagens de distribuição das *microtasks* de acordo com a sua categoria e a dispersão das *microtasks* no que diz respeito as etapas do ciclo de desenvolvimento. Percebeu-se que as *microtasks* que não pertencem à nenhuma etapa do ciclo de desenvolvimento possuem utilização antagônica. A categoria “Desktop” não alcançou 0.15% de utilização, evidenciando provável tendência no setor produtivo de software. Enquanto isso, a categoria “Primeira a finalizar” concentrou quase 30% de utilização. Além disso, destacam-se também as categorias “Montagem” e “Código” que reuniram mais que 20% da utilização. As demais categorias não ultrapassaram 7% de incidência. Apesar do ciclo de desenvolvimento ser sequencial as análises demonstraram um conjunto distinto de categorias dedicadas a cada etapa. Inicialmente a etapa de Análise concentra o maior número de categorias, seguida pelas etapas de *Design* e o Teste. A etapa de Implementação aparece logo após, finalizando com a etapa de Integração.

Uma grave atestação verificada na análise foi o papel e os erros gerados pelo *crowdsourcer*. Apesar da plataforma fornecer um conjunto de informação para auxiliar a alocação de uma *microtask*, diversos *crowdsourcers* selecionaram categorias consideradas errôneas. Foram encontrados exemplos de *microtasks* categorizadas à etapa de Integração que deveriam referenciar categorias das etapas de Análise e *Design*. Esse tipo de inconsistência pode acarretar diversos desafios na condução de um projeto CS, como a seleção de colaboradores com baixa experiência na área ou demora e falha de execução.

QP_{1.2} *Quais as principais plataformas e tecnologias empregadas?*

As plataformas referem-se à finalidade do projeto ou a sua aplicação, enquanto que as tecnologias referem-se aos meios utilizados para a execução das *microtasks*. Identificaram-se 8 plataformas, e um amplo conjunto de tecnologias. As plataformas foram ordenadas da seguinte maneira:

- **Serviços:** a plataforma mais comum de utilização refere-se aos serviços, ou projetos existentes e que precisam de alguma melhoria. Nessa plataforma destacam-se *microtasks* executadas pelas tecnologias Java, C e VB.Net.
- **Mobile:** a segunda plataforma mais comum refere-se ao emprego de *microtasks* referente aos dispositivos portáteis (*tablets*, *smartphones*, etc). Na plataforma móvel há uma grande utilização de *microtasks* apoiadas na tecnologia Android.
- **Web:** plataforma referente as *microtasks* executadas em algum navegador. Destacam-se diversas tecnologias, como *JavaScript*, *Angular.js* e *Jquery*.
- **Sistemas operacionais:** plataforma dedicada as *microtasks* cuja finalidade é gerenciar os

recursos ou a inicialização de um sistema. Nessa plataforma destaca-se a utilização da tecnologia iOS, que em determinados casos também pode ser classificada como tecnologia móvel.

- **Nuvem:** nessa plataforma encontram-se as *microtasks* que utilizam algum tipo de requisição feita a nuvem, como armazenamento ou processamento de dados. A tecnologia com maior evidência em plataformas de nuvem refere-se à *Salesforce*.
- **Redes sociais:** plataforma dedicada a concentrar *microtasks* cuja finalidade é alguma melhoria ou desenvolvimento em uma rede social. Essa plataforma apresenta grande amplitude de tecnologias, como *Node.js*, *PostgreSQL*, *ReactJS*, entre outras.
- **API:** nessa plataforma concentram-se as *microtasks* para interface de programação. Destaca-se a tecnologia referente ao *Google API*.
- **Outros:** as *microtasks* que não se enquadraram em nenhuma das categorias foram sumariamente ordenadas como outros.

O índice de uso de cada plataforma encontrada nas *microtasks* é apresentado na Figura 13.

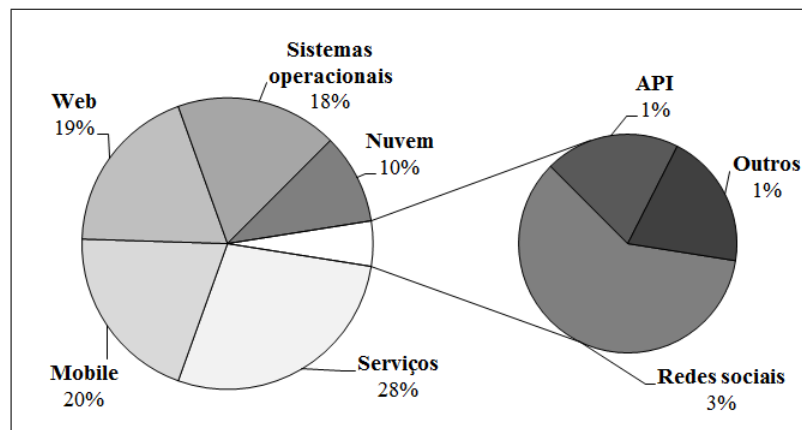


Figura 13: As plataformas usadas nas *microtasks*

Fonte: Autoria Própria

Após analisar as plataformas das *microtasks* foi investigado o emprego das tecnologias. Logo, percebeu-se que as tecnologias representaram um amplo aglomerado de opções, totalizando 148 tecnologias distintas.

As tecnologias tradicionalmente usadas em projetos *web* demonstraram maior utilização. Nesse sentido destacam-se as *microtasks* executadas por meio via *JavaScript*, *Cascading*

Style Sheets (CSS), Node.js, Hypertext Markup Language (HTML), Angular.js e HTML5. *Microtasks* executadas via Java também representaram uma parcela significativa de utilização. Alcançando a segunda posição entre as tecnologias mais empregadas nas *microtasks*.

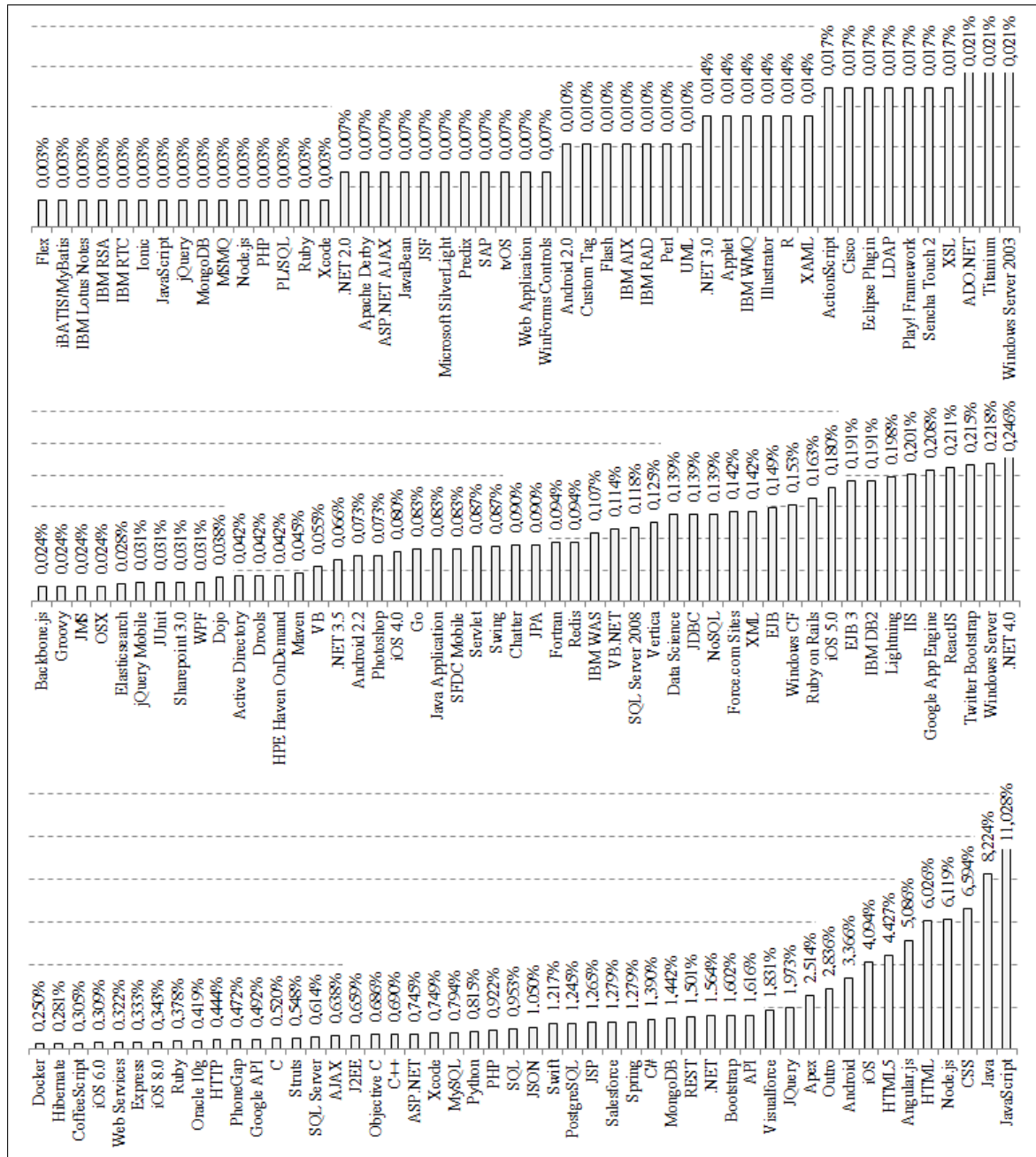


Figura 14: As tecnologias presentes nas *microtasks*

Fonte: Autoria Própria

Contudo, o que se destaca é a limitação dos *crowdsourcers* na definição sobre o uso de uma tecnologia na *microtask*. Para alertar isso foi criada uma categoria denominada “outro” para simbolizar as *microtasks* que não sinalizavam nenhuma tecnologia de execução ou sinali-

zavam de modo superficial as tecnologias utilizadas, alcançando a décima posição. A relação total das tecnologias identificadas e o seu índice de aplicação foi apresentada na Figura 14. A Figura foi organizada em três níveis distintos: no nível superior estão presentes as tecnologias com menor utilização, no nível intermediário encontram-se as tecnologias com utilização moderada e no nível de base as tecnologias mais utilizadas.

QP_{1.3} Qual o índice de sucesso e falha?

Para iniciar a análise sobre as falhas encontradas nas *microtasks* adotou-se uma abordagem similar a de Mao et al. (2015). Nessa abordagem, os dados são adicionados em um vetor e computacionalmente classificados. A expectativa em aplicar essa abordagem foi verificar se as *microtasks* possuem algum tipo de descrição complementar para auxiliar a execução, evitando possíveis falhas. Para isso, o conjunto de dados foi computacionalmente classificado e as descrições das *microtasks* foram exploradas, identificando os mecanismos de suporte fornecidos pelo *crowdsourcer*.

A análise final demonstrou três diferentes características: utilização de links externos (geralmente para apresentar a documentação complementar), repositório (usado para armazenar uma parte ou a totalidade do projeto) e e-mail (geralmente o endereço eletrônico do *crowdsourcer*). Além disso, também foram detectadas descrições híbridas, que combinavam duas ou três características. Entretanto, uma parte considerável das *microtasks* não possuem nenhum tipo de mecanismo de suporte similar. A relação total é apresentada na Figura 15.

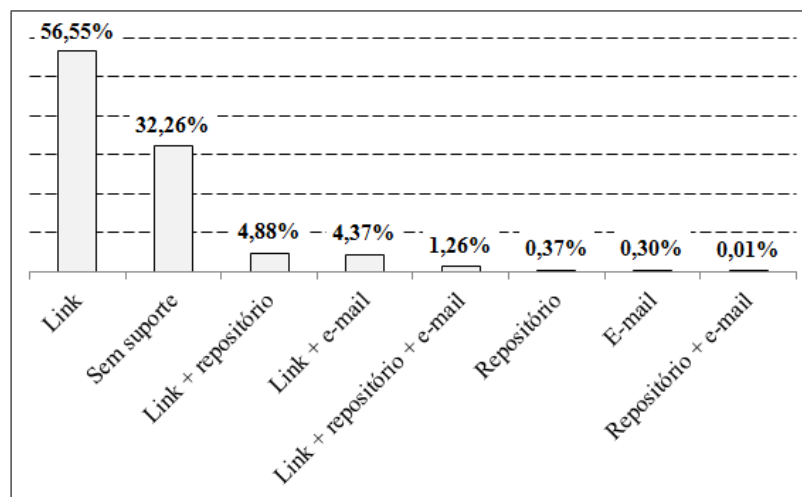


Figura 15: Os mecanismos para auxiliar a execução de *microtasks*

Fonte: Autoria Própria

Motivados pelo cenário em que quase um terço das *microtasks* não possuem nenhum tipo de auxílio externo identificado, foi analisado o índice de sucesso e falha de finalização de

microtasks. Inicialmente, percebeu-se que a maior parcela das *microtasks* são finalizadas com sucesso, entretanto, existe um contingente que demonstra diversas falhas. Ao todo foram filtrados seis tipos de falhas: **Sem submissões**, quando existe *crowd*, mas nenhum participante submeteu uma solução; **Cancelada pelo crowdsourcer**, quando o líder do projeto cancela arbitrariamente a *microtask*; **Cancelada na revisão**, a *microtask* foi finalizada, entretanto na revisão identificou-se alguma falha; **Impraticável**, *microtasks* consideradas com alta carga de complexidade; **Vencedor indiferente**, a *microtasks* foi finalizada, entretanto o participante responsável pela execução não procurou receber seu prêmio; e **Sem registro**, nenhum *crowd* foi formado. A Figura 16 exibe a porcentagem de finalização e falha das *microtasks*.

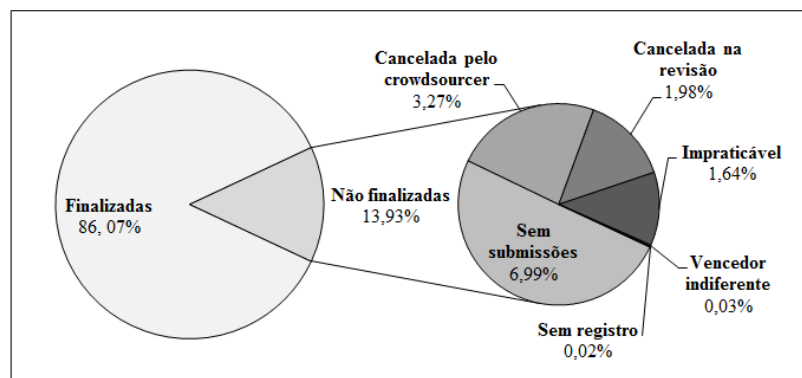


Figura 16: O índice de finalização e falha na execução de *microtasks*

Fonte: Autoria Própria

4.1.2 COMO A CONFIGURAÇÃO DO CROWDSOURCING AFETA AS MICROTASKS

Para conduzir as análises sobre como a configuração do CS afeta as *microtasks* foi necessário identificar e eliminar os *outliers* do caso analisado. Essa ação ocorreu devido a percepção de algumas *microtasks* concentraram um total de participantes dissonante da realidade. Os motivos que podem ter guiado esse crescimento de interesse na participação residiram no fato de tais *microtasks* fornecerem uma premiação acima da média do desenvolvimento CS ou pelo fato de organizações científicas estarem buscando soluções para os seus desafios. Assim, tendo em vista que a análise desta subseção reside na influência da configuração do CS sobre as *microtasks*, foi necessário remover dados que fossem considerados anormais, ou seja, situações que não ocorrem com frequência na maioria das *microtasks*. Além disso, também serviu para facilitar a visualização das informações realizadas durante a análise.

Com o objetivo de identificar os *outliers* foi aplicado um dos principais recursos disponíveis da estatística descritiva sugerido por Wohlin et al. (2000). Assim, foi utilizado o *box*

plot, tornando evidente que os *outliers* ocorreram a partir de *crowds* com mais de 180 participantes, como mostra a Figura 17. Logo, as *microtasks* do estudo de caso executadas por *crowds* com mais de 180 participantes foram removidas para as análises posteriores desta subseção.

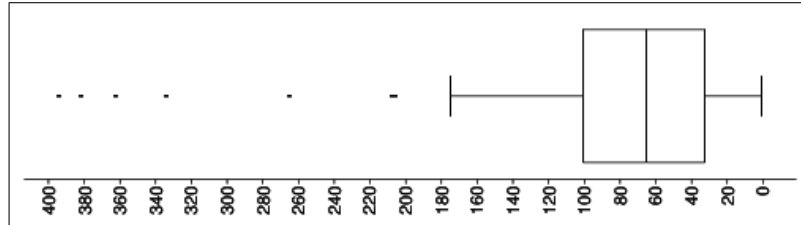


Figura 17: O tamanho dos *crowds* de forma detalhada

Fonte: Autoria Própria

A investigação conduzida nos *crowds* com até 180 participantes forneceu um conjunto de informações sobre a influência da configuração do CS nas *microtasks*. Percebeu-se que o CS torna o desenvolvimento de software via *microtasks* uma abordagem complexa e heterogênea. Mecanismos e presunções parecem não se comportarem de uma forma homogênea dentro da realidade do CS. Inicialmente, verificou-se que os *crowds* com menos participantes possuem uma tendência linear em dois sentidos:

- O aumento do tamanho do *crowd* também aumenta o índice de submissão, logo, quanto mais participantes registrados maior será o índice de submissão.
- O aumento do valor de uma *microtasks* também aumenta o tamanho do *crowd*, logo, *microtasks* com premiação mais alta retêm mais participantes.

Porém, esses exemplos apresentam um comportamento distinto nos *crowds* que concentram mais participantes. Tais *crowds* possuem uma complexa heterogeneidade de funcionamento e comportamento. As investigações conduzidas são apresentadas a seguir.

QP_{2.1}: *Qual a influência do tamanho do crowd?*

Os dados analisados permitiram investigar a influência do tamanho do *crowd* em três diferentes aspectos das *microtasks*: i) índice de submissão; ii) valor das premiações e iii) prazo de execução. Em relação ao índice de submissão, existe uma tendência inicial homogênea: conforme maior o total de registros aumenta o total de submissão. Nesse sentido, deve-se destacar os *crowds* com até 110 participantes concentraram uma taxa de submissão crescente. Contudo, os *crowds* mais populosos não demonstram tal comportamento. Foi verificado que os *crowds* entre 111-120 participantes possuíram uma média de submissão igual aos *crowds*

com 31-40 participantes. Além disso, os *crowds* que concentraram entre 131-170 participantes obtiveram uma queda de submissão, ou seja, o aumento da força de trabalho não refletiu no aumento de submissões. Essa relação é ilustrada na Figura 18.

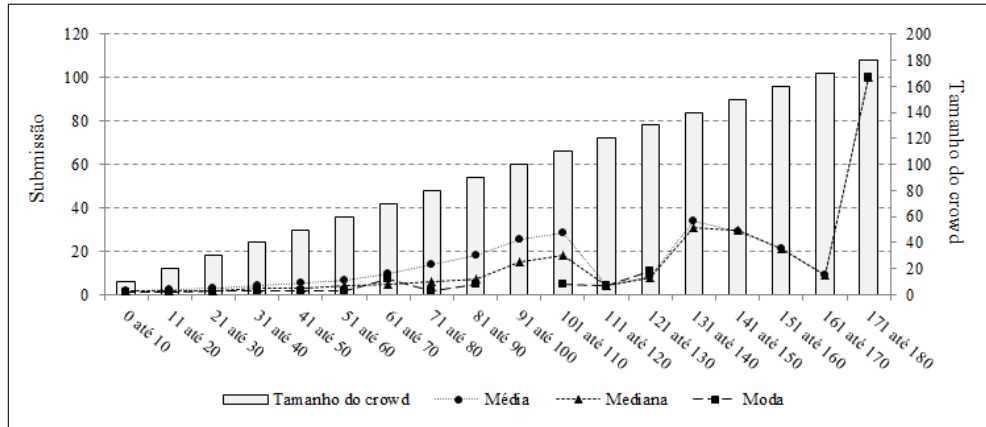


Figura 18: A influência das submissões nas *microtasks*

Fonte: Autoria Própria

Ao analisar o tamanho do *crowd* referente ao valor da premiação de uma *microtask* verifica-se um comportamento similar ao que ocorreu na submissão. Em *crowds* que concentraram até 40 participantes existiu um crescimento da premiação de uma *microtask* conforme houve aumento do tamanho do *crowd*. Entretanto, nos *crowds* entre 40-80 participantes o valor da premiação diminuiu. Enquanto que os *crowds* mais populosos seguiram uma distribuição completamente heterogênea. A Figura 19 apresenta essa relação.

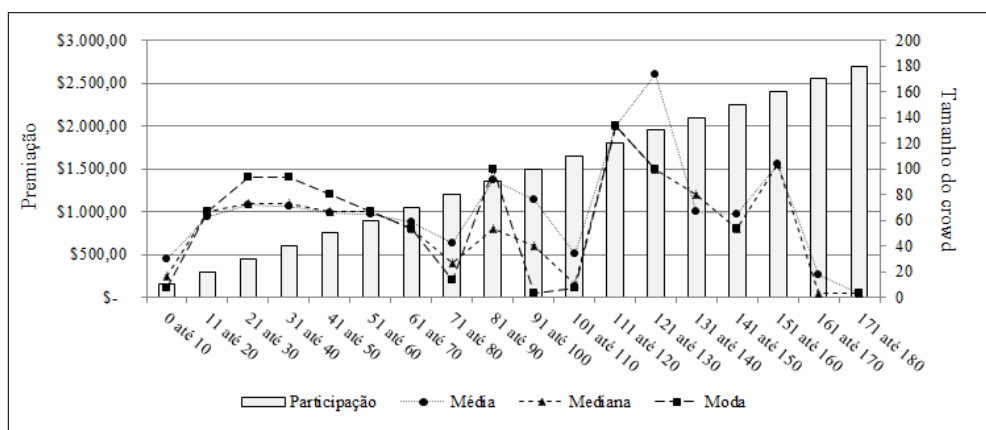


Figura 19: A influência da premiação nas *microtasks*

Fonte: Autoria Própria

Por fim, foi realizada a análise sobre a relação entre o prazo disponível para a execução de uma *microtask* e o tamanho do *crowd*. Nesse sentido, percebeu-se uma forte concentração

para execução de *microtasks* entre 2-3 dias. Esse prazo permaneceu com certa estabilidade nos *crowds* entre 21-110 participantes e em *crowds* entre 121-180 participantes. No entanto, os *crowds* formados por 0-20 participantes, 111-120 participantes e 161-170 participantes apresentaram uma distribuição variada. A Figura 20 apresenta a distribuição total entre o prazo disponível e o tamanho do *crowd*.

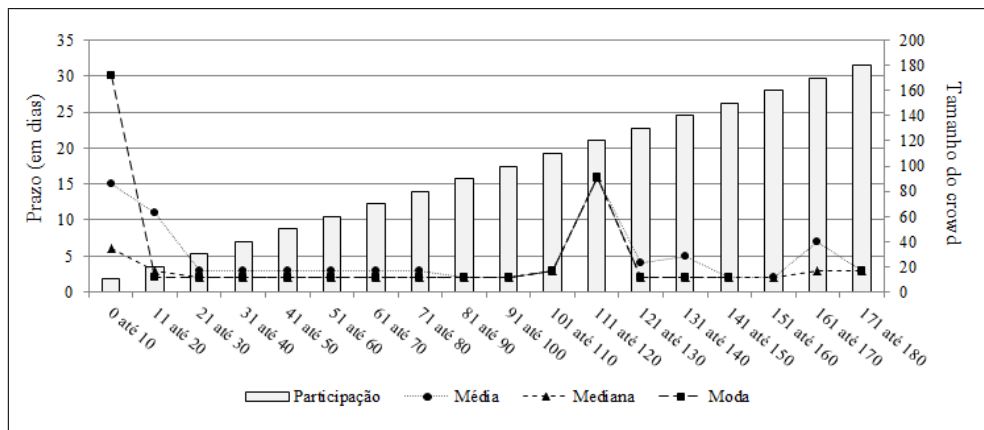


Figura 20: A influência dos dias disponíveis nas *microtasks*

Fonte: Autoria Própria

QP_{2.2}: Qual a complexidade dos projetos?

Segundo Saremi e Yang (2015b), o total de *microtasks* de um projeto CS pode denotar a sua própria complexidade. Logo, conforme os projetos CS concentram mais *microtasks*, esses possuem maior carga de complexidade. Baseado nesse contexto, o conjunto de dados foi analisado para identificar o total de *microtasks* presentes em cada projeto.

Ao todo foram identificados 1530 projetos distintos. Os menores projetos identificados concentraram apenas 1 *microtask* enquanto que o maior projeto possuiu mais de 1000 *microtasks* em sua composição. De um modo geral pode-se verificar que grande parte dos projetos CS utilizam um baixo número de *microtasks*, possuindo um conjunto de até 30 *microtasks* à medida que os projetos mais complexos somaram mais de 100 *microtasks*. A visão geral entre o total de *microtasks* e a porcentagem de incidência é encontrada na Figura 21.

Também foi analisada a complexidade dos projetos CS a partir do total de tecnologias empregadas em cada *microtask*. Nesse sentido, foi identificado que grande parte das *microtasks* contabilizaram apenas uma tecnologia. As *microtasks* que concentraram duas ou mais tecnologias apresentaram um índice decrescente, contudo nota-se um leve aumento em *microtasks* com 12 tecnologias. Outra constatação feita é sobre a variação do total de tecnologias, nessa perspectiva, nota-se um crescimento unitário de *microtasks* que concentram entre uma e 12 tec-

nologias. Entretanto, a partir desse total existem diferentes intervalos até alcançar um conjunto máximo com 28 tecnologias. A Figura 22 apresenta uma visão geral sobre o total de tecnologias empregadas nas *microtasks*.

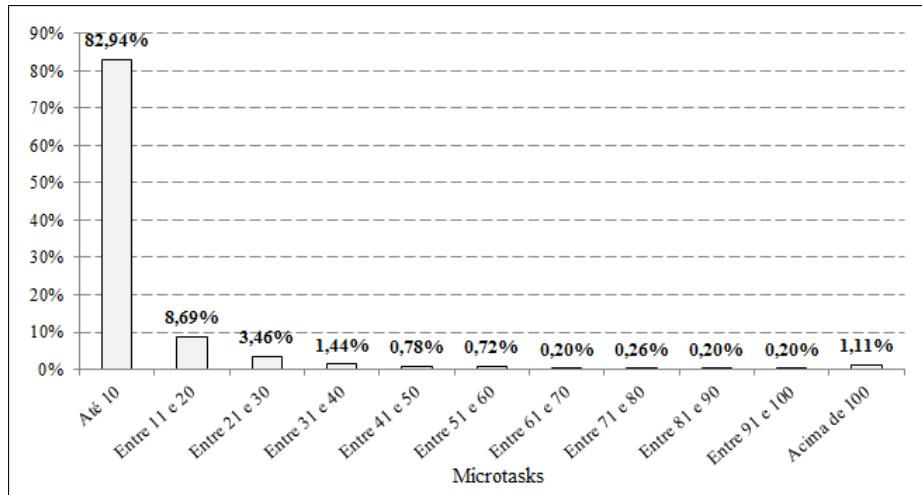


Figura 21: O total de *microtasks* dos projetos de software *crowdsourcing*

Fonte: Autoria Própria

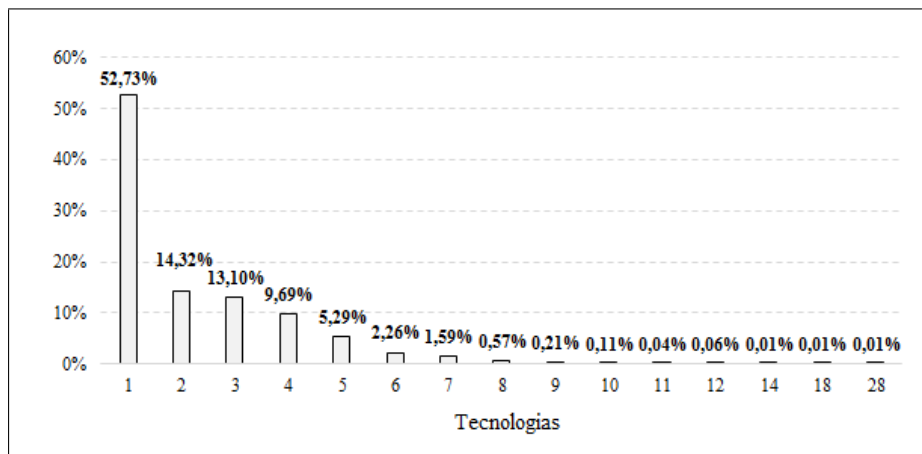


Figura 22: O total de tecnologias presentes nas *microtasks*

Fonte: Autoria Própria

4.1.3 RESUMO E DISCUSSÕES

Os resultados revelados pelo caso analisado forneceram um conjunto de considerações sobre o uso das *microtasks* no desenvolvimento de software CS. Inicialmente, se destaca a amplitude da análise, seja no total de *microtasks* (14485) ou no período investigado (cobrindo pra-

ticamente meia década de desenvolvimento). Assim, até o momento de escrita deste trabalho, este estudo de caso representa uma das pesquisas mais completas sobre o tema.

De maneira resumida, a análise da **QP₁** demonstrou a adaptabilidade das *microtasks* conforme o seu tipo de colaboração ou competição. Também foi evidenciado a sua aplicação em todas as etapas do ciclo de um projeto de software CS. As principais plataformas e tecnologias demonstraram a amplitude de aplicação que as *microtasks* possuem. Os mecanismos de auxílio, bem como o índice de sucesso e falha também foram discutidos e apresentados. A análise da **QP₂** investigou a influência das submissões, da premiação e do prazo das *microtasks*. Finalmente, a complexidade dos projetos de software CS com base nas *microtasks* e o total de tecnologias presentes em cada *microtask* também foi discutido.

Além disso, também foi revelado que as *microtasks* acabam gerando diferentes desafios para o desenvolvimento CS. Tais desafios se restringem a dois papéis principais do CS: *crowdsourcer* e *crowd*.

O *crowdsourcer* é a principal autoridade de um projeto CS. Ademais, é o responsável por cadastrar e fiscalizar a execução das *microtasks* na plataforma. Entretanto, durante o caso analisado, foram detectados exemplos de falhas das tarefas que são responsabilidade do *crowdsourcer*. Nesse sentido, deve-se destacar que diversas *microtasks* foram cadastradas em categorias erradas, com isso a formação do *crowd* podia ser comprometida. Também foram detectados casos em que o *crowdsourcer* não definiu a tecnologia necessária. Essa inexatidão podia fornecer maior liberdade de criação ao *crowd*, entretanto, também poderia apresentar resultados finais inconvenientes, como submissões inférteis. Ainda em relação ao *crowdsourcer*, cabe destacar que diversas *microtasks* não apresentaram nenhum mecanismo para auxiliar o seu desenvolvimento. Levando em consideração todo esse cenário desafiador, as *microtasks* representaram uma alta taxa de finalização, alcançando praticamente 87% de sucesso na finalização. Ou seja, grande parte das *microtasks* acabaram sendo entregues no prazo correto, dentro do custo estimado e com todos os requisitos completados. Esse panorama foi interessante, principalmente quando se coloca em perspectiva outros modelos de desenvolvimento de software. São comuns os casos de atividades e projetos que apresentam diversas falhas, sendo entregues fora do prazo ou com o orçamento maior do que o estimado. No desenvolvimento de software CS, apesar de haverem falhas desse sentido e desafios por parte do *crowdsourcer*, pode-se afirmar que o índice de sucesso ainda é elevado.

Em relação ao *crowd* - formado por participantes anônimos, com experiências variadas, e por vezes, não receberam nenhum apoio da liderança - também demonstrou um comportamento interessante. Como foi sumarizado brevemente por Naik (2016), o DDS e o *open*

source possuem diferentes aberturas de participação. Assim, conforme um projeto cresce é provável que isso também reflita no aumento de colaboradores. Todavia, esse comportamento não foi encontrado no CS. Como foi exposto no caso analisado, *crowds* populosos foram totalmente ineficazes, ao mesmo ponto que *crowds* menores demonstraram maior eficiência. Devido ao caráter dinâmico de cada *microtask* era completamente impreciso antecipar o tamanho que cada *crowd* iria alcançar. Caso o *crowdsourcer* não fiscalizasse ou estabelecesse mecanismos de controle, o desenvolvimento da *microtask* podia tornar-se inviável devido à amplitude do *crowd* e aos riscos inerentes dos participantes acessarem informações privilegiadas sem contribuir ao projeto.

Além dos desafios identificados, diferentes destaques e contribuições devem ser revisitados pela condução do estudo de caso. Primeiro, o fato de sistematizar as informações sobre as *microtasks* forneceu uma compreensão ampla sobre a sua aplicação e categorização. Foi necessário conduzir uma análise manual, investigando a descrição de um conjunto de *microtasks* para montar o quadro geral sobre quais as finalidades. Segundo, o caso analisado revelou que as *microtasks* suportam a execução de um projeto ou produto de software em qualquer etapa do ciclo de vida de desenvolvimento. Obviamente, deve-se destacar que desenvolver completamente um projeto ou produto de software via *microtasks* representa uma tarefa complexa, onerosa, e que irá absorver muito tempo de qualquer *crowdsourcer*. Contudo, a aplicação do CS em todas as etapas demonstram a portabilidade e o suporte que as *microtasks* fornecem. Assim, para tarefas pontuais, que necessitem da criatividade ou da força de trabalho, o uso das *microtasks* pode fornecer um grande apoio, principalmente pela redução do custo e rapidez de entrega. Terceiro, também foi interessante detectar que as *microtasks* fornecem apoio para etapas extremas do desenvolvimento de software, a Análise e a Integração. Na Análise existe grande atenção para identificar, planejar e estruturar os requisitos de um projeto. Além disso, diversas *microtasks* dedicadas à essa etapa do ciclo de vida foram identificadas. Enquanto que na Integração, apesar do suporte fornecido pela plataforma analisada, ainda possui uma utilização tímida das *microtasks*. Quarto e último, havia uma expectativa que a etapa de Implementação obtivesse grande utilização nas *microtasks*. Porém, tal etapa acabou ficando em desvantagem, principalmente pelo fato de grande parte das *microtasks* serem dedicadas ao *Design* e Teste de software.

Em resumo, o estudo de caso reportado nesta seção apresentou uma visão geral sobre o uso de *microtasks* no desenvolvimento de software CS. A experiência fornecida pelo caso analisado foi essencial para compreender a real natureza das *microtasks* e alicerçar o próximo pilar desta pesquisa. E em função disso, a próxima seção deste trabalho é dedicada a apresentar as características das *microtasks*.

4.2 CARACTERÍSTICAS

Para identificar quais são as características das *microtasks* foi necessário adotar duas abordagens distintas de investigação, o estudo de caso e o MS. Logo, foi necessário unir toda a experiência do caso analisado com as considerações detectadas na literatura nos diferentes estudos analisados para compreender qual a estrutura de uma *microtask*.

Durante a pesquisa do caso analisado tornou-se evidente a existência de diferentes configurações das *microtasks*, como o seu tipo de execução (colaborativa *x* competitiva). Assim, todas as informações que contribuíssem e revelassem as particularidades das *microtasks* foram extraídas e classificadas. Além disso, no MS, os estudos que forneceram considerações ou definições sobre as *microtasks* também foram agrupados e catalogados. Ao final do processo, todas as características foram organizadas e sistematizadas em uma taxonomia, coletando assim a compreensão sobre quais as porções que compõem uma *microtask*.

A função da taxonomia é sistematizar todo o conhecimento e conjunto de informação atualmente disponível sobre as características das *microtasks*. Obviamente, nesse sentido, algumas considerações devem ser evidenciadas. Inicialmente, o esforço em gerar uma taxonomia sobre as características das *microtasks* é dispendioso. Isso ocorre devido à dificuldade em generalizar os resultados obtidos. Logo, em uma visão realista, a taxonomia almejada representa o conjunto de características que todas, ou grande parte das *microtasks*, possuem. Contudo, é incerto garantir que esse objetivo será alcançado integralmente. Além disso, levando em consideração que a pesquisa foi baseada no MS, e atualmente existe um aumento de publicações sobre desenvolvimento de software CS e *microtasks*. Logo, acompanhar o rápido crescimento da literatura é uma tarefa complexa. Assim, existe uma necessidade de atualizar constantemente a taxonomia gerada.

A contribuição revelada pela construção da taxonomia representa uma unificação dos conceitos que os diversos estudos da área demonstraram sobre as *microtasks*. Além disso, também reúne toda a experiência fornecida pelo caso analisado, solidificando ampla gama de conhecimento literário. Para apresentar todas as contribuições obtidas nesse pilar da pesquisa, o restante do conteúdo desta seção refere-se à taxonomia que foi gerada, a estrutura de uma *microtask* e as considerações finais sobre as características identificadas.

4.2.1 TAXONOMIA

Ao todo foram identificadas treze características que as *microtasks* possuem, gerando assim a taxonomia. Desse conjunto, quatro foram detectadas pelo estudo de caso e as demais

via MS. Cada característica possui dois ou três estados de configuração possível. Algumas características foram referenciadas diversas vezes na literatura, como é o caso da finalidade de uma *microtask*. Ao momento que outras características, como a abertura que *microtask* possui para os participantes, foi referenciada apenas em um trabalho. Toda a estrutura da taxonomia é encontrada na Tabela 16.

Tabela 16: Taxonomia das características das *microtasks*

Característica	Fonte
1. Tipo	
1.1 Colaborativa	(Estudo de caso)
1.2 Competitiva	
2. Abertura	
2.1 Total	(Allahbakhsh et al. (2013))
2.2 Baseada na reputação	
2.3 Baseada na credencial	
3. Prazo	
3.1 Horas	(Estudo de caso)
3.2 Dias	
4. Granularidade	
4.1 Atomizada	Hosseini et al. (2014), Weidema et al. (2016), LaToza et al. (2014) e Stol e Fitzgerald (2014a)
4.2 Divisível	
5. Dependência	
5.1 Interna	
5.2 Externa	Stol e Fitzgerald (2014b) e Hosseini et al. (2014)
5.3 Nenhuma	
6. Sincronização	
6.1 Obrigatória	LaToza e Hoek (2016) e Tranquillini et al. (2015)
6.2 Não aplicável	
7. Limitação	
7.1 Possui	Tranquillini et al. (2015)
7.2 Não possui	
8. Expertise	
8.1 Determinada	Machado et al. (2016), Leano et al. (2016), LaToza e Hoek (2016), Yang e Saremi (2015), Zhao e Hoek (2015), Saremi e Yang (2015a), Zhao et al. (2015) e Tranquillini et al. (2015)
8.2 Criativa	
9. Tecnologia	
9.1 Definida	(Estudo de Caso)
9.2 Livre	
10. Contribuição	
10.1 Definida	Mao et al. (2017)
10.2 Indefinida	
11. Avaliação	
11.1 <i>Crowd</i>	
11.2 <i>Crowdsourcer</i>	(Estudo de Caso)
11.3 Integração	
12. Segurança	
12.1 Dados	
12.2 Intelectual	Mao et al. (2017), LaToza e Hoek (2016) e Bhatia et al. (2016)
12.3 Baixa	
13. Finalidade	
13.1 Prêmio	Mao et al. (2017), Afridi (2017), Saremi et al. (2017), LaToza e Hoek (2016), Machado et al. (2016), Yang et al. (2016), Alelyani e Yang (2016), Wang et al. (2016), Yang e Saremi (2015), Saremi e Yang (2015a, 2015b), Stol e Fitzgerald (2014b) e Mao et al. (2013)
13.2 Motivação	

Fonte: Autoria própria.

Nota: As características da taxonomia foram enumeradas apenas para facilitar a visualização e interpretação dos dados.

Cada característica possui uma finalidade e opera em uma porção da composição de

uma *microtasks*. Assim, para compreender efetivamente a sua organização, a seguir ocorre um aprofundamento sobre cada item da taxonomia:

- **Tipo:** refere-se ao modo que a *microtask* deverá ser executada pelo *crowd*. Nesse sentido, existem dois estados: colaborativa e competitiva. A opção colaborativa utiliza o poder do *crowd* como uma entidade única, enquanto que a opção competitiva busca a solução no menor tempo possível.
- **Abertura:** é relacionado ao modo como as *microtasks* estarão expostas ao *crowd*. Foram identificados três estados de abertura: total, baseada na reputação e baseada na credencial. Na abertura total todos os participantes cadastrados na plataforma podem executar a *microtask*. Na abertura baseada na reputação o *crowdsourcer* analisa o histórico do participante em *microtasks* anteriores. Finalmente, na abertura baseada na credencial somente os participantes credenciados pelo *crowdsourcer* podem executar a *microtask*.
- **Prazo:** refere-se ao tempo disponível em que uma *microtask* deverá ser executada. Foram identificados dois estados: horas e dias. As *microtasks* que possuem tempo disponível mensurado por horas, geralmente, possuem uma leve carga de dificuldade. Enquanto isso, as *microtasks* que possuem um ou mais dias de duração geralmente apresentam maior carga de trabalho.
- **Granularidade:** refere-se ao total de requisitos a serem cumpridos para a execução da *microtask*. Foram identificados dois estados de granularidade: atomizada e divisível. As *microtasks* com granularidade atomizada não podem ser divididas ou reduzidas. Enquanto que a granularidade divisível identifica *microtasks* que podem ser decompostas ou fracionadas em duas ou mais *microtasks*.
- **Dependência:** ocorre quando a manipulação de um artefato ou de uma *microtask* pode comprometer uma porção do projeto. A dependência pode ocorrer por meio de três estados: interna, externa ou nenhuma. A dependência interna ocorre quando a manipulação altera a própria *microtask*. Enquanto que a dependência externa modifica uma porção externa à *microtask*. As *microtasks* que não se coincidem com os estados anteriores não possuem nenhuma dependência.
- **Sincronização:** a sincronização refere-se ao modo como as *microtasks* estão conectadas por ordem de execução, servindo para alertar ao *crowd* ou ao *crowdsourcer* que uma *microtask* foi concluída e a sua subsequente pode ser executada. A sincronização pode assumir dois estados: obrigatória e não aplicável. A sincronização obrigatória ocorre

quando existe a necessidade de finalizar uma *microtask* para executar a subsequente, nas demais *microtasks* assume-se a sincronização como não aplicável.

- **Limitação:** determinadas *microtasks* podem requerer fatores limitantes, como mínimo de experiência, fluência em língua estrangeira, etc. A limitação pode ser considerada em dois estados: possui ou não possui. Logicamente, quando uma *microtask* apresenta qualquer tipo de limitação está é sinalizada como possui limitação. Em casos contrários, aplica-se que a *microtask* não possui nenhum tipo de limitação.
- **Expertise:** refere-se ao domínio de compreensão que os participantes do *crowd* devem possuir para realizar a execução da *microtask*. A expertise pode ser classificada por meio de dois estados: determinada ou criativa. Quando a expertise é determinada a *microtask* precisa de algum tipo de conhecimento específico. Em casos contrários, a expertise torna-se criativa.
- **Tecnologia:** representa o total de tecnologias necessárias para a execução das *microtasks*. As tecnologias podem assumir dois estados: definida ou livre. As *microtasks* com tecnologias definidas ocorrem quando o *crowdsourcer* especifica qual a tecnologia necessária para execução. Enquanto que nas *microtasks* com tecnologia livre a decisão cabe ao *crowd*.
- **Contribuição:** refere-se a dificuldade em identificar qual foi a porção que cada participante do *crowd* contribuiu para o projeto. Nesse sentido, foram identificados dois estados de contribuição entre as *microtasks*: definida e indefinida. As *microtasks* com contribuição definida necessitam identificar cada participante responsável, revelando o seu nível de contribuição. Enquanto que nas contribuições indefinidas a identificação da contribuição do participante é indiferente.
- **Avaliação:** representa a forma que uma *microtask* será avaliada para ser considerada concluída. Foram identificados três estados de avaliação: *crowd*, *crowdsourcer* e integração. Nas avaliações realizadas pelo *crowd* os próprios participantes sinalizam para a plataforma CS que a *microtask* foi executada e finalizada. Na avaliação pelo *crowdsourcer* o líder do projeto analisa e determina se a *microtask* foi executada e finalizada assertivamente. Finalmente, nas avaliações por integração logo após a *microtask* ser finalizada ela é automaticamente incorporada ao projeto e verifica-se o resultado final.
- **Segurança:** pode ser considerada como as restrições estratégicas que as *microtasks* podem possuir. Nesse sentido, identificaram-se três estados que a segurança pode assumir nas *microtasks*: dados, intelectual e baixa. A segurança de dados refere-se às *microtasks*

que possuem algum tipo de informação sensível, como registros em banco de dados. A segurança intelectual refere-se a um desafio em que uma solução apresentada por um participante do *crowd* não poderá ser compartilhada ou reutilizada. As *microtasks* que não se coincidem com as subcategorias anteriores precisam de pouca ou quase nenhuma relação de segurança.

- **Finalidade:** refere-se aos aspectos que tornam uma *microtask* atraente para o *crowd*. Nesse sentido foram identificados dois estados a serem consideradas: prêmio e motivação. O prêmio é o valor final concedido ao(s) participante(s) que finalizarem uma *microtask*. A motivação refere-se ao entusiasmo que cada participante pode possuir.

As características e estados de uma *microtasks* são apresentadas na Figura 23.

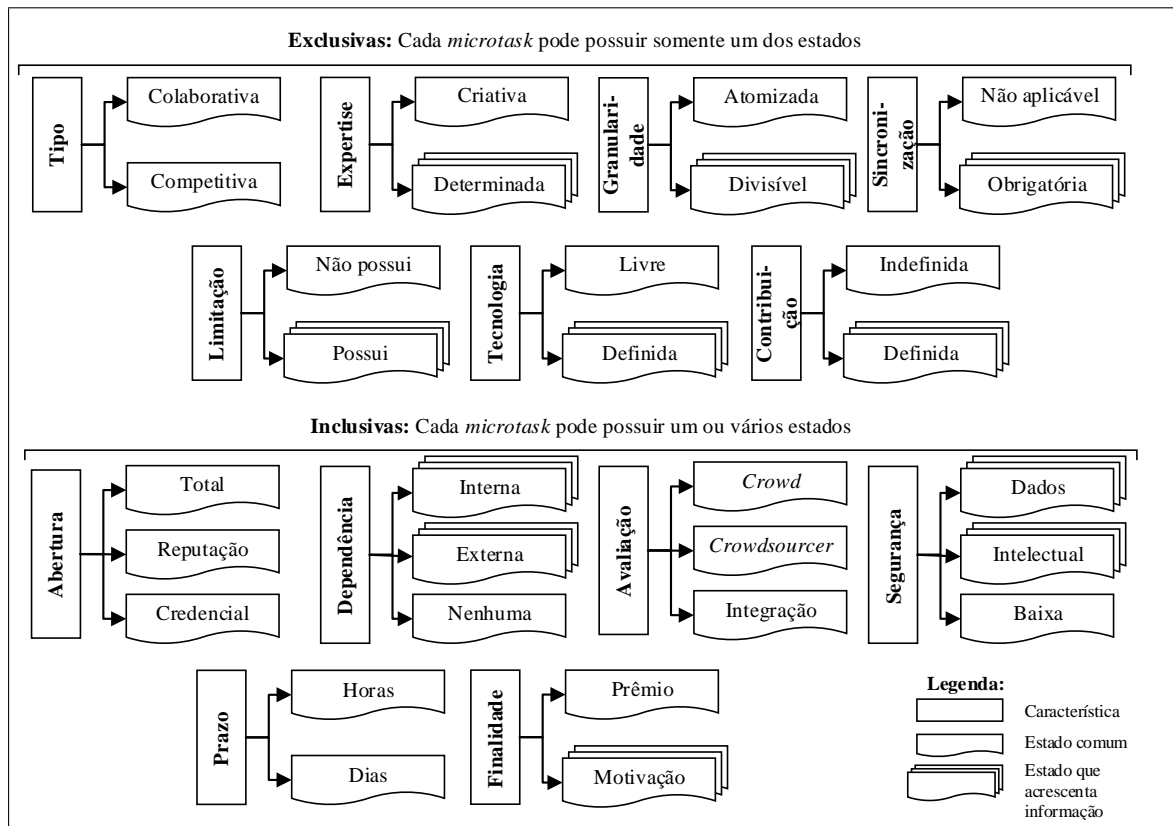


Figura 23: As características das *microtasks*

Fonte: Autoria Própria

A partir das análises efetuadas nos resultados do estudo de caso em conjunto com os autores consultados no MS identificou-se que as *microtasks* podem ser estruturadas em treze características distintas, que são fragmentadas por meio dos estados que pode assumir. Em síntese, os estados representam o conjunto de particularidades que cada *microtask* possui. E

nesse sentido, podem ser exclusivas (únicas) ou inclusivas (um ou vários). Existem algumas características que podem acrescentar informações conforme o estado assumido. Por exemplo, a **granularidade** quando é divisível pode aumentar conforme aumenta o total de requisitos. Similarmente, a **tecnologia** quando é definida representa que existe uma ou um conjunto de tecnologias necessárias para a execução. Assim, para demonstrar essa estrutura e os seus relacionamentos, foi gerado o metamodelo de uma *microtask*.

4.2.2 METAMODELO DE UMA *MICROTASK*

Para auxiliar a identificação das características e seus estados foi gerado um metamodelo de *microtask*, apresentado na Figura 24.

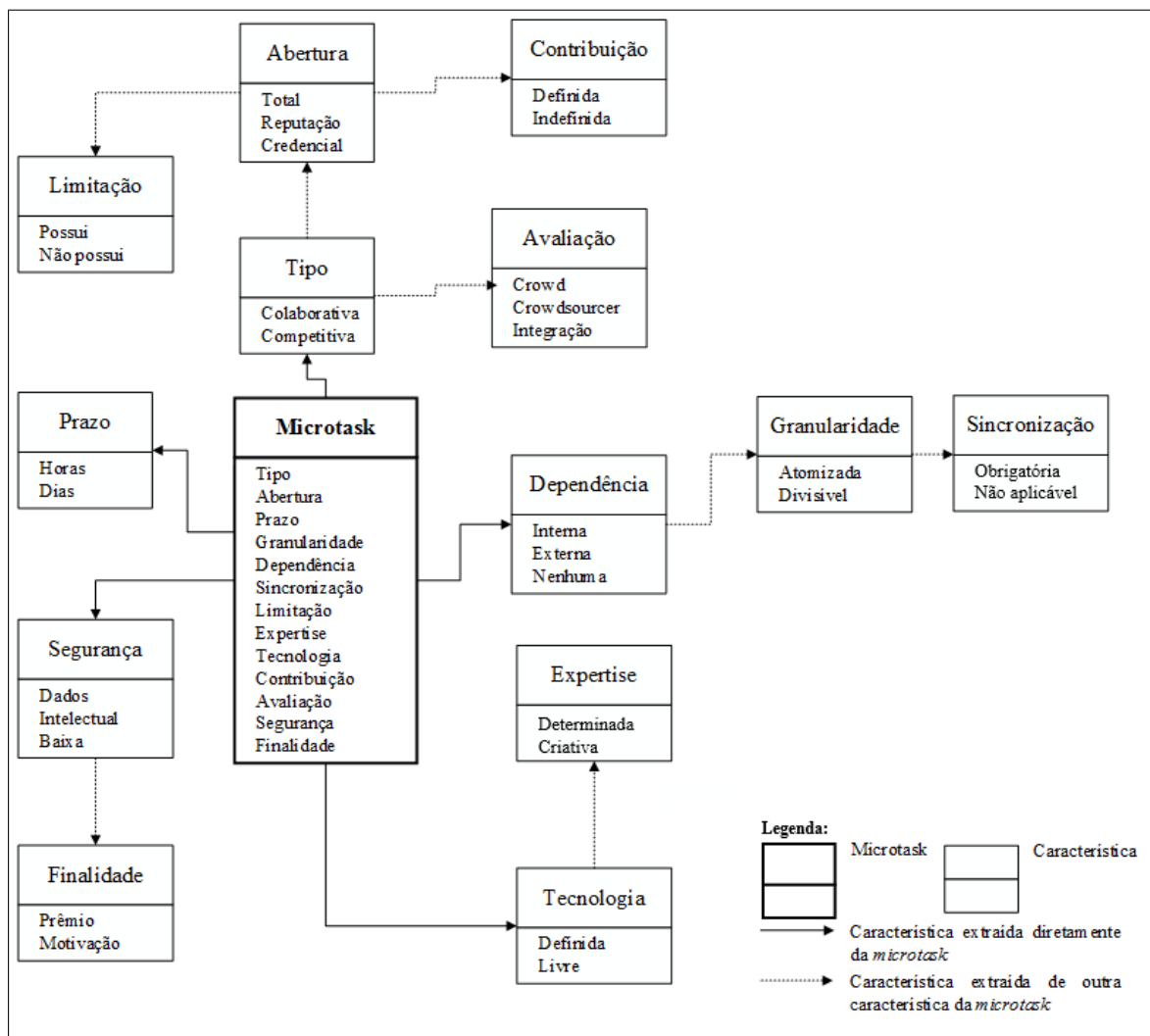


Figura 24: Metamodelo sobre características das *microtasks*

Fonte: Autoria Própria

O metamodelo pode ser aplicado para reduzir inconsistências, minimizar a carga de trabalho do *crowdsourcer* e auxiliar o *crowd*. Além disso, ele também apresenta uma simplificação e abstração das porções que compõem uma *microtask*. O núcleo do metamodelo representa a estrutura de uma *microtask*, contendo a definição de todas as suas características. Cada *microtask* possui um estado de execução (colaborativa ou competitiva) e a sua definição pode auxiliar as seguintes identificações:

- *Colaborativas*: as *microtasks* colaborativas tornam o *crowd* em uma entidade única. Dessa forma, essas *microtasks* podem apoiar a sua avaliação por meio do *crowd*, pois não existe nenhum tipo de competição entre os participantes. Ademais, *microtasks* colaborativas podem significar uma abertura total do *crowd*, buscando aumentar a amplitude da participação, dessa forma a limitação possivelmente não será estabelecida. Finalmente, em grande parte, as *microtasks* colaborativas não possuem contribuição definida visto que seu foco é na força de trabalho.
- *Competitivas*: as *microtasks* competitivas são centradas na velocidade de finalização. Logo, recomenda-se que essas *microtasks* sejam avaliadas por meio da integração e/ou *crowdsourcer*, evitando conflitos de interesses entre os participantes. Além disso, *microtasks* competitivas podem ter sua abertura realizada a partir de uma análise na reputação dos participantes, selecionando os que estejam mais alinhados com o propósito da *microtask*. Consequentemente, a abertura poderá influenciar nas limitações das *microtasks*. Por fim, *microtasks* competitivas pode possuir a contribuição definida, visto que apenas um conjunto limitado de participantes irão finalizá-la.

As dependências de uma *microtask* podem demonstrar o seu impacto na granularidade e influência na sincronização:

- *Interna e/ou externa*: *microtasks* com alguma dependência podem significar porções de trabalho que não estão completamente reduzidas. Dessa forma, a granularidade provavelmente será divisível. Ademais, as *microtasks* com dependências também demonstram indícios de sincronizações obrigatórias, pois possuem algum tipo de conexão.
- *Nenhuma*: as *microtasks* que não possuem nenhum tipo de dependência demonstram que sua granularidade, provavelmente, está atomizada. Com isso, a sincronização poderá não ser obrigatória.

A identificação das tecnologias de uma *microtask* alerta para a seleção das expertises dos participantes:

- *Definida*: as *microtasks* com tecnologias definidas geralmente estão relacionadas em algum domínio das expertises dos participantes. Dessa forma, ao definir uma tecnologia torna-se mais visível a necessidade de vincular e identificar a sua expertise.
- *Livre*: as *microtasks* com tecnologias livres geralmente não demandam a compreensão, e dessa forma a expertise pode ser identificada como criativa.

O nível de segurança que uma *microtask* possui pode auxiliar a identificação da sua premiação, bem como os estímulos de motivação:

- *Dados e/ou Intelectual*: as *microtasks* que exigem algum nível segurança podem indiciar uma relação com a sua finalidade. Conforme aumenta o nível da segurança de dados e da segurança intelectual pode indicar uma relação direta com a premiação final da *microtask*. Além disso, determinados níveis de segurança podem atrair diferentes perfis de participantes.
- *Baixa*: *microtasks* com nível baixo de segurança, provavelmente, terão a sua premiação final menor que as *microtasks* que exigem níveis maiores de segurança.

Ao finalizar a seleção de todas as características e estados, a definição do prazo para *horas* ou *dias* pode ser facilitada. Isso decorre em virtude do conjunto de informação revelada por meio dos estados anteriormente definidas.

4.2.3 RESUMO E DISCUSSÕES

As características identificadas revelaram uma abrangente compreensão sobre as porções e a estrutura que uma *microtask* possui. Até o momento da escrita deste trabalho, a taxonomia que foi apresentada sobre as *microtasks* representa o estudo mais amplo e completo sobre o tema disponível na literatura.

Em uma síntese geral, as *microtasks* possuem treze características distintas. Cada característica apresenta dois ou três estados de configuração. Além disso, existem características **exclusivas**, quando os estados somente assumem uma configuração possível (como o tipo de execução, que sempre será competitivo ou colaborativo); e existem também as características **inclusivas**, em que os estados podem assumir diversas configurações (como a avaliação, que pode ocorrer via *crowd*, *crowdsourcer* ou por meio da integração). Ademais, existem alguns estados que possuem um conjunto maior de informação, como a expertise, que ao assumir o

estado “determinada” pode possuir uma ou várias expertises relacionadas para a execução da *microtask*.

Também foi possível identificar que existem características que estão diretamente relacionadas com outras, e assim foi possível gerar o metamodelo de uma *microtask*. O núcleo do metamodelo apresenta as treze características catalogadas e é contornado pelas suas ligações. Assim, torna-se prático e explícito a estrutura e as ligações de uma *microtask*. A exemplo, ao definir que uma *microtask* assume uma tecnologia “definida” automaticamente se estabelece a expertise relacionada.

As contribuições fornecidas referem-se a diferentes vertentes da literatura, e nesse sentido, pode-se destacar três importantes colaborações: Primeiro, como foi exposto na Seção 4.1 deste trabalho, as *microtasks* possuem uma amplitude de uso muito grande, desde a análise até a integração de um projeto CS. Logo, compreender os componentes semelhantes de *microtasks* com finalidades tão distintas representam uma valiosa contribuição. Segundo, existe uma escassez de estudos concentrados nas características das *microtasks*. Assim sendo, a taxonomia exposta contribuiu para diminuir os desafios gerados por tal lacuna. Terceiro, o metamodelo apresentou uma ilustração sobre as ligações das porções que compõem uma *microtask*. Assim, torna-se fácil e prático compreender de fato qual a estrutura de uma *microtask*.

Com a conclusão do segundo pilar dessa pesquisa, acredita-se que o objetivo de identificar as características que todas, ou grande parte das *microtasks* possuem, foi alcançado com êxito. A abordagem adotada, mesclando estudo de caso e MS, serviu de base e forneceu uma visão plena sobre o atual panorama das *microtasks*.

Em resumo, pode-se dizer que as *microtasks* apesar de possuírem um conceito de minimização ou atomização do trabalho, possuem também uma estrutura complexa formada por um conjunto de características e diferentes estados de configuração. Com essa percepção, diversos questionamentos emergiram, principalmente sobre como essa estrutura se comporta perante outras atividades durante o desenvolvimento de software.

Desse modo, após compreender quais são as características que as *microtasks* possuem, foi possível avançar para o próximo pilar desta pesquisa que investigou o comportamento das *microtasks* frente as atividades de software tradicionais. Assim sendo, o foco da próxima seção é apresentar quais são as diferenças em utilizar *microtasks* ou atividades tradicionais no desenvolvimento de um produto ou projeto de software.

4.3 CONTRASTES

Para identificar quais as diferenças entre as *microtasks* com as atividades de software foi necessário adotar uma abordagem mesclando o estudo de caso com o MS. Em síntese, o objetivo almejado foi investigar como as *microtasks* que são executadas no desenvolvimento de software CS se diferem das atividades de software adotadas em um modelo tradicional de desenvolvimento.

A abordagem adotada para executar a comparação foi similar à utilizada na identificação das características das *microtasks*. Assim, a experiência fornecida pelo caso analisado foi unificada com as considerações dos estudos do MS para detectar como as características se comportam durante a execução das *microtasks*. Porém, além disso, a literatura foi consultada novamente para identificar como cada característica de uma *microtask* se comporta durante a execução de uma atividade de software. Assim, diversos trabalhos foram analisados e as considerações foram organizadas e sistematizadas.

A finalidade da comparação é avançar o atual estado da arte sobre as *microtasks*. As principais contribuições proporcionadas pela comparação é identificar, efetivamente, como as *microtasks* se diferenciam das atividades tradicionais de desenvolvimento. Apresentando assim, os benefícios e desafios da sua execução. O restante do conteúdo desta seção relata qual a definição de uma atividade de software, a seleção o modelo de desenvolvimento, os resultados obtidos e as considerações finais sobre a comparação de uma *microtask* e de uma atividade de software.

4.3.1 DEFINIÇÃO ADOTADA

A primeira etapa para identificar os contrastes entre as *microtasks* e as atividades tradicionalmente empregadas no desenvolvimento de software, foi definir o que é uma atividade de software. E tendo em vista o cenário literário atual, recorreu-se ao padrão proposto pelo *Institute of Electrical and Electronic Engineers* sobre desenvolvimento de software¹ para estabelecer uma definição formal. Assim, uma atividade de software foi definida como qualquer tipo de tarefa executada durante o desenvolvimento de um produto de software e que pode ser classificada como pertencente ao *requisito*, *design* ou *implementação* desse mesmo produto. Sendo que os *requisitos* compreendem, definem e estruturaram as exigências do produto. O *design* apresenta de forma coerente a representação dos requisitos. E a *implementação* compre-

¹Disponível em: <https://web.archive.org/web/20180112213104/http://ieeexplore.ieee.org/document/741936/> acesso em 18 dez. 2017

ende toda a interação necessária com qualquer tipo de linguagem de programação do produto.

Deve-se enfatizar que este trabalho adotou o termo “*microtask*” para representar as atividades executadas no desenvolvimento de software CS. Contudo, pode-se admitir que as *microtasks* partilham em grande parte da definição anteriormente apresentada quando a sua execução ocorre por meio do CS. Tendo em vista a definição de uma atividade de software e uma *microtask*, foi iniciado a etapa de seleção de modelos e papéis a serem analisados.

4.3.2 SELEÇÃO DO MODELO DE DESENVOLVIMENTO E DEFINIÇÃO DOS PAPÉIS

Considerar uma comparação somente entre as atividades de software com as *microtasks* poderia comprometer a análise, visto que as *microtasks* sofrem influências do CS, enquanto que as atividades de software podem sofrer influências distintas conforme o modelo de desenvolvimento adotado (NAIK, 2016). Logo, além de considerar a definição de uma atividade de software, também foi necessário selecionar um modelo de desenvolvimento. Para isso, foram utilizadas as seguintes perspectivas: primeiro, o modelo selecionado deveria executar atividades de software como foi definido na subseção anterior deste trabalho; segundo, o modelo deveria ter semelhanças com o CS, visto que seu distanciamento resultaria em uma comparação frágil e vaga, pois estariam em perspectivas modelos com muitas diferenças. Baseado nesse panorama, optou-se pela seleção do modelo DDS.

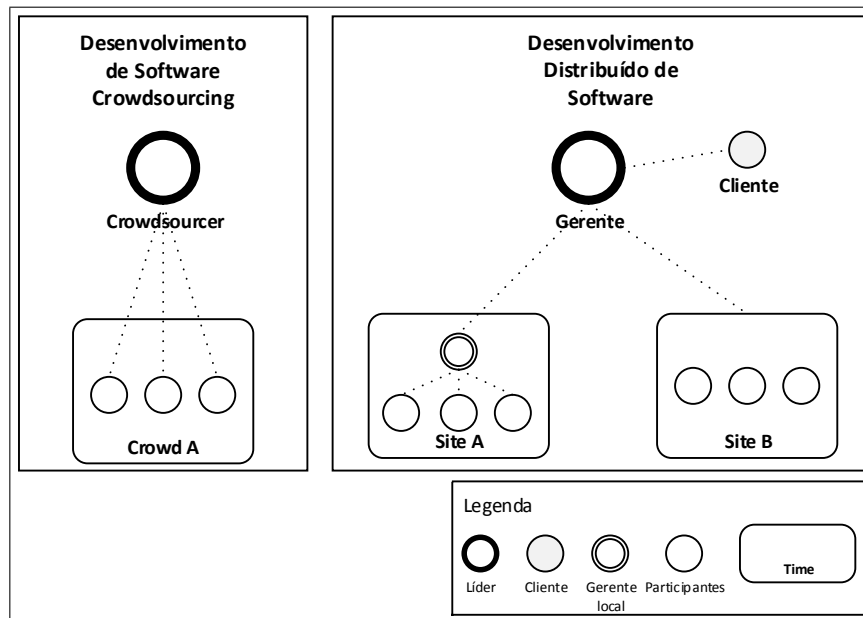


Figura 25: Comparação dos modelos de desenvolvimento de software *crowdsourcing* e distribuído

Fonte: Adaptado de Mahmood et al. (2017).

Em síntese, pode-se dizer que o DDS ocorre quando um projeto de software é particio-

nado e possui as suas atividades distribuídas em diferentes *sites* (equipes de desenvolvimento). Contudo, foram seguidas as considerações de Mahmood et al. (2017) sobre DDS, que estabelece duas importantes características: i) o DDS possui um líder responsável pelo projeto e pela comunicação com o cliente; ii) o líder pode comunicar-se diretamente com um *site* do projeto ou com o líder local desse *site*. O desenvolvimento de software CS se assemelha com essas considerações, todavia, o cliente é o próprio *crowdsourcer*, que comunica-se diretamente com a sua equipe de desenvolvimento (*crowd*). Para exemplificar esses conceitos, a Figura 25 ilustrou uma comparação entre os dois modelos.

Antes de realizar a comparação, também se deve destacar que foram adotados três papéis análogos entre o desenvolvimento de software CS e DDS: **Líder**, autoridade máxima do projeto, sendo considerado o gerente (DDS) e o *crowdsourcer* (CS); **Time**, equipe de desenvolvimento, designada como *site* (DDS) e *crowd* (CS); e **Processo**, obrigações do líder ou time, definidos como processo CS e processo DDS.

Finalmente, portando a definição de atividade de software e a seleção de um modelo de desenvolvimento e dos seus papéis, a comparação pode ser executada. A seguir, são encontrados os resultados gerados.

4.3.3 OS CONTRASTES IDENTIFICADOS

Foram identificados diversos contrastes entre as *microtasks* e as atividades de software DDS no ponto de vista do líder, do time e do processo durante a comparação. O que se segue representa uma síntese sobre os principais aspectos notados:

- **Tipo:** O CS permite que as *microtasks* possam assumir os estados colaborativos ou competitivos, estimulando a criação em grupo ou a rapidez de execução. As atividades de software são restritas à execução de forma colaborativa (BICK et al., 2017). Sendo assim, geralmente, cada *site* desenvolve uma porção distinta de um projeto.
 1. *Líder:* Do ponto de vista do *crowdsourcer* as *microtasks* permitem maior amplitude de trabalho. Em contrapartida, o uso do CS colaborativo pode representar um desafio para identificar a contribuição de cada participante. Do ponto de vista do gerente, as atividades colaborativas podem ser rapidamente alocadas conforme a especialidade de cada *site*. Ademais, fica transparente ao gerente qual a porção de trabalho que cada *site* está executando.
 2. *Time:* As *microtasks* competitivas podem tornar o *crowd* uma entidade veloz na busca de uma solução. Enquanto que as *microtasks* colaborativas e atividades de

software podem estimular o trabalho em equipe.

3. *Processo*: As *microtasks* competitivas oneram o *crowdsourcer* quando existe a necessidade de identificar as contribuições de cada participante. Enquanto que as *microtasks* e atividades colaborativas podem representar resultados rápidos e a possibilidade do trabalho em equipe.

- **Abertura**: O CS permite até três formas de abertura de participação nas *microtasks*. Com isso, pode-se realizar uma filtragem conforme a necessidade de execução. Em DDS, além do gerente e dos *sites*, o cliente também pode aderir a participação do projeto, e dessa forma influenciar o gerenciamento (NIAZI et al., 2016).

1. *Líder*: O *crowdsourcer* precisa traçar uma delicada estratégia para a abertura de uma *microtask*. Conforme maior o nível de filtragem de abertura, teoricamente, haverá um número menor de participação, porém com maior aptidão. Enquanto isso, nos projetos DDS a abertura é dedicada à totalidade do *site*, nesse sentido, todos os participantes devem executar alguma porção de trabalho. Porém, deve-se compreender que determinadas atividades de software poderão possuir abertura ao cliente, como atividades de *requisitos* ou *design*.
2. *Time*: Pode-se dizer que a abertura influencia a formação do *crowd*, visto que os participantes que não obtiverem os requisitos solicitados não devem compor o *crowd*. Em DDS, existem duas possibilidades: os participantes do *site* decidem quais atividades executarão, ou a decisão recai sobre o gerente/gerente local.
3. *Processo*: A abertura onera o processo de trabalho do *crowdsourcer* devido à verificação que cada participante deve receber. Em DDS, o gerente possui uma visão mais transparente, além de poder determinar quais participantes executarão as atividades.

- **Prazo**: As *microtasks* possuem o prazo definido pelo *crowdsourcer*. Devido a participação anônima, o uso de mecanismos auxiliares (históricos, *logs*...) tornam-se restritos. E com isso, tal tarefa se baseia em uma perspectiva empírica. O DDS fornece mecanismos para auxiliar a definição do prazo, de acordo com Bick et al. (2017) essa atividade pode ser organizada pela união entre gerente, *sites* e experiências passadas.

1. *Líder*: O *crowdsourcer* deve definir o prazo de execução de uma *microtask* baseado unicamente na sua perspectiva. Nas atividades DDS o prazo final ocorre mediante um entendimento comum entre gerente e *site*. Dessa forma, devido à experiência dos participantes, os prazos podem possuir uma margem de segurança maior em DDS do que em CS.

2. *Time*: O *crowd* é obrigado a seguir o prazo estipulado pelo *crowdsourcer*. Obviamente, cada participante pode decidir abandonar a *microtask* sem nenhum tipo de multa ou penalidade. Os *sites* possuem maior facilidade no que tange o prazo de execução de uma atividade, visto que a sua definição é realizada com o auxílio do gerente. Reuniões regulares podem atestar o avanço ou atraso do projeto, e assim, mecanismos mitigadores podem ser adotados.
 3. *Processo*: O prazo torna o processo mais oneroso em projetos DDS. Isso ocorre devido à necessidade de reuniões, elaboração e correção de cronogramas, fiscalização do andamento do projeto, entre outras fiscalizações. Em CS, após a definição do prazo ocorre a espera pela finalização da *microtask*. Em caso de sucesso, o(s) participante(s) é(são) premiado(s). Em caso de falha, a *microtask* pode ser reestruturada.
- **Granularidade**: A granularidade apresenta grande distinção entre CS e DDS. As *microtasks* geralmente são porções de trabalho menores que as atividades. Essa divisão ocorre para tornar o CS executável. Em DDS, devido à estrutura dos projetos, as atividades podem suportar diversas granularidades (MAHMOOD et al., 2017).
 1. *Líder*: O único responsável pela granularidade de uma *microtask* é o *crowdsourcer*. A atomização ou divisibilidade dependem unicamente da sua concepção. Em DDS, o gerente auxilia a definição da granularidade.
 2. *Time*: O *crowd* não possui muitos mecanismos a disposição para identificar a granularidade de uma *microtask*. Nesse cenário, algumas *microtasks* podem necessitar de um esforço distinto ao presumido em sua descrição. Os *sites* possuem métodos estratégicos para identificar a granularidade, como reuniões, métricas e históricos, e assim, o esforço torna-se mensurável.
 3. *Processo*: O processo de uma *microtask* apresenta uma carga de trabalho direcionada ao *crowdsourcer* e *crowd*. Com isso, a granularidade definida de forma equivocada pelo *crowdsourcer* pode impossibilitar a execução de uma *microtask* pelo *crowd*. Em DDS, o processo pode ser considerado mais oneroso, entretanto é dividido entre o gerente e os *sites*.
 - **Dependência**: A dependência apresenta similaridade entre as *microtasks* e as atividades DDS. Como aborda Bick et al. (2017), o DDS pode empregar dependências entre atividades que encontram-se em *sites* distintos. Assim, podem emergir desafios como a priorização ou alocação entre atividades dos *sites*. Esse cenário também ocorre no CS, logo podem existir *microtasks* que alteram o trabalho de outro *crowd*.

1. *Líder*: O *crowdsourcer* e o gerente devem atender e avaliar todas as dependências. Nesse cenário, pode-se utilizar um mecanismo como a “sincronização” para evitar que uma *microtask* ou uma atividade DDS dificulte outra porção de trabalho.
 2. *Time*: Cada *crowd* é independente, assim sendo, os possíveis transtornos causados por dependências das *microtasks* são mitigados pelo *crowdsourcer* do projeto. Em DDS o cenário é diferente, e cada *site* pode ajudar a solucionar e contornar cenários de dependências.
 3. *Processo*: O processo de uma *microtask* também onera a carga de trabalho do *crowdsourcer* devido a necessidade de acompanhamento e correções. Os gerentes possuem maior visibilidade no que tange a execução do processo de uma atividade DDS devido ao apoio fornecido pelos *sites* do projeto.
- **Sincronização**: A sincronização pode representar um mecanismo para evitar erros causados pela dependência. Em CS, as *microtasks* podem ser sequencialmente ligadas, e somente após a sua conclusão e aprovação deve-se iniciar a *microtask* subsequente. Em DDS, pode-se utilizar práticas como reuniões ou planejamento estratégico a fim de definir a sincronização e a alocação de atividades (BICK et al., 2017) .
 1. *Líder*: O *crowdsourcer* e o gerente possuem diferentes níveis de atribuições para estabelecer a sincronização entre *microtask*/atividades DDS. Essa tarefa é única e exclusiva do *crowdsourcer*, enquanto que em DDS a sincronização pode ser definida entre o gerente e os *sites*.
 2. *Time*: Em DDS e CS o time possui aspectos similares referente à sincronização das atividades/*microtasks*. De um modo geral, somente após a finalização de uma *microtask* o *crowd* emite um alerta ao *crowdsourcer*. Comportamento similar ao DDS, contudo, nas atividades de software um participante também pode emitir alerta para os demais participantes do *site*.
 3. *Processo*: O processo das atividades é mais complexo em DDS. Isso ocorre devido à possibilidade de alertas entre participante-*site* e *site*-gerente. Em CS tende a ser mais simples, pois após definida a sincronização pelo *crowdsourcer*, o *crowd* apenas pode emitir alertas para o líder e não para outros participantes.
 - **Limitação**: As limitações das atividades DDS estão diretamente relacionadas com as limitações da própria abordagem distribuída de software. Nesse sentido, o DDS apresenta um cenário amplo que pode envolver desde o fuso horário até as regulamentações do governo local, entre outros aspectos (MAHMOOD et al., 2017). Ademais, as limitações

das atividades DDS devem ser solucionadas com a participação dos *sites*, gerente e cliente do projeto. Em CS as limitações estão restritas a concepção do *crowdsourcer*.

1. *Líder*: Devido a distribuição global e o contrato estabelecido, o gerente deve respeitar a legislação que cada participante está submetido. Isso pode influenciar diversos fatores, como o custo de operação das atividades de software. Assim, o DDS emprega automaticamente em suas atividades um conjunto de fatores limitativos, que são acompanhados pelo gerente. O *crowdsourcer* não possui tal restrição, principalmente pela ausência de um contrato físico. Logo, as limitações das *microtasks* podem partir internamente de seu comando, e não externamente, como em DDS.
 2. *Time*: Os *sites* também estão sob as limitações impostas pela legislação ou qualquer outra entidade. Assim, o desenvolvimento das atividades deve obedecer ao conjunto de regras expressas. Em CS, as *microtasks* podem apresentar restrições para a formação do *crowd*, porém, tais restrições dificilmente ocorrem durante a execução da *microtask*.
 3. *Processo*: Devido ao conjunto de limitações que uma atividade DDS pode possuir, o seu processo é mais complexo que nas *microtasks* CS. Isso decorre em virtude das *microtasks*, tradicionalmente, apresentarem algum tipo de limitação baseada na perspectiva do *crowdsourcer*. Em DDS as limitações podem ser encontradas antes da execução, durante e após o encerramento da atividade.
- **Expertise**: A expertise está presente nas *microtasks* e nas atividades de software, porém existem diferentes relações. No CS a expertise está associada ao que o *crowdsourcer* define necessário para a execução de uma *microtask*. Em DDS, o próprio *site* pode definir a sua expertise para auxiliar a distribuição de atividades (MAHMOOD et al., 2017).
 1. *Líder*: O *crowdsourcer* pode definir as expertises para facilitar a formação de um *crowd* capaz de executar a *microtask*. Entretanto, a definição fica restrita ao líder do projeto CS. As atividades DDS podem apresentar maior facilidade de distribuição devido ao apoio que os *sites* fornecem ao gerente do projeto.
 2. *Time*: O *crowd* está totalmente submetido as expertises delineadas. Sendo assim, não existem muitos mecanismos para que os participantes possam requerer algum tipo de alteração. Os *sites* são mais participativos, e podem apresentar ao gerente a solicitação de determinadas atividades devido a sua especialização.
 3. *Processo*: O processo das *microtasks* é restritivo para a seleção de expertises do *crowd*. Enquanto que em DDS, existe troca de conhecimento e comunicação direta entre o gerente e os *sites*.

- **Tecnologia:** A tecnologia de uma *microtask* pode assumir o estado “definida” ou “livre”, facilitando trabalhos predeterminados ou criativos. Ademais, devido a dispersão do *crowdsourcer* e do *crowd* pode existir maior liberdade de criação. Porém, em DDS, as atividades criativas podem reter algum tipo de obstrução indireta. Como foi abordado por Hoda et al. (2017), o ensino e aprendizado de DDS para estudantes apresenta aspectos de influência sociocultural, como os hábitos de trabalho ou a influência do gerente de projeto. Sendo assim, as atividades DDS podem possuir indiretamente um fluxo de trabalho e de uso de tecnologias preestabelecidas mesmo quando são criativas.
 1. *Líder:* As tecnologias das *microtasks* variam conforme a concepção do *crowdsourcer*. No caso de definição, deve-se identificar o conjunto de tecnologias necessárias. Em caso contrário, quem define as tecnologias necessárias é o próprio *crowd*. Enquanto que em DDS, o gerente pode ter essa tarefa auxiliada pelos *sites* ou cliente.
 2. *Time:* O *crowd* e os *sites* podem estar submetidos ou serem autônomos para o uso de tecnologias. Entretanto, as atividades DDS com tecnologia livre podem representar um gargalo específico. Os fatores socioculturais podem desmotivar participantes a inovarem durante a criação, e assim, aplicar somente tecnologias comumente utilizadas pelo *site* em atividades similares.
 3. *Processo:* O processo para seleção de tecnologias nas *microtasks* e nas atividades DDS representam diferenças. Em CS o papel encarregado pela seleção de tecnologia é o *crowdsourcer*. Porém, caso opte por “tecnologia livre” o *crowdsourcer* ficará submetido à solução proposta pelo *crowd*. Em DDS existe maior transparência do processo devido a comunicação e o relacionamento do gerente com o *site* ou cliente.

- **Contribuição:** Identificar a parcela e o nível de contribuição de cada participante em uma *microtask* pode representar um grande desafio, influenciando no aumento de carga para o *crowdsourcer*. Entretanto, as atividades DDS não possuem essa dificuldade, visto que a contribuição de uma atividade, em grande parte dos projetos DDS, é relacionada a um *site* e não à um participante (BICK et al., 2017).
 1. *Líder:* A identificação das contribuições de uma *microtask* são fiscalizadas pelo *crowdsourcer*. Nesse sentido, conforme aumenta o total de submissão, torna-se oneroso a identificação da contribuição individual. Enquanto que no DDS essa função pode ser simplificada durante a análise do projeto, e as atividades podem ser previamente definidas e alocadas para *sites* distintos. E com isso, o gerente pode ter uma prévia do que esperar da contribuição de cada *site*.

2. *Time*: O *crowd* pode representar níveis distintos de contribuição por meio de cada participante. Isso ocorre em virtude do caráter dinâmico do CS, tornando praticamente impossível premeditar contribuições esperadas. Em DDS, as atividades são planejadas para cada *site* e as contribuições são previamente definidas.
 3. *Processo*: O processo de contribuição das *microtasks* é mais oneroso ao *crowdsourcer*. Isso ocorre devido a necessidade do *crowdsourcer* fiscalizar e abstrair a contribuição de cada participante. Em DDS o processo de contribuição pode ser simplificado, pois cada *site* recebe um conjunto de atividades que são direcionadas ao *site* e não ao participante.
- **Avaliação**: A avaliação de uma *microtask* pode ser efetuada por meio do *crowd*, *crowdsourcer* e/ou integração. Dessa forma, existem diferentes abordagens a serem consideradas conforme a natureza de cada projeto. Em DDS, as atividades também podem ser avaliadas pelo gerente, *sites* e na integração. Porém, como apresentou Khan et al. (2017), no DDS o cliente exerce influência no projeto. E nesse cenário, o cliente também realiza a avaliação final do projeto.
 1. *Líder*: O *crowdsourcer* pode realizar a avaliação em cada *microtask*. Porém, se um projeto possui um conjunto amplo de *microtasks* isso pode aumentar consideravelmente a sua carga de trabalho. O gerente também pode participar da avaliação de uma atividade DDS. Todavia, grande parte dos projetos DDS são modularizados, assim, o gerente ao invés de realizar a avaliação por atividade, realiza a avaliação por meio dos módulos finalizados.
 2. *Time*: O *crowd* e o *site* podem avaliar de forma similar as *microtasks*/atividades do projeto. Contudo, em DDS um participante “A” pode solicitar que um participante “B” realize a avaliação da atividade. Em CS, devido aos participantes serem anônimos, cada participante realiza a avaliação de sua própria execução da *microtask*. Para evitar má intenção dos participantes, o *crowdsourcer* pode requerer que outros participantes do *crowd* realizem a avaliação da *microtask*.
 3. *Processo*: A avaliação é complexa nas *microtasks* e nas atividades de software. Isso ocorre devido ao CS, em determinados casos, necessitar de um segundo *crowd* para executar a avaliação. Enquanto que no DDS, além do gerente e dos *sites*, o cliente também participa do processo.
 - **Segurança**: As *microtasks* podem possuir restrições quanto aos dados ou a proteção intelectual. Essas restrições ocorrem em vista do caráter dinâmico do CS, da participação anônima, e da contratação realizada somente por meio eletrônico. Em DDS, o papel

do cliente também pode ser adicionado a segurança das atividades. Com isso, segundo Niazi et al. (2016), a proteção intelectual pode ser considerada um importante desafio do relacionamento entre gerente e cliente.

1. *Líder*: O *crowdsourcer* pode possuir desafios com a segurança das *microtasks*. Assim, dados sensíveis (como senhas ou informações pessoais) podem ser criptografados ou substituídos. Em caso de impossibilidade de tratar a segurança de dados, o *crowdsourcer* pode optar por expor as informações ou buscar uma nova estratégia. Quanto a proteção intelectual, o *crowdsourcer* torna-se dependente do *crowd*. Em DDS, cabe ao gerente conduzir avisos, firmar contratos e realizar diálogos com o cliente, a fim de minimizar desafios da segurança das atividades.
 2. *Time*: A segurança das *microtasks* influencia diretamente o *crowd*. A exposição de informações sensíveis ou os desafios do uso de propriedade intelectual podem maximizar diversos riscos devido a participação anônima. Além disso, soluções criativas para as *microtasks* tornam-se propriedade do *crowdsourcer*. Com isso, podem ser replicadas e/ou distribuídas. Em DDS, a segurança é diretamente tratada entre o gerente e o cliente, e assim, os *sites* podem buscar auxílio com o líder a fim de determinar a estratégia necessária.
 3. *Processo*: O processo da segurança nas *microtasks* é tratado entre o *crowdsourcer* e o *crowd*, e com isso apresentar todas as considerações de execução, apesar de trabalhoso, pode representar a fluidez no desenvolvimento. Em DDS, o processo de segurança é dividido em duas partes: a primeira parte é encarregada no relacionamento cliente-gerente, ocorrendo principalmente o diálogo para alinhar as estratégias. A segunda parte é associada entre o *sites*-gerente, ocorrendo o alinhamento com base nos diálogos anteriormente realizados.
- **Finalidade**: A finalidade de uma *microtask* pode influenciar de forma distinta cada participante. Sendo assim, podem existir participantes focados na premiação final, na melhoria de seu portfólio, no desafio proposto, etc. Em DDS existe maior rigidez devido aos contratos estabelecidos e a rotina de trabalho. Entretanto, de acordo com Farias et al. (2017), o contexto do DDS pode enriquecer um projeto. Nesse sentido, o autor destaca um exemplo em que o intenso contato dos participantes de um *site* pode motivar a tomada de decisões.
 1. *Líder*: A finalidade de uma *microtask* é definida pelo *crowdsourcer*, todavia, somente questões intrínsecas podem ser evidenciadas. Nesse sentido, destaca-se a premiação que o *crowdsourcer* atribui a *microtask*. Em DDS, o gerente fica restrito

as definições do projeto. Dessa forma, o líder DDS fica submetido as aspirações de cada participante.

2. *Time*: O *crowd* pode possuir diferentes percepções relacionadas à finalidade de cada *microtask*. Por isso, pode-se dizer que os participantes estão relacionados as questões extrínsecas que a execução da *microtask* irá propiciar. Em DDS, similarmente ao gerente, os *sites* também podem estar restritos devido aos termos contratuais. Entretanto, pode-se optar pela criação de um ambiente que favoreça o crescimento profissional e busque motivar os participantes para a execução das atividades.
3. *Processo*: A finalidade de uma *microtask* pode ser dividido em questões intrínsecas (que dizem respeito ao *crowdsourcer*) e extrínsecas (com referência ao *crowd*). Enquanto que a finalidade de uma atividade de software demonstra ser relativa ao ambiente de trabalho. O gerente e o site estão submetidos ao projeto, sendo assim, o gerente precisa ocupar-se com o desenvolvimento de um ambiente favorável ao mesmo tempo que precisa receber apoio dos participantes de cada *site*.

4.3.4 RESUMO E DISCUSSÕES

Em vista do conjunto de comparações efetuadas foram identificados diversos aspectos que contrastam entre as *microtask* e as atividades de software executada pelo DDS. Em síntese, as análises foram centradas em três pares de papéis similares do CS e do DDS: *crowdsourcer* x gerente, *crowd* x *site* e processo CS x processo DDS. Os três papéis investigados em CS foram suficientes para analisar as *microtasks*. Entretanto, as análises das atividades DDS apresentaram a inclusão de um papel denominado cliente, que também atua nas atividades do projeto. O CS apresenta uma facilitação desse cenário, e com isso, os papéis do gerente e do cliente são unificados e consolidados no *crowdsourcer*.

Contudo, a unificação de dois papéis DDS (gerente + cliente) em um único papel CS (*crowdsourcer*) resulta em uma grande carga de responsabilidade e trabalho. Com a condução das análises foi identificado que o *crowdsourcer* pode tornar sua rotina de trabalho onerosa e isolada, quando comparada a rotina do gerente DDS. Ademais, a tomada de decisões e estruturação das *microtasks* podem representar diversos desafios devido à falta de um auxílio externo, como o apoio fornecido pelos *sites* ao gerente.

As análises demonstraram também o caráter funcional que as *microtasks* possuem. Inicialmente, é responsabilidade do *crowdsourcer* separar, estruturar e organizar as porções de trabalho. Após isso, a execução é dividida entre *crowdsourcer* e *crowd*. Ao *crowd* cabe

a responsabilidade de executar a *microtask* obedecendo estruturas previamente definidas. Ao *crowdsourcer* cabe a responsabilidade de fiscalizar o processo de execução. No DDS as responsabilidades são divididas entre o gerente e os *sites*. Para isso, apesar do gerente possuir a responsabilidade com maior grau hierárquico, as decisões são frutos de um relacionamento entre gerente e *sites*, e por vezes, com o cliente.

O processo das *microtasks* e das atividades DDS apresentaram diferentes contrastes. As diferenças decorrem em vista da carga de trabalho que o líder ou o time recebe. Com isso, conforme ocorre a variação da carga de trabalho, o processo torna-se automaticamente mais oneroso de ser executado.

Em resumo, pode-se dizer que as *microtasks* são voltadas, principalmente, a “liberdade de execução”. Nelas, os *crowdsourcers* possuem independência para definirem o prazo, o preço e os requisitos necessários para execução. Ademais, diversos os aspectos, como a granularidade ou as tecnologias necessárias, são definidas unicamente pelo *crowdsourcer*. Na outra ponta, o *crowd* acaba possuindo autonomia para selecionarem as *microtasks* e optarem por se desligarem sem nenhum tipo de multa ou penalização. Enquanto que as atividades de software podem ser definidas pelo termo “dependência de execução”. Em tais atividades, existe uma complexa rede de papéis que trabalham e cooperam para a sua execução. Assim, a falha de qualquer papel acaba gerando um efeito em cascata que influencia todos os papéis que estão inseridos na execução da atividade.

Com base nos resultados alcançados pode ser verificado que as atividades de software apresentam uma abordagem com maior suporte devido ao gerente e os *sites* operarem de maneira conjunta. Assim, grande parte da carga de trabalho é centrada em dois papéis. Enquanto isso, as *microtasks* tornam oneroso o papel do *crowdsourcer* devido a necessidade de equilibrar diversas responsabilidades. Contudo, as *microtasks* apresentam alguns benefícios ao líder CS, como a flexibilização de contrato e a centralização de decisões.

Finalmente, a conclusão do terceiro pilar desta pesquisa simbolizou o encerramento da aplicação do estudo de caso e do MS como abordagem para investigar as *microtasks*. Toda a experiência proporcionada forneceu insumos para avançar a pesquisa para o método mais rígido de investigação que a Engenharia de Software permite aplicar, a experimentação controlada. Os três pilares iniciais (uso, características e contrastes das *microtasks*) serviram de base para compreender as *microtasks*. Além disso, também geraram uma ampla gama de experiência sobre o tema, permitindo executar o quarto e último pilar desta pesquisa sobre as *microtasks*. Logo, a próxima seção deste trabalho apresenta como um experimento controlado foi executado para modelar uma abordagem capaz de aferir a complexidade das *microtasks*.

4.4 COMPLEXIDADE

Para analisar a complexidade das *microtasks*, edificando o quarto e último pilar deste trabalho, foi necessário compreender a dinâmica de funcionamento do desenvolvimento de software CS. Com a execução dos três pilares iniciais da pesquisa, tornou-se claro que o conceito do CS se baseia em um princípio orgânico. Ou seja, existe liberdade para que o *crowdsourcer* ou os participantes do *crowd* estruturem e executem as *microtasks* conforme a sua própria percepção. Assim, o desenvolvimento de software CS se assemelha a uma floresta, em que o resultado final é difícil de ser previsto em decorrência dos fatores aleatórios e externos, como a experiência de cada participante. Enquanto isso, os modelos tradicionais de desenvolvimento são similares a uma plantação, ou seja, o resultado final é fruto de um entendimento lógico entre todos os participantes. Assim, quase sempre existe uma expectativa do que deverá ser entregue.

Tendo em vista este conceito orgânico de desenvolvimento, foi necessário adaptá-lo para a abordagem adotada nesta etapa da pesquisa, a experimentação controlada. Todavia, a literatura sobre esse assunto ainda é muito incipiente. Atualmente, existem poucos trabalhos disponíveis que possam fornecer insumos sobre a configuração, as variáveis e os desafios da execução de experimentos CS. Logo, o primeiro passo foi desenvolver e modelar toda a expertise necessária. Para isso, foram conduzidos dois testes experimentais, respectivamente, para verificar: i) como simular o CS e ii) como aplicar *microtasks* em ambientes controlados. Assim, todo o conhecimento fornecido pelos testes experimentais foi utilizado para planejar e executar com maior qualidade o experimento controlado.

O experimento buscou simular um ambiente de desenvolvimento CS em todos os aspectos, como a o uso de repositórios para o projeto; a distribuição e assincronismo dos participantes; os diferentes níveis de conhecimento e a ausência de reuniões presenciais. Ademais, as *microtasks* utilizadas no experimento controlado foram baseadas em *microtasks* reais. Com isso, a simulação forneceu alto grau de semelhança com o desenvolvimento de software CS.

O objetivo deste pilar da pesquisa foi propor uma abordagem capaz de aferir a complexidade das *microtasks*. Em resumo, a abordagem deveria apontar se uma *microtask* seria ou não seria executada pelo *crowd* devido a sua complexidade. Para alcançar este objetivo o restante desta seção revisa as métricas de complexidade disponíveis; apresenta a parte teórica e prática da abordagem proposta; demonstra o procedimento de validação adotado; expõe os resultados alcançados e o teste de hipótese. Finalmente, são apresentadas as considerações finais sobre a complexidade das *microtasks*.

4.4.1 MÉTRICAS EXISTENTES

A Engenharia de Software e o próprio desenvolvimento de software concentram um amplo conjunto de métodos, modelos ou abstrações sobre como aferir a complexidade de um projeto ou atividade de software. Diversos pesquisadores analisaram formas de mensurar as dimensões da complexidade de um projeto, resultando em abordagens distintas:

- *Mood/Chidamber e Kemerer (C&K)*: *Mood* é uma métrica proposta por Abreu e Carapuça (1994) e a *C&K* é uma métrica proposta por Chidamber e Kemerer (1991). Ambas as métricas são focadas na aferição de complexidade do paradigma de orientação a objetos. Dessa forma, abstraem-se um conjunto de fatores, como o encapsulamento, a herança e o polimorfismo para aferir a complexidade de um sistema ou atividade de software.
- *Halstead*: é uma métrica proposta por Halstead (1977) para tratar empiricamente a complexidade de um sistema por meio da sua implementação. Em sua concepção original, a métrica buscava exibir dados diversos, como o tamanho, a dificuldade e o esforço de um sistema.
- *Complexidade Ciclomática*: métrica originalmente proposta por McCabe (1976) para mensurar e ilustrar a complexidade de um sistema. Baseado em uma abordagem de grafos, a complexidade ciclomática demonstra os diversos caminhos que uma execução pode seguir.
- *Mean Time between Failures e suas variações*: a métrica discutida por Engelhardt e Bain (1986) é aplicada para prever e identificar as falhas, abortos, erros críticos ou respostas inexatas de um sistema em execução.
- *Pontos por função*: métrica apresentada por Albrecht (1979) para mensurar a produtividade nas diferentes fases de um projeto de software. A aplicação de pontos por função em um projeto pode ser utilizada para diversos fins, como mensuração do custo financeiro, temporal, de esforço ou de complexidade.
- *Constructive Cost Model (COCOMO) e suas variações*: métrica proposta por (BOEHM, 1981) para estimar diversas informações de um projeto de software, tal como o custo, o esforço e a duração base do projeto. O COCOMO foi estruturado originalmente para classes distintas de projeto, levando em consideração características como tamanho, necessidade de inovação e ambiente de desenvolvimento.

Acoplar alguma métrica existente para a realidade CS poderia tornar-se inviável por um conjunto de fatores. O desafio das métricas *Mood* ou *C&K* reside na amplitude de aplicação das *microtasks*. Como discutido na Seção 4.1 deste trabalho, a aplicação das *microtasks* pode variar entre análises, prototipação ou codificação, e dessa forma, acoplar a métrica *Mood* ou *C&K* para as *microtasks* poderia descaracterizar completamente o conceito original. Esse desafio também ocorreu frente à métrica *Halstead* que trata apenas da etapa de implementação. A complexidade ciclomática e a *Mean Time between Failures* foram desconsideradas devido ao aumento considerável de carga de trabalho do *crowdsourcer* e a possível descaracterização das abordagens originais. A métrica de pontos por função não demonstrou aplicabilidade para o desenvolvimento CS, pois utiliza uma base histórica e o consenso de um grupo de participantes, e em CS, o histórico nem sempre está disponível, tal como os participantes são anônimos. Finalmente, aplicar o COCOMO para as *microtasks* demonstrou baixa performance devido ao caráter dinâmico do CS e das *microtasks* (MAO et al., 2013).

Existem outros exemplos de métricas para aferição de complexidade presentes na literatura. Entretanto, grande parte utiliza um princípio semelhante com base em cálculos matemáticos. Desse modo, cada métrica possui entradas, variáveis, processamento e resultados distintos que podem guiar o desenvolvimento de um projeto ou de uma atividade de software. Porém, com base nos resultados obtidos no MS e nas considerações de LaToza e Hoek (2016) e Deus et al. (2017a) ainda persiste uma lacuna de análise referente à uma métrica própria para aferir a complexidade de uma *microtask*.

4.4.2 ABORDAGEM TEÓRICA

Enquanto grandes partes das métricas disponíveis se baseiam em características comuns de projetos, as *microtasks* possuem diversas particularidades. E nesse sentido, foram considerados os seguintes aspectos para modelar uma abordagem voltadas ao desenvolvimento de software CS:

1. *Abrangência*: as *microtasks* podem estar concentradas em diversas etapas de um projeto, dessa forma, deve ser considerada a sua amplitude de aplicação.
2. *Praticidade*: o *crowdsourcer* possui grande parte da carga de trabalho, e assim, deve ser considerada uma abordagem que não torne sua responsabilidade ainda mais onerosa.
3. *Informativa*: as características presentes nas *microtasks* podem influenciar a complexidade, e nesse sentido, deve ser extraído o conjunto de informação presente em suas características.

4. *Objetividade*: as *microtasks* podem assumir diversos estados, e com isso, deve-se tornar visível ao *crowdsourcer* qual (ou quais) estados podem estar influenciando a complexidade.

4.4.3 ABORDAGEM PRÁTICA

Em vista da abordagem teórica apresentada anteriormente foi realizada uma conversão e quantificação das características e estados que uma *microtask* pode assumir. Como foi apresentado na Figura 23 (página 73), as características são divididas em dois grupos: exclusivas ou inclusivas. Ademais, alguns estados assumidos por cada característica de uma *microtask* podem reter um conjunto maior de informação, como exemplo, ao assumir que uma *microtask* possui o estado de tecnologia definida, tal *microtask* pode reter uma ou várias tecnologias. Assim sendo, para captar toda a informação presente nas características das *microtasks* foi necessário gerar três conjuntos distintos:

- O primeiro conjunto (*A*) representa as características que podem assumir um estado considerado ausente, ou seja, nenhum tipo de melhoria ou otimização conseguirá diminuir a complexidade da *microtask* sob o ponto de vista daquele estado. Nesse conjunto encontram-se os seguintes estados das características:
 - *Sincronização*: pode assumir não obrigatoriedade.
 - *Granularidade*: pode ser atomizada.
 - *Limitação*: pode não estar presente.
 - *Contribuição*: pode ser indefinida.
 - *Dependência*: pode não existir.
 - *Segurança*: pode ser considerada baixa.
- O segundo conjunto (*B*) representa as características que assumem um estado considerado presente, com isso, pode-se afirmar que sob o ponto o vista desse estado existe um acréscimo da complexidade da *microtask* que não pode ser reduzido. Esse conjunto reúne as seguintes características:
 - *Tipo*: deve ser colaborativa ou competitiva.
 - *Tecnologia*: pode ser livre.
 - *Expertise*: pode ser criativa.
 - *Abertura*: pode ser total.

- *Avaliação*: pode ocorrer pelo *crowd*.
- *Finalidade*: pode ocorrer pelo prêmio.
- O terceiro conjunto (C) foi gerado para representar a única característica temporal. Nesse conjunto é encontrada somente a seguinte característica:
 - *Prazo*: pode ser mensurado em horas ou dias.

O terceiro conjunto possui uma mensuração distinta dos demais, apresentando duas particularidades. Inicialmente, percebe-se que conforme o tempo disponível torna-se menor, a complexidade aumenta. Ou seja, se aumentar o total de tecnologias presentes em uma *microtask* aumentará também a complexidade. Porém, conforme aumentar o prazo de execução, a complexidade diminuirá. Além disso, existe uma segunda particularidade sobre o prazo das *microtasks* variarem entre horas ou dias. Nesse sentido, o prazo de uma *microtask* pode variar de três formas: i) de hora em hora até alcançar o total de vinte e quatro horas, ii) dia em dia sem prazo máximo definido ou iii) somando dias e horas. A exemplo, tem-se que se uma *microtask* pode assumir o estado de tecnologia definida, sendo assim essa sempre possuirá uma ou mais tecnologias, similarmente ocorre à uma segunda *microtask* com o mesmo estado. Porém, ao assumir que uma *microtask* tem seu prazo assumido por duas horas existe uma diferença entre uma outra *microtask* em que seu prazo é assumido por dois dias. Apesar do índice “dois” ser comum, representa diferentes escalas. Para contornar esse desafio, optou-se pela conversão do prazo baseada no total de dias. Sendo assim, ao assumir que uma *microtask* possui seu prazo definido por horas, executa-se uma conversão simples do total de horas dividido por 24 horas.

4.4.4 CONVERSÃO

Para identificar os elementos “ x ” dos conjuntos “ A, B e C ” deve-se realizar um processo de conversão de características e estados para valores. Com isso, inicialmente, o *crowdsourcer* deve selecionar o projeto CS e extrair a porção de trabalho definida para cada *microtask*. Após isso, as informações relevantes de todas as características devem ser abstraídas por meio de quatro etapas:

1. O *crowdsourcer* deve selecionar uma característica.
 - a Nessa etapa, o metamodelo apresentado na Figura 24 (página 74 deste trabalho) pode auxiliar ao *crowdsourcer* a identificar as influências de cada característica.

2. O *crowdsourcer* deve identificar o primeiro estado possível da característica selecionada. De uma forma geral, pode-se dizer:
 - a. Caso o estado assumido seja considerado ausente, deve ser atribuído o valor $x = 0$.
 - b. Caso o estado assumido seja considerado presente, deve ser atribuído o valor $x = 1$.
 - c. Caso o estado assumido acrescente alguma informação, deve ser atribuído o valor $x = N$ e, logo após, iniciar o processo de identificação de N :
 - i) O primeiro passo é abstrair as informações que o estado possui. Exemplo: esta *microtask* possui o estado de tecnologia definida pois necessita do uso de tecnologias específicas para a sua execução.
 - ii) O segundo passo é realizar a quantificação. Com isso, o *crowdsourcer* estabelece o valor de N . No caso do exemplo utilizado anteriormente, o *crowdsourcer* deveria verificar o total de tecnologias necessárias para a execução. Supondo que fossem necessárias três tecnologias distintas, isso resultaria em um valor do $N = 3$, e logo se $x = N$ então $x = 3$. Caso esteja sendo convertido o prazo da *microtask*, deve-se usar o total de dias, se o prazo estiver estipulado em horas deve-se converter ($horas/24$).
3. O *crowdsourcer* deve verificar se o estado assumido é exclusivo ou inclusivo.
 - a. Caso o estado assumido seja exclusivo devem ser verificadas as seguintes condições:
 - i) se $x = 0$ deve-se converter o outro estado da característica.
 - ii) se $x \geq 1$ o outro estado da característica recebe $x_2 = 0$.
 - b. Caso o estado assumido seja inclusivo, obrigatoriamente, deve-se converter o(s) outro(s) estado(s) da característica.
4. Após a definição do valor de x , deve-se adicioná-lo ao respectivo multiconjunto da característica. Caso o valor de x_2 também tenha sido calculado durante o processo, deve-se adicioná-lo ao multiconjunto.

Para facilitar a compreensão do processo de conversão das características, a Tabela 17 sintetiza a definição de cada multiconjunto e o intervalo possível de ser adicionado ao valor de x . O fluxo de informação e decisão referente ao processo de conversão das características das *microtasks* encontra-se apresentado por meio da Figura 26. A Figura ilustra as etapas e as tomadas de decisões que o *crowdsourcer* deve executar durante criação dos elementos de cada multiconjunto.

Tabela 17: A conversão das características das *microtasks*

Conjunto	Exclusiva/ Inclusiva	Característica	Estado	Intervalo de x	
				Mínimo	Máximo
(A)	Exclusiva	Sincronização	Obrigatória	1	N
			Não aplicável	0	0
	Exclusiva	Granularidade	Atomizada	0	0
			Divisível	1	N
	Exclusiva	Limitação	Possui	1	N
			Não possui	0	0
	Exclusiva	Contribuição	Definida	1	N
Indefinida			0	0	
Inclusiva	Dependência	Interna	1	N	
		Externa	1	N	
		Nenhuma	0	0	
Inclusiva	Segurança	Dados	1	N	
		Intelectual	1	N	
		Baixa	0	0	
(B)	Exclusiva	Tipo	Colaborativa	1	1
			Competitiva	1	1
	Exclusiva	Tecnologia	Definida	1	N
			Livre	1	1
	Exclusiva	Expertise	Determinada	1	N
			Criativa	1	1
	Inclusiva	Abertura	Total	1	1
			Reputação	1	N
			Credencial	1	N
	Inclusiva	Avaliação	<i>Crowd</i>	1	1
			<i>Crowdsourcer</i>	1	N
			Integração	1	N
	Inclusiva	Finalidade	Prêmio	1	1
			Motivação	1	N
(C)	Inclusiva	Prazo	Horas	1	24
			Dias	0	N

Fonte: Autoria própria.

Como foi apresentado anteriormente neste trabalho (subseção 4.4.3, páginas 93 e 94) o conjunto “C” representa a o tempo disponível de execução da *microtask*, e por isso, conforme menor o tempo disponível, maior a complexidade. Contudo, os conjuntos “A” e “B” representam os demais estados e características da *microtasks*, e com isso, conforme maior o valor dos elementos desses multiconjuntos, maior a complexidade. Em vista desse cenário, inicialmente, pode-se especificar a complexidade de uma *microtask* como a soma dos elementos dos multiconjuntos A e B. Então, para x sendo os elementos dos conjuntos, C_{ab} o valor final, a complexidade de uma *microtask*, será dado pela fórmula:

$$C_{ab} = \sum_{x \in A} x + \sum_{x \in B} x \quad (1)$$

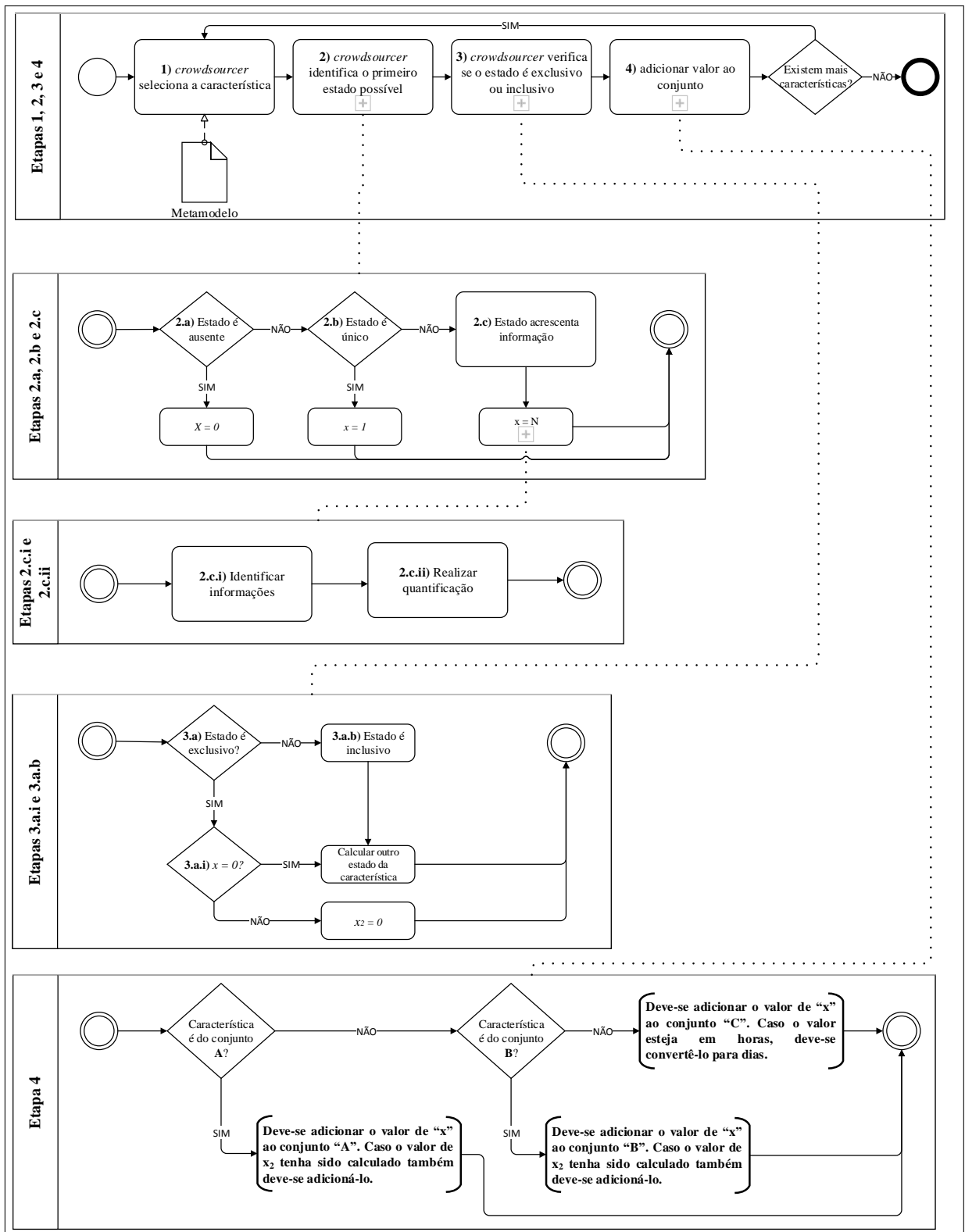


Figura 26: Fluxograma de conversão das características e estados das *microtasks*

Fonte: Autoria Própria

Contudo, o cálculo de complexidade apenas torna-se completo com a utilização do prazo da *microtask* presente no multiconjunto C . Para isso, optou-se por definir dois índices fixos: 0 e 1. O objetivo desses índices foi definir uma variação para complexidade da *microtask*, com isso, conforme mais próximo de 1, maior a complexidade. Em contrapartida, quanto mais próximo à zero, menor será a complexidade da *microtask*. Para validar o intervalo do índice e assegurar que a complexidade final de uma *microtask* será restrita entre 0 e 1 foi necessário impor uma regra: o prazo final C_c sempre deverá ser menor ou igual que C_{ab} . Assim, pode-se escrever que:

$$C_x = \sum_{x \in C} x \quad (2)$$

$$C_c = \begin{cases} C_{ab}, & \text{se } C_x > C_{ab} \\ C_x, & \text{se } C_x < C_{ab} \end{cases} \quad (3)$$

Finalmente, para concluir o processo de conversão pode-se utilizar a seguinte notação, sendo C_o o índice final de complexidade da *microtask*:

$$C_o = \left(\frac{(C_{ab} - C_c)}{C_{ab}} \right) \quad (4)$$

Para representar a aplicação do cálculo de complexidade de uma *microtask* pode-se adotar o seguinte exemplo: supondo que determinada *microtask* possui em seus conjuntos A e B o valor final $C_{ab} = 35$, possuindo apenas 2 dias de prazo de execução. Na abordagem proposta, isso irá representar ao final um índice de complexidade $C_o = 0.94$. Ao modificar o seu prazo para 10 dias, o índice de complexidade final será $C_o = 0.71$. Ou seja, ao aumentar o prazo, o índice de complexidade diminuiu. Em vista disso, pode-se afirmar que o conjunto de instruções apresentados na subseção 4.4.2 deste trabalho (páginas 92 e 93) foram considerados da seguinte forma: no ponto de vista da *abrangência* percebe-se que todas as características foram passíveis de serem identificadas nas *microtasks*. A *praticidade* também foi acoplada na abordagem exposta, sendo aplicado na própria rotina do *crowdsourcer*, além disso, pode servir como auxílio para cadastrar informações da *microtask* na plataforma CS. O aspecto *informativo* representou o principal apoio, visto que uma das premissas adotadas refere-se ao total de informações extraídas e o nível de complexidade. Finalmente, ao fazer uso da abordagem exposta, o aspecto de *objetividade* foi alcançado com base no cálculo que pode ser realizado.

4.4.5 VALIDAÇÃO EMPÍRICA

A abordagem proposta para aferir a complexidade de uma *microtask* foi avaliada empiricamente por meio da experimentação controlada. Assim, ocorreram previamente dois testes experimentais executados em ambientes acadêmicos. No primeiro teste experimental foi investigado se o CS poderia ser simulado, e no segundo teste experimental foi avaliado a simulação do uso de *microtasks*. Por fim, o experimento controlado foi executado.

4.4.5.1 PRIMEIRO TESTE EXPERIMENTAL

O primeiro teste experimental foi executado para verificar se o CS poderia ser simulado em um ambiente acadêmico. Para isso, os participantes deveriam realizar a montagem de um robô *Lego Mindstorms* por meio do CS. Dessa forma, foram utilizados dois papéis: *crowd* e *crowdsourcer*. O *crowd* recebeu uma planilha descritiva das peças do kit *Lego Mindstorms* e o *crowdsourcer* recebeu as peças físicas do kit. O *crowd* deveria propor estratégias de montagem e enviar as informações ao *crowdsourcer*.

Para a condução do primeiro teste experimental foram formados dois grupos. Cada grupo possuía um *crowd* e um *crowdsourcer*. Os grupos foram nomeados aleatoriamente por grupo “A” e grupo “B”, e ambos atingiram o objetivo final de realizar a montagem do robô usando CS. Durante a execução do teste experimental, o grupo “A” utilizou cerca de 39 minutos e 24 segundos, enquanto que o grupo “B” empregou 44 minutos e 25 segundos. Porém, o grupo “A” utilizou 167 interações via ferramenta de comunicação, à medida que o grupo “B” utilizou apenas 103 interações.

Também foi notado um exemplo de desafio resultante da abordagem CS. Durante a execução do teste experimental o *crowd* modificou diversas vezes a estratégia de montagem, e com isso, algumas porções de trabalho acabaram desperdiçadas. Todavia, ambos os grupos concluíram com êxito o teste experimental. O produto final gerado pelos grupos “A” e “B” podem ser consultados na Figura 27.

Durante a execução do primeiro teste experimental duas variáveis dependentes foram utilizadas: a distância entre o *crowd/crowdsourcer* e a ferramenta de comunicação. Notou-se que utilizar um mesmo ambiente para o *crowd* e o *crowdsourcer* trouxe um conjunto de desafios, como o ruído provocado pelo participantes e a complexidade de bloquear a transmissão de diálogos ou mensagens além da ferramenta de comunicação. Porém, apesar desses desafios, verificou-se que o objetivo de realizar a montagem do robô usando CS foi atingido. Assim, a hipótese H_1 foi considerada verdadeira: O CS pode ser simulado em um ambiente acadêmico.

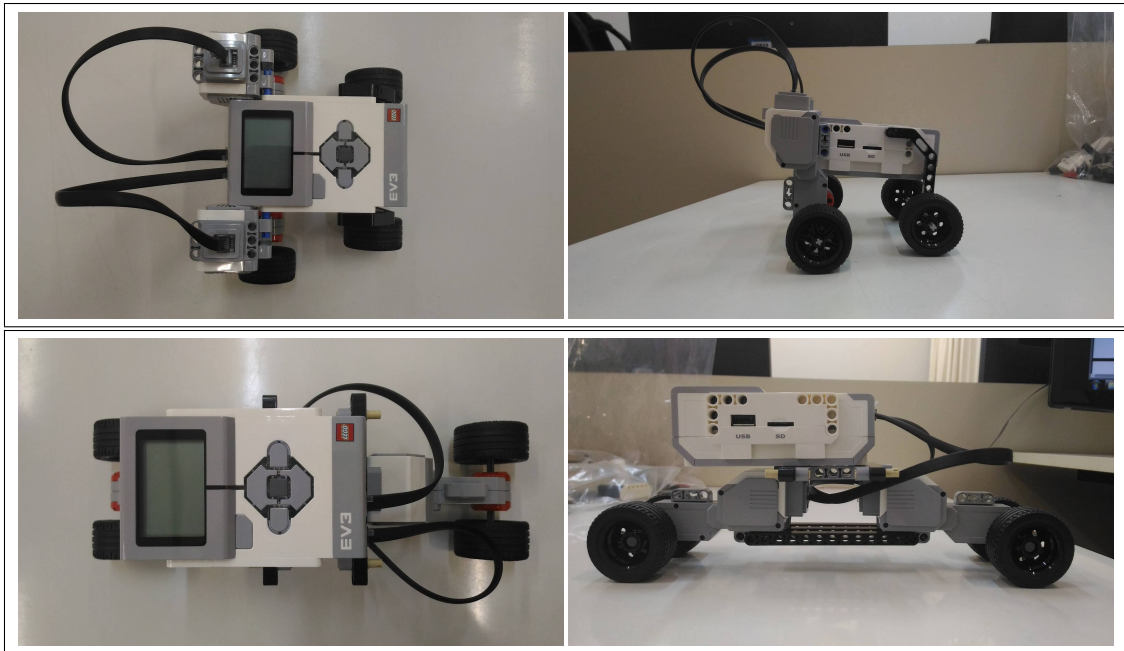


Figura 27: Montagem dos robôs *Lego Mindstorms* grupo “A” (acima) e grupo “B” (abaixo)

Fonte: Autoria Própria

4.4.5.2 SEGUNDO TESTE EXPERIMENTAL

O objetivo do segundo teste experimental resultou do amadurecimento dos resultados obtidos no teste anterior. Dessa forma, foi verificado como um projeto de software CS utilizando *microtasks* poderia ser simulado em um ambiente acadêmico. Para isso, o *crowdsourcer* apresentou a orientação de um projeto de software a ser desenvolvido via CS. Os participantes deveriam formar um *crowd* para elaborar, documentar e codificar o projeto por meio de *microtasks*. Cabe salientar que por tratar-se de um projeto de software CS, todas as *microtasks* foram analisadas conforme a sua granularidade. As *microtasks* consideradas onerosas ou impraticáveis foram removidas do teste experimental.

Na condução do segundo teste experimental os participantes realizaram diversas contribuições para o projeto de software CS. Inicialmente, foram elaborados 20 requisitos para o projeto. Cada requisito foi vistoriado pelo *crowdsourcer* e por um orquestrador auxiliar para analisar a viabilidade do requisito, bem como a sua aplicação no escopo do sistema. O *crowd* operou como uma entidade única, padronizando as porções de trabalho e respeitando as orientações do *crowdsourcer*. Com isso, podem ser destacados diferentes exemplos de execução, como a elaboração de digramas do projeto, apresentado na Figura 28; a organização do repositório, exposto na Figura 29; e um exemplo de implementação do projeto, como mostra a Tabela 18.

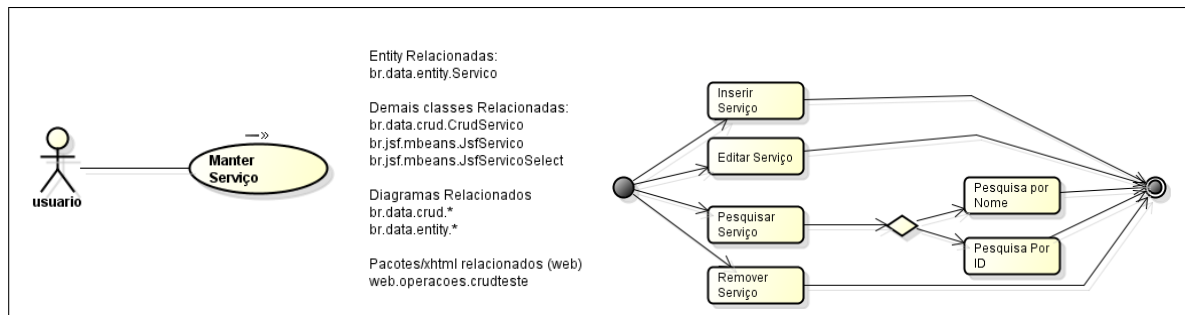


Figura 28: Exemplo de digrama gerado pelo crowd

Fonte: Autoria Própria

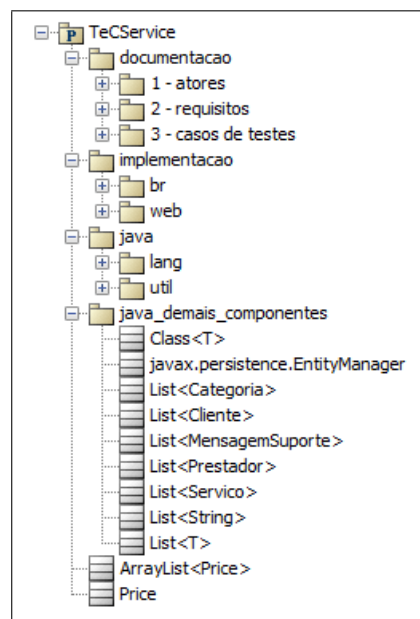


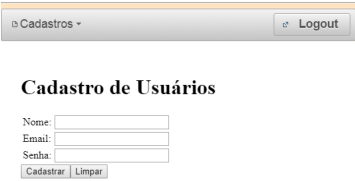
Figura 29: Estrutura do repositório gerado pelo crowd

Fonte: Autoria Própria

O projeto foi desenvolvido utilizando um repositório online, e com isso, foi possível registrar todas as contribuições dos participantes. Com base na relação de registros disponíveis ficou evidenciado diferentes dimensões da adoção do CS. Inicialmente, houveram registros de participantes que ao concluírem determinada *microtask* buscaram corrigir erros de configuração no projeto. Também foram identificados participantes que contribuíram em *microtasks* distintas, auxiliando outros participantes. Por fim, alguns participantes monitoraram a criação de testes automatizados, com a finalidade de garantir a correta execução do projeto.

Porém, a simulação de um projeto de software CS também evidenciou desafios. O primeiro desafio foi a necessidade de intervenção do *crowdsourcer* em diferentes momentos. Apesar de alguns participantes realizarem operações de correção no projeto, o *crowdsourcer*

Tabela 18: Exemplo de página gerado pelo crowd

Página gerada	Código fonte
	<pre> <?xml version='1.0' encoding='UTF-8' ?> <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> <html xmlns="http://www.w3.org/1999/xhtml" xmlns:h="http://xmlns.jcp.org/jsf/html" xmlns:f="http://xmlns.jcp.org/jsf/core" xmlns:ui="http://xmlns.jcp.org/jsf/facelets"> <link href="index.css" rel="stylesheet" type="text/css"/> <h:head> <title>Usuários</title> </h:head> <ui:composition template="..layout_operacoes.xhtml"> <ui:define name="content"> <h:body class="background-cad"> <h1> <div class="txt-Cadastro">Cadastro de Usuários</div> </h1> <h:form> <div class="cad-center"> <h:panelGrid columns="3"> <h:outputLabel class="txt-input" value="Nome:" /> <h:inputText id="nome" class="c-input" size="20" value="#{jsfCliente.nome}" /> <h:message for="nome" /> <h:outputLabel class="txt-input" value="Email:" /> <h:inputText id="nome1" class="c-input" size="20" value="#{jsfCliente.email}" /> <h:message for="email" /> <h:outputLabel class="txt-input" value="Senha:" /> <h:inputText id="nome2" class="c-input" size="20" value="#{jsfCliente.senha}" /> <h:message for="senha" /> </h:panelGrid> <div class="bts"> <h:commandButton id="btnE" class="bt-padrao" value="Cadastrar" action="#{jsfCliente.persist()}" /> <h:commandButton id="btnL" class="bt-padrao" value="Limpar" /> </div> </h:form> </h:body> </ui:define> </ui:composition> </html> </pre>

Fonte: Autoria própria.

sempre precisava monitorar a rotina de correção. Além do mais, operações mais complexas do código fonte, como “merge” ou “revert” eram operadas unicamente pelo *crowdsourcer*. Finalmente, o *crowdsourcer* também precisava operar como um gerente do projeto, fiscalizando datas e entregas.

No segundo teste experimental foram adotadas duas variáveis dependentes: a dispersão física e a granularidade das *microtasks*. A dispersão física dos participantes permitiu minimizar quase todos os desafios do primeiro teste experimental. Desse modo, o ruído foi extinto e a dificuldade em bloquear a transmissão de diálogos ou mensagens sem utilizar uma ferramenta de comunicação foi eficientemente reduzido. Ademais, a granularidade foi fiscalizada diretamente pelo *crowdsourcer* apoiado por auxiliares. E com isso, todos os requisitos foram transformados em *microtasks*, que foram elaboradas e codificadas. Com base nessas evidências a hipótese H_1 foi considerada assertiva, ou seja, foi possível simular o CS e a execução de *microtasks* em um ambiente acadêmico.

4.4.5.3 EXPERIMENTO CONTROLADO

O experimento controlado foi executado com o propósito de identificar se a abordagem proposta de aferição de complexidade de *microtasks* seria aplicável na realidade do CS. Todavia, os resultados gerados pelos dois testes experimentais foram essenciais para a execução do experimento controlado. Com isso, foram identificadas as seguintes dimensões da simulação do CS e das *microtasks* no ambiente acadêmico:

- a. Comunicação: a comunicação do CS é um dos aspectos mais importantes para a condução de uma simulação. Por isso, para adaptar-se à realidade CS deve-se utilizar uma ferramenta de comunicação. Tal ferramenta possuía um mecanismo para registrar todas as interações entre os participantes, como envio de mensagens e horários.
- b. Dispersão: o CS emprega a dispersão física e temporal dos participantes. Logo, para a execução do experimento controlado esse aspecto foi estruturado da seguinte forma: a reunião de apresentação foi conduzida de maneira eletrônica e não houveram encontros pessoais no experimento. Assim, o risco dos participantes se aglomerarem pessoalmente durante a execução do experimento foi sensivelmente reduzido. Havia apenas duas formas de contato: envio de e-mail ou uso da ferramenta de comunicação. Em ambos os casos *crowdsourcer* e participantes poderiam enviar mensagens.
- c. Líder: ficou evidenciado na execução dos testes experimentais a necessidade do *crowdsourcer* atuar como um líder do projeto. Dessa forma, as responsabilidades do *crowdsourcer* foram: criação do projeto, criação das *microtasks*, alocação do projeto e das *microtasks* em uma plataforma CS, fiscalização do *crowd*, resolução de problemas, gerenciamento de prazos entre outras atividades para garantir o andamento do projeto.

Para executar o experimento controlado, o produto final gerado no segundo teste experimental foi refinado e reaproveitado pelo *crowdsourcer*. Dessa forma, o projeto construído via CS foi remodelado e extraíram-se quinze *microtasks*. As *microtasks* foram inspiradas por dados de *microtasks* reais e por isso mesclavam diferentes características para amplificar a sua configuração. A lista completa de *microtasks* encontra-se no Apêndice B deste trabalho para ser consultado.

4.4.6 VISÃO GERAL

As características e os estados de cada *microtask* foram atribuídos aleatoriamente. Dessa forma, não havia uma sequência lógica de distribuição das. Além disso, os participantes

eram livres para a seleção das *microtasks*, não havendo nenhum mecanismo para restringir o acesso. Quanto aos índices de complexidade obtidos, percebeu-se que a *microtask* com maior simplicidade alcançou $C_o = 0,33$. Enquanto isso, a *microtask* com maior complexidade alcançou $C_o = 0,96$. A apresentação completa da complexidade das *microtasks* utilizadas no experimento encontram-se na Tabela 19 .

Tabela 19: Complexidade das *microtasks* utilizadas no experimento controlado

Código	C_{ab}	C_c	C_o
#M001	28	1	0,96
#M002	18	6	0,66
#M003	24	10	0,58
#M004	15	10	0,33
#M005	17	9	0,47
#M006	15	10	0,33
#M007	16	7	0,56
#M008	38	10	0,73
#M009	12	5	0,58
#M010	20	10	0,50
#M011	27	10	0,62
#M012	27	10	0,62
#M013	17	10	0,41
#M014	29	10	0,65
#M015	29	10	0,65

Fonte: Autoria própria.

O experimento controlado também evitou que houvesse concentração de *microtasks* com a mesma complexidade. Esse mecanismo foi adotado para evitar que os participantes selecionassem apenas *microtasks* com complexidades similares, limitando a investigação. A frequência completa das *microtasks* investigadas pode ser encontrada na Tabela 20.

Tabela 20: Frequência da complexidade (C_o) das *microtasks* do experimento controlado

C_o	Frequência	Frequência Relativa
0,33	2	13,33%
0,41	1	6,67%
0,47	1	6,67%
0,50	1	6,67%
0,56	1	6,67%
0,58	2	13,33%
0,62	2	13,33%
0,65	2	13,33%
0,66	1	6,67%
0,73	1	6,67%
0,96	1	6,67%
Total	15	100%

Fonte: Autoria própria.

Os prazos das *microtasks* foram definidos dinamicamente durante a execução do experimento. Sendo assim, inicialmente, as *microtasks* não possuíam prazo de conclusão definido.

Tabela 21: Frequência do prazo das *microtasks* do experimento controlado

Prazo (em dias)	Frequência	Frequência Relativa
1	1	6,67%
5	1	6,67%
6	1	6,67%
7	1	6,67%
9	1	6,67%
10	10	66,67%
Total	15	100%

Fonte: Autoria própria.

Tabela 22: Nível de conhecimento dos participantes do experimento controlado

Nível de conhecimento	Descrição	Especificação
Alto	Experiência acadêmica e profissional	Adquirida no setor produtivo e de ensino
Médio	Apenas experiência profissional	Adquirida somente no setor produtivo
Baixo	Apenas experiência acadêmica	Adquirida somente no setor de ensino
Nenhum	Sem experiência	Não possui experiência adquirida

Fonte: Autoria própria.

Tabela 23: Frequência do nível conhecimento dos participantes sobre modelos de desenvolvimento de software

Nível de conhecimento	Modelos de desenvolvimento		
	<i>Open source</i>	<i>Distribuído</i>	<i>Crowdsourcing</i>
<i>Alto</i>	24,00%	6,00%	6,00%
<i>Médio</i>	23,00%	12,00%	12,00%
<i>Baixo</i>	18,00%	23,00%	23,00%
<i>Nenhum</i>	35,00%	59,00%	59,00%
Total	100,00%	100,00%	100,00%

Fonte: Autoria própria.

Tabela 24: Frequência do nível conhecimento dos participantes sobre as tecnologias das *microtasks*

Nível de conhecimento	Tecnologias		
	<i>JavaServer Faces</i>	<i>SQL</i>	<i>Projetos Web</i>
<i>Alto</i>	0,00%	25,00%	19,00%
<i>Médio</i>	0,00%	28,00%	25,00%
<i>Baixo</i>	41,00%	47,00%	56,00%
<i>Nenhum</i>	59,00%	0,00%	0,00%
Total	100,00%	100,00%	100,00%

Fonte: Autoria própria.

Entretanto, ao identificar que uma *microtask* iniciava uma concentração de registros, essa era automaticamente definida como “prazo encerrado” para não permitir o registro de novos participantes. Esse mecanismo teve de ser adotado devido ao número limitado de participantes que o experimento controlado concentrou. E, além disso, ao definir um prazo fixo, haveria uma tendência natural que os participantes guiassem a seleção de *microtasks* com os maiores prazos

de execução. A relação do prazo em dias e em horas encontra-se na Tabela 21.

Os participantes do experimento foram convidados a declararem o seu conhecimento sobre o CS ou modelos similares de desenvolvimento de software. Com isso, foi investigada a caracterização dos participantes, pois o desenvolvimento de software CS emprega uma população heterogênea, e nesse sentido, conforme maior a heterogeneidade dos participantes, mais similar com a realidade será a execução experimental. Como foi exposto na Tabela 22, os participantes deveriam optar por uma das quatro alternativas sobre seu conhecimento em cada assunto, variando desde nenhuma experiência até a experiência acadêmica e profissional.

A Tabela 23 exibiu que os participantes possuíram maior nivelamento quanto ao desenvolvimento de software *open source*. Em relação ao DDS e ao desenvolvimento de software CS o nível de conhecimento foi similar, e em síntese, poucos participantes detinham conhecimento acadêmico e profissional de tais modelos.

Os participantes também declararam o seu nível de conhecimento sobre as tecnologias empregadas no experimento. Nesse sentido, destaca-se que nenhum participante declarou conhecimento alto ou médio em *JavaServer Faces*. Enquanto que em *Structured Query Language* (SQL) e projetos *web*² todos os participantes haviam ao menos adquirido conhecimento de forma acadêmica. A Tabela 24 apresentou a frequência do nível de conhecimento com base nas tecnologias.

Durante a experimentação, os participantes foram convidados a declararem quais motivos os guiaram para a seleção de uma *microtask*. Os seguintes motivos foram identificados:

- **Interesse:** representa a vontade inata do participante em selecionar determinada *microtask*. Alguns exemplos de interesses identificados foram definidos como “*interesse pelo tipo de tarefa*”, “*job com foco em desenvolvimento*” e “*entusiasmo na realização da tarefa*”.
- **Experiência:** alguns participantes se guiaram pela experiência nas tecnologias presentes. Alguns exemplos mapeados como experiência incluem descrições como “*área profissional em que eu atuo*”, “*posso facilidade com prototipação de Interfaces e também por ter estudado matérias sobre o assunto no meu intercâmbio*” e “*aptidão na área*”.
- **Ordenação:** a disposição das *microtask* guiou alguns participantes. Exemplos encontrados referentes à ordenação foram definidos como “[*por*] *ser a primeira*”, “*ordem de atividades livres disponíveis*” e “*continuando desenvolvendo em ordem as tarefas disponíveis*”.

²Foram considerados projetos *web* os sistemas que são executados em navegadores

- **Complexidade:** foi definido como a seleção das *microtasks* com menor complexidade pelos participantes. Alguns exemplos de definição de simplicidade foram apresentados como: “*atividade simples para ajudar no entendimento do código-fonte do projeto*”, “*rápido para realização*” e “*fácil manutenção no código*”.

Os demais aspectos foram classificados como “outros” e representaram a menor porcentagem de incidência. A relação completa dos aspectos que guiaram os participantes durante o experimento controlado é encontrada na Tabela 25 .

Tabela 25: Frequência dos aspectos que guiaram os participantes a selecionarem uma *microtask*

Aspecto	Frequência
Interesse	16,00%
Experiência	20,00%
Ordenação	20,00%
Complexidade	36,00%
Outros	8,00%
Total	100,00%

Fonte: Autoria própria.

4.4.7 TESTE DE HIPÓTESE

Foram geradas duas hipóteses a serem testadas no experimento controlado. Em síntese, as hipóteses verificam a eficiência da abordagem proposta por este trabalho em aferir a complexidade de uma *microtask*. Para recapitular as hipóteses postuladas, elas são apresentadas a seguir:

- H_0 : O índice de execução de uma *microtask* não poderá ser aferido com base na abordagem proposta para mensurar a complexidade de uma *microtask*.
- H_1 : O índice de execução de uma *microtask* poderá ser aferido com base na abordagem proposta para mensurar a complexidade de uma *microtask*.

O índice de execução exposto nas hipóteses foi representado pelo total de participantes registrados *versus* o total de participantes que submeteram alguma contribuição para a *microtask*. Para verificar a correlação desse índice, foram considerados os aspectos que os participantes declararam guiá-los durante a experimentação (Tabela 25). Assim, o índice de execução foi confrontado com o interesse, a experiência, a ordenação e a complexidade das *microtasks*.

Para realizar o teste de hipótese foi adotado o coeficiente de correlação de *rank* de *Spearman's* (também conhecido como *rô* de *Spearman*). Esse teste, segundo Gibbons e Chakraborti

(2011), deve ser aplicado quando as amostras são pareadas e pretende-se identificar o grau de influência de uma determinada amostra sobre uma segunda amostra. Como as hipóteses geradas consideram a eficiência em relacionar o índice de execução com a abordagem proposta para aferir a complexidade de uma *microtask*, o teste deveria apontar um elevado grau de correlação entre essas amostras para provar a H_1 e assim refutar a H_0 . Ademais, sob o ponto de vista relacionado ao ponto de corte para rejeitar a H_0 foi definido o valor de 0,05. Isso significa que se o *Valor – P* obtido pelo coeficiente de correlação de *rank* de *Spearman's* fosse menor que 0,05 existia um intervalo de confiança de 95% para rejeitar H_0 .

O valor final do coeficiente de correlação de *rank* de *Spearman's* ocorre somente entre um intervalo que varia entre -1 até 1 . Conforme o coeficiente de correlação se aproxima a 0 , existem fortes indícios de não existir nenhum tipo de correlação entre as amostras. À medida que se o coeficiente de correlação se afasta de 0 existe uma taxa de correlação amostral (GIBBONS; CHAKRABORTI, 2011). Além disso, o coeficiente de correlação também demonstra a tendência das amostras. Nesse sentido, existe uma tendência natural das amostras crescerem unidas (se o coeficiente for > 0) e de um amostra crescer ao mesmo instante que a segunda amostra diminui (se o coeficiente for < 0).

O poder do coeficiente de correlação de *rank* de *Spearman's* depende diretamente das amostras que estão sendo utilizadas no teste. De acordo com Yue et al. (2002), se houverem amostras com diferentes propriedades, o poder do teste pode sofrer fortes influências. Para mitigar esse risco, as amostras foram tratadas e convertidas para apresentarem as mesmas propriedades. Duas amostras encontravam-se previamente com as mesmas propriedades, variando entre 0 e 1 , sendo elas o índice de execução (divisão dos registros pelas submissões das *microtasks*) e o índice de complexidade. Com isso, as demais amostras tiveram de ser convertidas para apresentar as mesmas propriedades por meio do seguinte processo:

- Identificar o maior elemento (M_{ax}) das amostras do interesse, da experiência necessária e da ordenação das *microtasks*.
- Converter os elementos das amostras para o índice entre 0 e 1 . Sendo i_a o índice final e x_a o elemento da amostra, foi utilizada a seguinte equação:

$$i_a = \left(\frac{x_a}{M_{ax}} \right) \quad (5)$$

A Tabela 26 apresenta as amostras após o processo de conversão. Para identificar o interesse foi adotado índice de registros da *microtask*. A ordenação foi classificada em duas formas distintas: i) ordem natural das *microtasks* e ii) ordem das *microtasks* conforme o prazo

disponível. A experiência foi mensurada com base no total de expertises necessárias para executar a *microtask*. Para a complexidade foi utilizada a abordagem proposta.

Tabela 26: Amostras utilizadas no teste de hipótese do experimento controlado

Código	Registros	Finalizações	Prazo	Ordem	Amostras convertidas					
					Inte- resse	Orde- nação	Ordenação (em dias)	Experi- ência	Comple- xidade	Índice de execução
#M001	4	3	1	1 ^a	0,5714	1,0000	0,1000	0,4000	0,9643	0,7500
#M002	6	4	6	2 ^a	0,8571	0,9300	0,6000	0,2000	0,6667	0,6667
#M003	3	2	10	3 ^a	0,4286	0,8667	1,0000	0,2000	0,5833	0,6667
#M004	2	1	10	4 ^a	0,2857	0,8000	1,0000	0,2000	0,3333	0,5000
#M005	4	1	9	5 ^a	0,5714	0,7333	0,9000	0,2000	0,4706	0,2500
#M006	1	1	10	6 ^a	0,1429	0,6667	1,0000	0,2000	0,3333	1,0000
#M007	7	3	7	7 ^a	1,0000	0,6000	0,7000	0,2000	0,5625	0,4286
#M008	0	0	10	8 ^a	0,0000	0,5333	1,0000	0,4000	0,7368	0,0000
#M009	1	0	5	9 ^a	0,1429	0,4667	0,5000	0,4000	0,5833	0,0000
#M010	4	3	10	10 ^a	0,5714	0,4000	1,0000	0,2000	0,5000	0,7500
#M011	2	0	10	11 ^a	0,2857	0,3333	1,0000	1,0000	0,6296	0,0000
#M012	1	0	10	12 ^a	0,1429	0,2667	1,0000	0,4000	0,6296	0,0000
#M013	2	2	10	13 ^a	0,2857	0,2000	1,0000	0,2000	0,4118	1,0000
#M014	0	0	10	14 ^a	0,0000	0,1333	1,0000	0,6000	0,6552	0,0000
#M015	1	0	10	15 ^a	0,1429	0,0667	1,0000	0,4000	0,6552	0,0000

Fonte: Autoria própria.

A aplicação de um teste de hipótese deve levar em consideração as contaminações da amostra, como discorreu Wohlin et al. (2000), as contaminações ocorrem quando um participante não compreendeu a dinâmica de execução do experimento, não aplicou com seriedade o procedimento experimental, ou os dados apresentam pontos anormais, gerando *outliers*. O índice de contaminação das amostras do coeficiente de correlação de *rank* de *Spearman's*, conforme aferido por Croux e Dehon (2010), vai aumentando gradativamente após 1% de contaminação. Isso significa que caso existam *outliers* superiores a 1% nas amostras, os dados correspondentes devem ser removidos. Para detectar os *outliers* das amostras capazes de contaminar o teste de hipótese, foi utilizada a estatística descritiva. Com isso, as amostras foram organizadas em *box splots*, como mostra a Figura 30, e os *outliers* revelados foram removidos.

Para executar o teste de hipótese foi utilizada uma ferramenta estatística, automatizando o cálculo do coeficiente de correlação de *rank* de *Spearman's* e fornecendo maior confiabilidade dos dados gerados. A ferramenta apresenta ao final do cálculo o valor do coeficiente de correlação (nomeado *Rô de Spearman*) e o *Valor – P*. Um exemplo da execução do cálculo executado pela ferramenta é apresentado a seguir:

Rô de Spearman

Correlações: índice de execução e índice de ordenação

Rô de Spearman 0,436

Valor-P 0,119

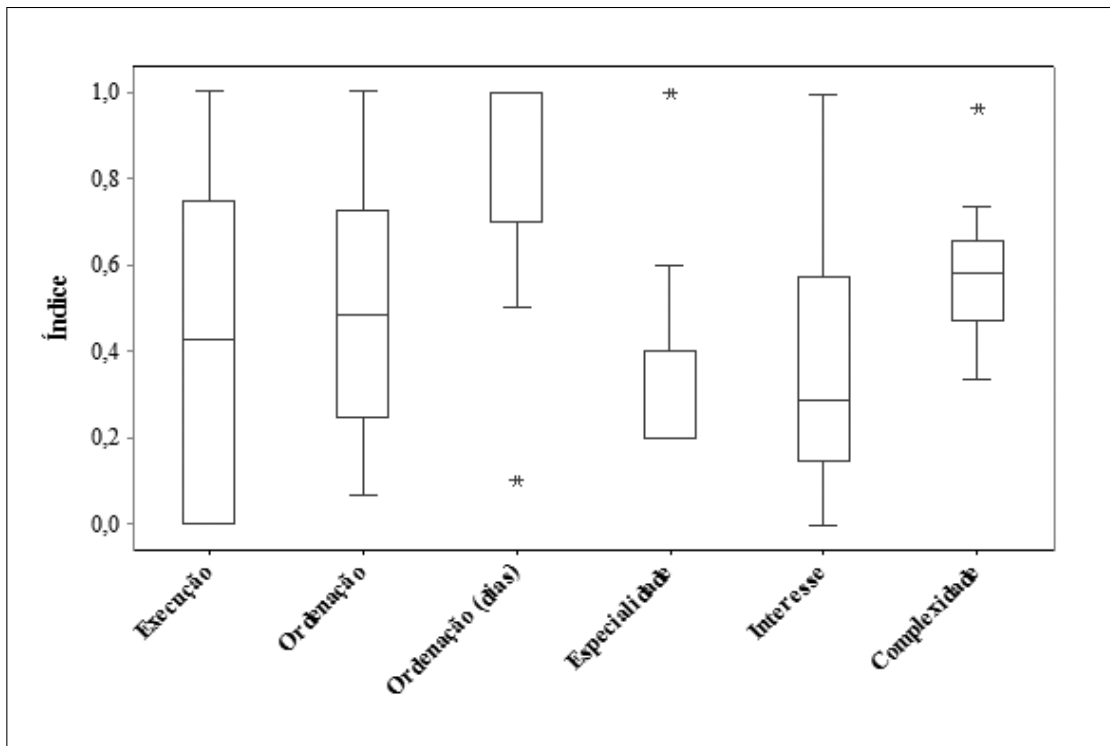


Figura 30: A detecção de *outliers* das amostras

Fonte: Autoria própria

Para executar o teste, a amostra do índice de execução foi comparada com as demais amostras (índice de interesse, de experiência, de ordenação e de complexidade). Para considerar o índice de correlação foi adotado os coeficientes propostos por Heiman (2011), que classifica quatro níveis de correlação:

- $-0.20 \geq R\hat{o} \text{ de Spearman} \leq +0.20$: o coeficiente é considerado fraco.
- $(-0.20 > R\hat{o} \text{ de Spearman} \leq -0.60)$ ou $(+0.20 > R\hat{o} \text{ de Spearman} \leq +0.60)$: o coeficiente é considerado forte.
- $(-0.60 > R\hat{o} \text{ de Spearman})$ ou $(R\hat{o} \text{ de Spearman} > +0.60)$: o coeficiente é considerado extremamente forte.
- $R\hat{o} \text{ de Spearman}$ próximo ou igual a 1: a raridade de obter esse coeficiente pode atestar, na verdade, um erro operacional ou computacional.

Em síntese, as amostras analisadas demonstraram três classificações distintas. Uma amostra demonstrou possuir fraca correlação com índice de execução, enquanto que duas amostras atestaram existir uma correlação forte, e as demais amostras foram consideradas extrema-

mente fortes. Assim, os resultados obtidos pelo índice de correção das amostras foram os seguintes:

- **Correlação fraca:** A correlação entre o índice de execução e a ordenação em dias foi considerada fraca. O *Rô de Spearman* obtido foi de 0,063, e com isso, o coeficiente obtido não foi suficiente para demonstrar correlação.
- **Correlação forte:** A ordenação natural das *microtasks* obteve um desempenho forte de correlação. O *Rô de Spearman* obtido foi de 0,436, e com isso, pode-se notar a existência de uma correlação entre a ordem que as *microtasks* se encontram e o índice de execução. Ou seja, existe uma tendência natural dos participantes do *crowd* selecionarem as *microtasks* que se encontram em destaque ou a frente de outras *microtasks*. O interesse dos participantes também demonstrou forte indício de correlação com o índice de execução. O *Rô de Spearman* obtido foi de 0,518, e assim, percebe-se que conforme maior o número de participantes registrados, existe uma tendência de crescimento do número de participantes que deverão contribuir para a *microtask*.
- **Extremamente forte:** A correlação entre a experiência necessária de uma *microtask* e o seu índice de execução foi considerada extremamente forte. Nesse caso, o *Rô de Spearman* foi de $-0,635$. Com isso, compreende-se que conforme diminuiu a especialidade necessária para executar uma *microtask*, aumentou a tendência dessa *microtask* ser executada pelo *crowd*. A abordagem proposta para aferir a complexidade de uma *microtask* também foi considerada extremamente forte com índice de execução. Além disso, a abordagem obteve o melhor desempenho no teste de hipótese, sendo o *Rô de Spearman* obtido de $-0,644$. Com isso, pode-se notar que conforme a abordagem atestava simplicidade de uma *microtask* havia crescimento do seu índice de execução.

Seguindo as orientações de Heiman (2011), a amostra referente a ordenação (em dias) foi removida do teste de hipótese devido a fraca correlação. Para dar prosseguimento, e finalizar o cálculo do coeficiente de correlação de *rank* de *Spearman's*, foi utilizado o gráfico de dispersão. Como propôs Gibbons e Chakraborti (2011), esse gráfico é altamente recomendado para este teste estatístico adotado, visto que exhibe a dispersão amostral, tornando nítida e visual a correlação dos dados obtidos no cálculo.

Apesar de ser considerada forte a correlação entre as amostras de ordenação natural e interesse com a amostra do índice de execução, ficam visíveis os riscos dessa afirmação. Em síntese, ambas as amostras possuem um alto grau de dispersão, com pontos que não estão alinhados junto à linha de regressão do gráfico (onde deveriam se aglutinar para demonstrar coesão).

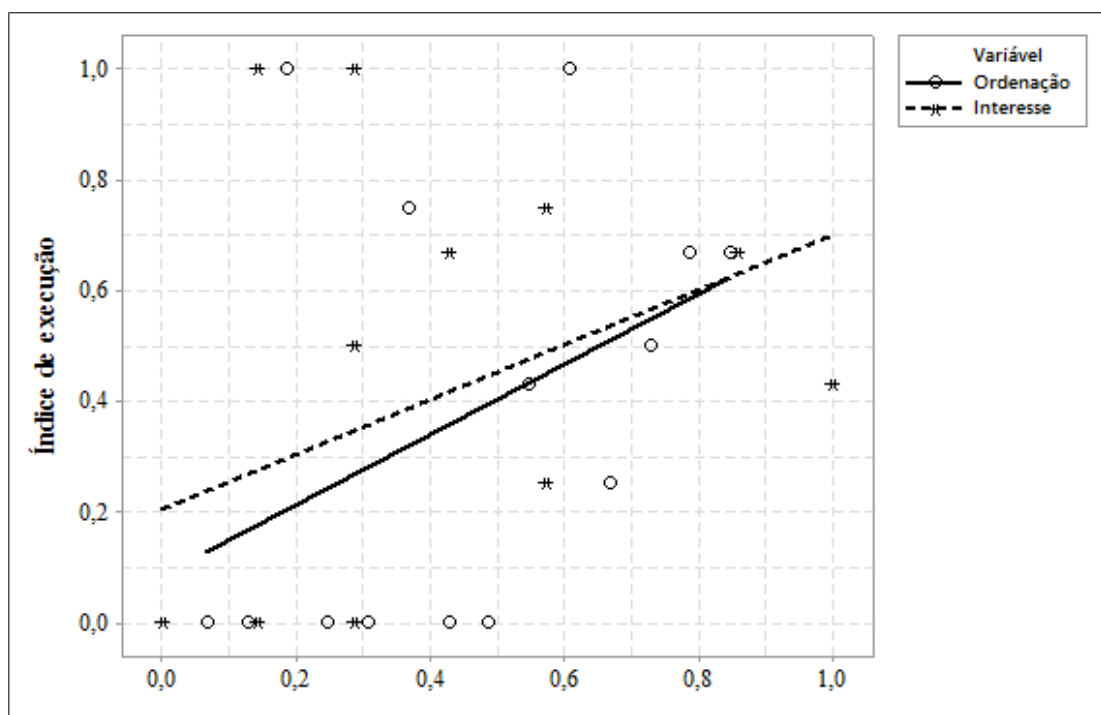


Figura 31: A dispersão das amostras de ordenação e de interesse que apresentaram correlação forte com o índice de execução

Fonte: Autoria própria

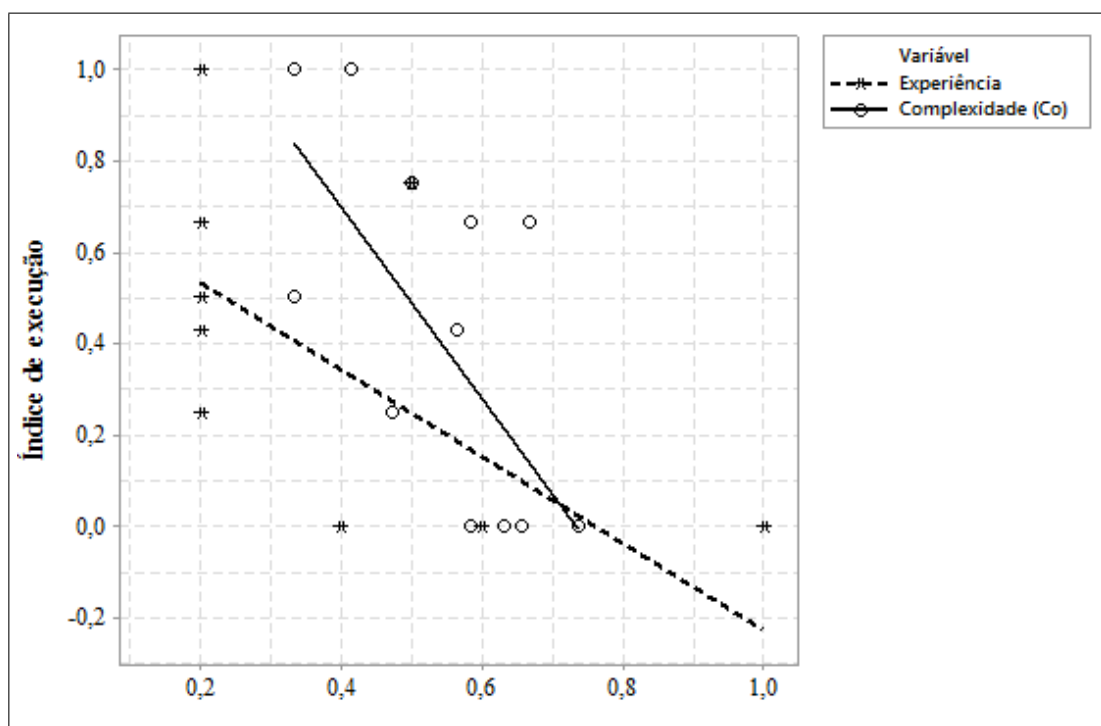


Figura 32: A dispersão das amostras de especialidade e de complexidade que apresentaram correlação extremamente forte com o índice de execução

Fonte: Autoria própria

Baseado em Heiman (2011), pode-se afirmar categoricamente que apesar da forte correlação, existem indícios da dispersão dos pontos em relação a linha de regressão, que tornam pouco acurada a relação dessas amostras com o índice de execução. Os indícios não são perceptíveis usando somente o teste estatístico, e tornam-se visualmente detectáveis somente sob o uso de recurso gráfico. O gráfico de dispersão foi apresentado na Figura 31.

A amostra sobre a especialidade e a complexidade (C_o) demonstraram uma pequena variação no *Rô de Spearman* obtido. Todavia, essa diferença reproduziu um conjunto de considerações importantes que foram detectadas após a visualização dos dados. A dispersão dos pontos de experiência alcançou uma grande amplitude, contrariando um aspecto da correlação que busca sempre a aproximação dos pontos. Todavia, a amplitude dos pontos da amostra de complexidade foram condensadas em uma área menor, próximas a linha de regressão. Isso representa que a amostra de complexidade obteve maior linearidade, alcançando, em termos de dispersão, resultado mais considerável. Ademais, a linha de regressão demonstra que a amostra de experiência obteve sete níveis de amplitude vertical, contra os oito níveis obtidos pela linha de regressão da amostra de complexidade. De acordo com as considerações de Heiman (2011), conforme maior o número de níveis que a linha de regressão demonstra, maior é a correlação entre as amostras. O gráfico de dispersão das amostras com correlação extremamente forte foi ilustrado na Figura 32.

Por meio dos resultados obtidos pelo teste de hipótese, usando o coeficiente de correlação de *rank* de *Spearman's* e a estatística descritiva foi determinado que existe uma correlação extremamente forte entre o índice de execução e a abordagem proposta para mensurar a complexidade de uma *microtask*. Com isso, identificou-se uma correlação em que o crescimento do índice de complexidade diminui o índice de execução das *microtasks*. Esses resultados foram estatisticamente significante para o ponto de corte 0,05, em que o *Rô de Spearman* obtido foi de $-0,644$ e o *Valor - P* foi de 0,013. Com isso, pode-se refutar com um intervalo de confiança de 95% a H_0 .

4.4.8 RESUMO E DISCUSSÕES

As métricas existentes de aferição de complexidade não podem ser aplicadas nas *microtasks* devido à um conjunto de especificidades do CS. Dessa forma, foi apresentada uma abordagem capaz de converter as características e os estados assumidos de uma *microtasks* a fim de mensurar a sua complexidade.

Para conduzir a avaliação empírica foi utilizado o teste estatístico de correlação de *rank* de *Spearman's* e a estatística descritiva. O teste estatístico demonstrou existir uma correlação

extremamente forte com o índice de execução e a abordagem proposta. A estatística descritiva, por sua vez, apresentou a distribuição dos dados analisados por meio de gráficos de dispersão. Com essas análises foi possível realizar o teste de hipótese e afirmar a H_1 , destacando a eficiência em aferir a complexidade de uma *microtask* com base na abordagem proposta.

Todavia, algumas considerações devem ser expostas. Este trabalho demonstrou (ver Figura 18, página 64) que o *crowd* apresenta grande heterogeneidade quando sua amplitude cresce. Assim, *crowds* populosos podem ser menos eficazes que *crowds* menores. Como o experimento controlado limitou o total de participantes, não foi possível simular a execução de *microtasks* por *crowds* considerados populosos. Ademais, percebe-se também que houveram amostras com correlação forte e extremamente forte com o índice de execução. Contudo, essas amostras foram menos eficazes que a abordagem proposta sobre aferição de complexidade. Assim, pode-se identificar a existência de algumas tendências que podem influenciar a execução de *microtask* além da sua própria complexidade.

4.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma síntese sobre a caracterização das *microtasks* no desenvolvimento de software CS.

Na Seção 4.1 foi apresentado o resultado da investigação sobre o uso das *microtasks* nos projetos de software. Um conjunto de dados formado por 14485 *microtasks* foi utilizado para extrair o tipo do uso das *microtasks*, a aplicação no ciclo de desenvolvimento, as plataformas e as tecnologias empregadas, além da relação entre o índice de sucesso e falha. Também foi analisado o impacto do CS na execução das *microtasks*. Nesse sentido percebeu-se que o CS empregou uma tendência homogênea em *crowds* com até 100 participantes. Logo, o tamanho do *crowd*, a taxa de submissão, o valor do prêmio e o tempo disponível foram similares e possuíam uma certa linearidade. Entretanto, os *crowds* maiores concentraram uma amplitude superior de desafios, e as tendências percebidas não foram aplicáveis. Além disso, também merece destaque quatro considerações sobre desafios identificados nas *microtasks* analisadas. Primeiro, o caso analisado exibiu que as *microtasks* podem ser aplicadas em qualquer etapa do ciclo de vida de um projeto de software. Segundo, o alto índice de *microtasks* que não apresentaram nenhum tipo de auxílio (endereço, e-mail ou repositório) para a sua execução. Terceiro, o equívoco por parte do *crowdsourcer* na etapa de cadastrar a categoria da *microtask*, atestando falta de fiscalização por parte da plataforma. Quarto, a ausência da definição de alguma tecnologia para a execução das *microtasks*. Contudo, apesar desse cenário, que poderia induzir resultados insatisfatórios, quase 87% das *microtasks* foram finalizadas com sucesso. Isso

demonstrou que a abordagem do desenvolvimento de software CS com *microtasks*, apesar de custosa e desafiadora, ainda proporcionou alto grau de segurança.

A Seção 4.2 apresentou o conjunto de características que cada *microtask* pode possuir. E com isso, a grande contribuição fornecida neste pilar da pesquisa foi a taxonomia gerada sobre as características das *microtasks*, sistematizando diversos conceitos detectados na literatura e no estudo de caso. Além disso, também foi exposto que as características podem assumir diversos estados, variando conforme a natureza de cada *microtask* e acrescentando uma gama maior de informação. Em vista desse cenário, foi gerado o metamodelo de uma *microtask* que apresenta as ligações de cada característica de uma *microtask*. O metamodelo pode ser aplicado em diferentes formas, sendo capaz de nortear o *crowdsourcer* na estruturação de uma *microtask* e auxiliar o *crowd* na percepção.

O foco da Seção 4.3 foi conduzir uma investigação sobre como as *microtasks* podem ser comparadas com as atividades do DDS. Os papéis análogos e compatíveis entre DDS e CS foram analisados. E nesse sentido foi destacado que o *crowdsourcer* possui um conjunto maior de responsabilidades que tornam oneroso a sua carga de trabalho quando comparado ao gerente DDS. Isso ocorreu em virtude do gerente DDS possuir apoio dos *sites* em diversas tarefas, cenário que não aconteceu entre o *crowdsourcer* e o *crowd*. Além de que o processo CS apresentou maiores dificuldades operacionais devido à constante atuação do *crowdsourcer*. Apesar disso, o DDS empregou um papel representado pelo cliente, que tornou a execução de um projeto mais desafiadora. Em suma, as *microtasks* forneceram maior liberdade de planejamento e execução, ao momento que as atividades de software foram mais inflexíveis e dependentes.

A complexidade das *microtasks* foi investigada na Seção 4.4. Para isso, foram gerados três conjuntos referentes às características e estados de uma *microtask*. Logo após, os elementos dos conjuntos foram convertidos e quantificados. Com base nas evidências captadas, a abordagem proposta para verificar a complexidade de uma *microtask* foi eficaz. Os resultados demonstraram que as *microtasks* com maior simplicidade foram selecionadas e executadas pelos *participantes* do *crowd* enquanto que as *microtasks* mais complexas não tiveram esse comportamento. Porém, deve-se destacar que existiram relações além da complexidade, como a experiência e o interesse dos participantes, que também impactaram o processo de execução.

4.5.1 AMEAÇAS À VALIDADE

Foram identificadas diferentes ameaças à validade deste estudo. Para sintetizá-las adotou-se a sumarização proposta por Wohlin et al. (2000) que classifica as ameaças de acordo com a validade de construção, interna, externa e de confiabilidade/conclusão.

Validade de construção: está relacionada com a teoria proposta e a observação realizada. Nesse sentido, destaca-se que para gerar as QPs do estudo de caso foram utilizados trabalhos que investigavam ou identificavam as lacunas sobre a aplicação de *microtasks*. Todavia, a relevância e atualidade desses trabalhos não foram discutidas devida à rápida atualização da literatura sobre CS e *microtasks*. Para a experimentação controlada, foram conduzidos previamente dois testes experimentais antes da realização do experimento. Contudo, alguns aspectos proporcionados pelo CS não puderam ser reproduzidos, como a participação anônima.

Validade interna: refere-se a existência de fatores, desconhecidos ou desconsiderados, que podem estar atuando na análise. Nessa perspectiva, a principal ameaça à validade interna do caso analisado reside no processo de extração de dados da plataforma CS. Devido a automatização do processo de extração, informações com importância significativa podem ter sido omitidas ou negligenciadas. Ao ponto que as principais ameaças à validade da experimentação controlada residem no teste de hipótese realizado. Foram consideradas as opiniões dos participantes para investigar as correlações do índice de execução de uma *microtask*. Contudo, foram obtidas diferentes classificações de correlação, e isso demonstra que apesar da abordagem de complexidade ser o mais significativo, existem outras dependências.

Validade externa: é relativa a generalização dos resultados obtidos. Assim, no estudo de caso conduzido foi atestado que, apesar de ser representativo o conjunto de dados analisados, não houve comparação com dados de outras plataformas. Na experimentação controlada existiram diversos aspectos que podem dificultar a generalização dos resultados, como o número limitado de participantes, o prazo das *microtasks*, e o total de *microtasks* utilizadas.

Confiabilidade/Validade de conclusão: para o estudo de caso a confiabilidade representa que os dados e a análise são dependentes e podem ser reproduzidos. Em vista desse cenário, percebe-se a dependência do fornecimento de dados pela plataforma. Com isso, a extração de novas informações pode ser considerada arbitrária e modificada sem aviso prévio. A validade de conclusão da experimentação está relacionada com o teste utilizado, a significância obtida e a ligação com os dados analisados. Assim, a principal ameaça reside na dificuldade de classificar o coeficiente correlação do *Rô de Spearman*. Existem diferentes propostas disponíveis na literatura, porém, este estudo considerou apenas a classificação de Heiman (2011).

5 CONCLUSÕES

Este capítulo encerra o trabalho apresentando uma síntese sobre a caracterização das *microtasks*, as contribuições, as limitações da abordagem e a perspectiva de caminhos futuros. Os capítulos anteriores reuniram resumos e discussões concentradas, entretanto, este capítulo busca apresentar uma discussão geral do trabalho realizado.

Desenvolver um software, ou parte dele, sempre foi uma tarefa sinuosa. Essa ação envolve um conjunto de papéis distintos, contrastes de visões e particularidades que adicionam diversas singularidades em cada projeto. Apesar desse cenário aparentemente caótico, o avanço da tecnologia tornou a ação de desenvolvimento de software aprimorada conforme a necessidade e a natureza de cada projeto.

Em 2006 quando o termo CS foi usado pela primeira vez para definir um novo modelo de negócios, o setor produtivo de software foi alertado para uma nova possibilidade no desenvolvimento de seus produtos. Apesar da existirem modelos com alta similaridade, o desenvolvimento de software CS foi adaptado e responsável por criar uma nova tendência. Em vista disso, a tradição de separar cliente e responsável técnico de um projeto em papéis distintos foi adaptado para um único papel denominado *crowdsourcer* e o time de desenvolvimento tornou-se “*crowd*”.

O papel do *crowdsourcer* é o ponto de partida de um projeto de software CS, reunindo o investimento, a organização, a estrutura e o responsabilidade para a condução do projeto. Na ponta final encontra-se o *crowd*, formado por trabalhadores *online*, contratados eletronicamente e de forma anônima.

A ligação entre *crowd* e *crowdsourcer* ocorre por meio das *microtasks*, em síntese, uma terminologia para identificar as atividades de software CS. Porém, a literatura sobre *microtask* ainda é muito incipiente e existem diversos exemplos de autores que se contradizem ao defini-las. Além disso, percebe-se também a ausência de análises centradas em *microtasks*. Nesse sentido, se destaca que a própria terminologia ainda não é comum, sendo utilizado termos como “*task*”, “*tasks CS*” ou antagonicamente “*macro tasks*”.

Em vista desse cenário, a condução deste trabalho buscou fornecer contribuições teóricas e empíricas sobre as *microtasks*. A investigação realizada foi baseada em quatro pilares, referentes ao uso, as características, a comparação e a complexidade das *microtasks*. Em união tais pilares simbolizam a caracterização das *microtasks*, e apresentaram diversas contribuições, que são revisitadas a seguir.

5.1 CONTRIBUIÇÕES

O desenvolvimento deste trabalho proporcionou uma ampla compreensão sobre como são caracterizadas as *microtasks* e as principais contribuições são destacadas a seguir:

- **Uso:** os resultados apresentados durante a investigação sobre o uso das *microtask* revelaram considerações importantes para a compreensão do desenvolvimento de software CS. Inicialmente, destaca-se a amplitude de aplicação das *microtasks*, fornecendo suporte desde a análise até a integração de um projeto CS. Apesar de existirem desafios, as *microtasks* analisadas alcançaram um alto índice de sucesso na finalização, demonstrando que o desenvolvimento de software CS é uma perspectiva interessante e segura para diversos cenários. Além disso, também foi exposto que as *microtasks* fornecem suporte para todas as etapas do ciclo de vida de um projeto ou produto de software. Demonstrando a portabilidade de sua utilização.
- **Características:** as características e os estados das *microtasks* possibilitaram um método eficiente para auxiliar a sua compreensão, realizar comparações e aferir a sua complexidade. Devido a sua identificação foi possível nortear este trabalho, revelando sua indispensável contribuição. Ademais, também foi possível gerar o metamodelo de uma *microtask*, que pode ser um mecanismo útil para a rotina do *crowdsourcer* e também para os participantes do *crowd*. Finalmente, devido à escassez de estudos sobre o tema, a taxonomia das características das *microtasks* representou uma valorosa contribuição para compreender a estrutura de uma *microtask*.
- **Contrastes:** não foram detectados estudos focados na comparação entre as *microtasks* e atividades de software no MS. Com isso, é evidenciado o caráter original da investigação conduzida comparando as *microtasks* com as atividades de software do DDS. Nesse sentido, os resultados obtidos podem auxiliar na compreensão dos benefícios e desafios das *microtasks* e atividades sob o ponto de vista de diferentes papéis do desenvolvimento de software.

- **Complexidade:** a investigação conduzida sobre a complexidade revelou uma abordagem para identificar o nível de dificuldade que as características e estados produzem em uma *microtask*. De acordo com as análises efetuadas, a complexidade é o principal aspecto que torna uma *microtask* executável pelo *crowd*. Porém, fatores concomitantes também podem interferir. De todo modo, devido a escassez de estudos na literatura sobre complexidade das *microtasks*, a contribuição apresentada e avaliada representa uma ferramenta útil para a rotina do *crowdsourcer* e para o trabalho do *crowd*.

Por fim, grande parte das contribuições e achados revelados por esta pesquisa podem ser generalizados para diversos cenários. As características são encontradas em todas, ou pelo na maioria das *microtasks*. Isso também ocorre nos resultados revelados pelos contrastes das *microtasks* com as atividades de software devido a análise de configurações distintas do modelo DDS. Enquanto que a abordagem proposta para aferir a complexidade foi planejada levando em consideração a abrangência de aplicação das *microtasks* em diferentes etapas do ciclo de vida de um projeto de software. Contudo, apesar da generalização de resultados, este trabalho também possui um conjunto de limitações que devem ser expostas.

5.2 LIMITAÇÕES

As restrições deste trabalho estão relacionadas aos métodos e procedimentos aplicados para a sua condução, bem como os próprios resultados obtidos.

Inicialmente, destaca-se a utilização de duas abordagens distintas para a condução do trabalho: estudo de caso e experimentação controlada. Assim, deve-se atestar que não foi realizado nenhum estudo exploratório ou teste inicial para a condução do estudo de caso. Nesse sentido, o protocolo não pode ser aferido antes de sua aplicação para corrigir possíveis desvios. A experimentação controlada foi aferida em dois momentos utilizando testes experimentais, entretanto, deve-se ressaltar a dificuldade em realizar uma simulação do CS considerando aspectos como a participação anônima e a comunicação assíncrona. Ademais, algumas análises deste trabalho foram obtidas com base nos resultados do MS. Assim, existe uma restrição direta com a sistematização realizada e os estudos examinados.

A investigação conduzida por este trabalho foi relacionada à quatro pilares de uma *microtask*: uso, características, contrastes e complexidade. E assim, cada pilar possui um conjunto de limitações. Quanto ao uso das *microtasks*, as análises ficaram sujeitas aos dados de uma única plataforma CS, restringindo a generalização dos resultados. Sobre as características de uma *microtask*, existe uma restrição direta referente a sistematização de informações presen-

tes na literatura e no estudo de caso. Além disso, o metamodelo proposto para uma *microtask* pode apresentar limitações conforme diferentes necessidades. E por isso, deve ser considerado como um mecanismo de suporte, e não um conjunto de regras obrigatórias. Essas limitações igualmente são encontradas nos contrastes entre *microtasks* e atividades de software DDS. Ou seja, os resultados apresentaram uma sistematização da informação e, devido as diferentes configurações que o CS e o DDS podem assumir, determinados cenários não foram contemplados. Enquanto que a abordagem proposta para calcular a complexidade de uma *microtask* foi avaliada por um número restrito de participantes.

Em resumo pode-se perceber que este trabalho possui um claro conjunto de fatores limitantes. Todavia, as restrições aqui apresentadas podem evidenciar a condução de novas investigações, reveladas a seguir.

5.3 LACUNAS E TRABALHOS FUTUROS

Existem diversas oportunidades de criação e disseminação de pesquisas sobre as *microtasks*. Assim, destaca-se a possibilidade de conduzir estudos de casos em outras plataformas de desenvolvimento de software CS, investigando como ocorre a aplicação de *microtasks*. Além disso, a condução de um *survey* com especialistas que estudam e profissionais que utilizam o desenvolvimento de software CS pode refletir uma pesquisa valiosa para compreender novas as características e estados de uma *microtask*. Também existe a possibilidade de confrontar as *microtasks* com as atividades *open source* ou do desenvolvimento ágil. E nesse sentido, surge a necessidade de investigar se as *microtasks* podem ser adotadas em outros modelos além do desenvolvimento de software CS. Por fim, a complexidade de uma *microtask* ainda pode apresentar novas análises, como a criação de ferramentas para automatização do cálculo.

De todo modo, tendo em vista o cenário produtivo de software, com grande demanda do uso do CS e suas *microtasks*; com o cenário acadêmico, repleto de terminologias distintas e contradições; existe uma área fecunda que pode guiar diversas investigações.

REFERÊNCIAS

- ABREU, F. B.; CARAPUÇA, R. Object-oriented software engineering: Measuring and controlling the development process. In: INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, 4., 1994, McLean. **Anais...** McLean, 1994. p. 1–8. Versão revisada. Disponível em: <<https://pdfs.semanticscholar.org/af59/c743d532c1f9682615e9f6287856a0de5834.pdf>> acesso em 04 jan. 2018.
- AFRIDI, H. G. Empirical investigation of correlation between rewards and crowdsource-based software developers. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING COMPANION, 39., 2017, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2017. p. 80–81.
- ALBRECHT, A. Measuring application development productivity. **IBM Application Development Symposium**, p. 83–92, 1979.
- ALELYANI, T.; YANG, Y. Software crowdsourcing reliability: An empirical study on developers behavior. In: INTERNATIONAL WORKSHOP ON SOFTWARE ANALYTICS, 2., 2016, Paderborn. **Anais...** Paderborn: ACM, 2016. p. 36–42.
- ALI, T.; NASR, E. S.; GHEITH, M. Self-management of distributed computing using hybrid-computing elements. In: AFRICA AND MIDDLE EAST CONFERENCE ON SOFTWARE ENGINEERING, 2., 2016, Cairo. **Anais...** Cairo: ACM, 2016. p. 15–20.
- ALLAHBAKHS, M. et al. Quality control in crowdsourcing systems: Issues and directions. **IEEE Internet Computing**, Washington, v. 17, n. 2, p. 76–81, Mar. 2013.
- ARIS, H.; DIN, M. M. Crowdsourcing evolution: Towards a taxonomy of crowdsourcing initiatives. In: INTERNATIONAL WORKSHOP ON BENCHMARKS FOR UBIQUITOUS CROWDSOURCING: METRICS, METHODOLOGIES, AND DATASETS, 1., 2016, Sydney. **Anais...** Sydney: IEEE, 2016. p. 1–6.
- BABA, Y.; KASHIMA, H. Statistical quality estimation for general crowdsourcing tasks. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 19., 2013, Chicago. **Anais...** Chicago: ACM, 2013. p. 554–562.
- BASIL, V. R. The past, present, and future of experimental software engineering. **Journal of the Brazilian Computer Society**, Rio de Janeiro, v. 12, n. 3, p. 7–12, Out. 2006.
- BESSAI, K.; CHAROY, F. Business process tasks-assignment and resource allocation in crowdsourcing context. In: INTERNATIONAL CONFERENCE ON COLLABORATION AND INTERNET COMPUTING, 2., 2016, San Jose. **Anais...** San Jose: IEEE, 2016. p. 11–18.
- BHATIA, J.; BREAUX, T. D.; SCHAUB, F. Mining privacy goals from privacy policies using hybridized task recomposition. **Transactions on Software Engineering and Methodology**, ACM, New York, v. 25, n. 3, p. 22:1–22:24, 2016.

BICK, S. et al. Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings. **IEEE Transactions on Software Engineering**, Washington, PP, n. 99, p. 1–21, 2017.

BOEHM, B. W. **Software engineering economics**. New Jersey: Prentice-hall Englewood Cliffs, 1981. 767 p. ISBN 978-0-138-22122-5.

BURTON, M. D.; NICHOLAS, T. Prizes, patents and the search for longitude. **Explorations in Economic History**, Amsterdam, v. 64, n. Supplement C, p. 21 – 36, 2017.

CHIDAMBER, S. R.; KEMERER, C. F. Towards a metrics suite for object oriented design. In: CONFERENCE PROCEEDINGS ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS, 1991, Phoenix. **Anais...** New York: ACM, 1991. p. 197–211.

CLELAND-HUANG, J. et al. Toward actionable, broadly accessible contests in software engineering. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 34., 2012, Zurique. **Anais...** Zurique: IEEE, 2012. p. 1329–1332.

CONSUELO, L. **An Experiment in Microtask Crowdsourcing Software Design**. 2016. 120 f. Dissertação (Mestrado em Engenharia de Software) — UNIVERSITY OF CALIFORNIA, IRVINE, 2016.

CROUX, C.; DEHON, C. Influence functions of the spearman and kendall correlation measures. **Statistical Methods & Applications**, v. 19, n. 4, p. 497–515, Nov. 2010.

DENG, X.; JOSHI, K.; GALLIERS, R. D. The duality of empowerment and marginalization in microtask crowdsourcing: Giving voice to the less powerful through value sensitive design. **Management Information Systems Quarterly**, Minnesota, v. 40, n. 2, p. 279–302, Jun. 2016.

DEUS, W. S. de; BARROS, R. M.; L'ERARIO, A. Um modelo para o gerenciamento do crowdsourcing em projetos de software. In: WORKSHOP SOBRE ASPECTOS SOCIAIS, HUMANOS E ECONÔMICOS DE SOFTWARE, 1., 2016, Maceió. **Anais...** Maceió: LDBD, 2016. p. 1–10.

DEUS, W. S. de; FABRI, J. A.; L'ERARIO, A. The management of crowdsourcing software projects: A systematic mapping. In: IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGIES, 12., 2017, Lisboa. **Anais...** Lisboa: IEEE, 2017. p. 1809–1813.

DEUS, W. S. de; FABRI, J. A.; L'ERARIO, A. The use of microtasks in crowdsourcing software development. In: IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGIES (CISTI), 12., 2017, Lisboa. **Anais...** Lisboa: IEEE, 2017. p. 1112–1117.

DEUS, W. S. de; FABRI, J. A.; L'ERARIO, A. The use of microtasks in crowdsourcing software development. **Journal on Advances in Theoretical and Applied Informatics**, Marília, v. 3, n. 1, p. 25–30, Out. 2017.

DEUS, W. S. de et al. Enhancing collaboration among undergraduates in informatics: A teaching and learning process based on crowdsourcing. In: FRONTIERS IN EDUCATION CONFERENCE, 47., 2017, Indianapolis. **Anais...** Indianapolis: IEEE, 2017. p. 1–8.

DUBEY, A. et al. Dynamics of software development crowdsourcing. In: INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING, 11., 2016, Orange County. **Anais...** Orange County: IEEE, 2016. p. 49–58.

DWARAKANATH, A. et al. Crowd build: A methodology for enterprise software development using crowdsourcing. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 2., 2015, Florence. **Anais...** Florence: IEEE, 2015. p. 8–14.

ENGELHARDT, M.; BAIN, L. J. On the mean time between failures for repairable systems. **IEEE Transactions on Reliability**, v. 35, n. 4, p. 419–422, Oct, 1986.

FAISAL, M. I. Predicting the quality of contests on crowdsourcing-based software development platforms: Student research abstract. In: SYMPOSIUM ON APPLIED COMPUTING, 32., 2017, Marrakech. **Anais...** Marrakech: ACM, 2017. p. 1305–1306.

FARIAS, I. de; AO, N. G. de S. L.; MOURA, H. P. de. An empirical study of motivational factors for distributed software development teams. In: IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGIES, 12., 2017, Lisboa. **Anais...** Lisboa: IEEE, 2017. p. 1–6.

FENG, Y. et al. Test report prioritization to assist crowdsourced testing. In: JOINT MEETING ON FOUNDATIONS OF SOFTWARE ENGINEERING, 10., 2015, Bergamo. **Anais...** Bergamo: IEEE, 2015. p. 225–236.

FENG, Y. et al. Multi-objective test report prioritization using image understanding. In: INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING, 31., 2016, Singapore. **Anais...** Singapore: ACM, 2016. p. 202–213.

FRASER, G. Gamification of software testing. In: INTERNATIONAL WORKSHOP ON AUTOMATION OF SOFTWARE TESTING, 12., 2017, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2017. p. 2–7.

GADIRAJU, U.; YANG, J.; BOZZON, A. Clarity is a worthwhile quality: On the role of task clarity in microtask crowdsourcing. In: PROCEEDINGS OF THE 28TH ACM CONFERENCE ON HYPERTEXT AND SOCIAL MEDIA, 28., 2017, Nova York. **Anais...** Nova York, 2017. p. 5–14.

GIBBONS, J. D.; CHAKRABORTI, S. Nonparametric statistical inference. In: _____. **International encyclopedia of statistical science**. Berlin: Springer, 2011. p. 977–979. ISBN 978-3-642-04897-5.

GUPTA, S.; GOUTTAM, D. Towards changing the paradigm of software development in software industries: An emergence of agile software development. In: INTERNATIONAL CONFERENCE ON SMART TECHNOLOGIES AND MANAGEMENT FOR COMPUTING, COMMUNICATION, CONTROLS, ENERGY AND MATERIALS, 2., 2017, Chennai. **Anais...** Chennai: IEEE, 2017. p. 18–21.

HALSTEAD, M. H. **Elements of Software Science**. New York: Elsevier Science Inc., 1977. 142 p. (Operating and programming systems series, 1). ISBN 044-4-00205-7.

HEIMAN, G. **Basic Statistics for the Behavioral Sciences**. Belmont: Wadsworth, 2011. 504 p. ISBN 978-3-540-45523-3.

- HETMANK, L. Components and functions of crowdsourcing systems – a systematic literature review. In: INTERNATIONAL CONFERENCE ON WIRTSCHAFTSINFORMATIK, 11., 2013, Leipzig. **Anais...** Leipzig, 2013. p. 55–69.
- HODA, R. et al. Socio-cultural challenges in global software engineering education. **IEEE Transactions on Education**, Norman, v. 60, n. 3, p. 173–182, Aug 2017. ISSN 0018-9359.
- HOSSEINI, M. et al. The four pillars of crowdsourcing: A reference model. In: INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE, 8., 2014, Marrakesh. **Anais...** Marrakesh: IEEE, 2014. p. 1–12.
- HOWE, J. **Crowdsourcing: A Definition**. 2006. Disponível em: <<http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing-a.html>>. Acesso em 04 jan. 2018.
- HOWE, J. The rise of crowdsourcing. **Wired magazine**, São Francisco, n. 6, Jun. 2006. São Francisco, jun. 2006. Disponível em: <<https://www.wired.com/2006/06/crowds/>>. Acesso em 18 nov. 2017.
- JIANG, H.; MATSUBARA, S. Improving crowdsourcing efficiency based on division strategy. In: INTERNATIONAL CONFERENCES ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY, 2., 2012, Macau. **Anais...** Macau: IEEE, 2012. p. 425–429.
- JURISTO, N.; MORENO, A. M. **Basics of Software Engineering Experimentation**. New York: Incorporated Springer Publishing Company, 2010. 396 p. ISBN 978-1-441-95011-6.
- KHAN, A. R. et al. Systematic literature review and empirical investigation of barriers to process improvement in global software development: Client–vendor perspective. In: SYMPOSIUM ON COMPUTER APPLICATIONS INDUSTRIAL ELECTRONICS, 3., 2017, Langkawi. **Anais...** Langkawi: Butterworth-Heinemann, 2017. p. 7–12.
- KITTUR, A. et al. Crowdforge: Crowdsourcing complex work. In: ACM, 24., 2011, SANTA BARBARA. **Anais...** SANTA BARBARA, 2011. p. 43–52.
- LATOZA, T. D.; HOEK, A. v. d. A vision of crowd development. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 37., 2015, Florence. **Anais...** Florence: IEEE, 2015. p. 563–566.
- LATOZA, T. D.; HOEK, A. van der. Crowdsourcing in software engineering: Models, motivations, and challenges. **IEEE Software**, v. 33, n. 1, p. 74–80, Jan. 2016.
- LATOZA, T. D. et al. Ask the crowd: Scaffolding coordination and knowledge sharing in micro-task programming. In: SYMPOSIUM ON VISUAL LANGUAGES AND HUMAN-CENTRIC COMPUTING, 10., 2015, Atlanta. **Anais...** Atlanta: IEEE, 2015. p. 23–27.
- LATOZA, T. D. et al. Microtask programming: Building software with a crowd. In: SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 27., 2014, Charlotte. **Anais...** Charlotte: IEEE, 2014. p. 43–54.
- LATOZA, T. D. et al. Crowd development. In: INTERNATIONAL WORKSHOP ON COOPERATIVE AND HUMAN ASPECTS OF SOFTWARE ENGINEERING, 6., 2013, São Francisco. **Anais...** São Francisco: IEEE, 2013. p. 85–88.

LEANO, R.; WANG, Z.; SARMA, A. Labeling relevant skills in tasks: Can the crowd help? In: SYMPOSIUM ON VISUAL LANGUAGES AND HUMAN-CENTRIC COMPUTING, 17., 2016, Cambridge. **Anais...** Cambridge: IEEE, 2016. p. 185–189.

L'ERARIO, A. et al. Teaching crowdsourcing development in undergraduate courses a comparative study. In: IBERIAN CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGIES, 12., 2017, Lisboa. **Anais...** Lisboa: IEEE, 2017. p. 685–690.

LETHBRIDGE, T. C.; SIM, S. E.; SINGER, J. Studying software engineers: Data collection techniques for software field studies. **Empirical Software Engineering**, v. 10, n. 3, p. 311–341, Jul. 2005.

LI, N.; MO, W.; SHEN, B. Task recommendation with developer social network in software crowdsourcing. In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, 23., 2016, Hamilton. **Anais...** Hamilton: IEEE, 2016. p. 9–16.

LI, W. et al. **Crowdsourcing: Cloud-Based Software Development**. Berlin: Springer-Verlag Berlin Heidelberg, 2015. 270 p. ISBN 978-3-662-51249-4.

MACHADO, L. et al. Task allocation for crowdsourcing using ai planning. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 3., 2016, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2016. p. 36–40.

MACHADO, L. et al. The good, the bad and the ugly: An onboard journey in software crowdsourcing competitive model. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 4., 2017, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2017. p. 2–8.

MAHMOOD, S. et al. Key factors that influence task allocation in global software development. **Information and Software Technology**, Amsterdam, v. 91, n. Supplement C, p. 102 – 122, 2017.

MAO, K. et al. A survey of the use of crowdsourcing in software engineering. **Journal of Systems and Software**, Amsterdam, v. 15, n. 01, p. 57–84, 2017.

MAO, K. et al. Pricing crowdsourcing-based software development tasks. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 35., 2013, Piscataway. **Anais...** Piscataway: IEEE, 2013. p. 1205–1208.

MAO, K. et al. Developer recommendation for crowdsourced software development tasks. In: SYMPOSIUM ON SERVICE-ORIENTED SYSTEM ENGINEERING, 11., 2015, San Francisco Bay. **Anais...** San Francisco Bay: IEEE, 2015. p. 347–356.

MCCABE, T. J. A complexity measure. **IEEE Transactions on Software Engineering**, SE-2, n. 4, p. 308–320, Dec 1976.

MOODLEY, F.; BELLE, J. P. V.; HASTEER, N. Crowdsourced software development: Exploring the motivational and inhibiting factors of the south african crowd. In: INTERNATIONAL CONFERENCE ON CLOUD COMPUTING, DATA SCIENCE ENGINEERING - CONFLUENCE, 7., 2017, Noida. **Anais...** Noida: IEEE, 2017. p. 656–661.

MORRIS, R.; DONTCHEVA, M.; GERBER, E. Priming for better performance in microtask crowdsourcing environments. **IEEE Internet Computing**, Piscataway, v. 16, n. 5, p. 13–19, 2012. ISSN 1089-7801.

MURRAY-RUST, D.; SCEKIC, O.; LIN, D. Worker-centric design for software crowdsourcing: Towards cloud careers. In: _____. **Crowdsourcing: Cloud-Based Software Development**. Berlin: Springer, 2015. p. 39–50.

NAIK, N. Crowdsourcing, open-sourcing, outsourcing and insourcing software development: A comparative analysis. In: SYMPOSIUM ON SERVICE-ORIENTED SYSTEM ENGINEERING, 11., 2016, San Francisco. **Anais...** San Francisco: IEEE, 2016. p. 380–385.

NIAZI, M. et al. Challenges of project management in global software development: A client-vendor analysis. **Information and Software Technology**, Amsterdam, v. 80, n. Supplement C, p. 1 – 19, 2016.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering. **Information and Software Technology**, Newton, MA, USA, v. 64, n. C, p. 1–18, Ago. 2015.

RAYMOND, D. R.; TOMPA, F. W. Hypertext and the new oxford english dictionary. In: **Conference on Hypertext**. New York, NY, USA: ACM, 1987. p. 143–153.

ROBSON, C. **Real World Research: A Resource for Social Scientists and Practitioner-Researchers**. [S.l.]: Wiley, 2002. 624 p. (Regional Surveys of the World Series). ISBN 978-0-631-21305-5.

RUNESON, P.; HØST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical Software Engineering**, v. 14, n. 2, p. 131, Dez. 2008.

SAREMI, R. L.; YANG, Y. Dynamic simulation of software workers and task completion. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 2., 2015, Piscataway. **Anais...** Piscataway: IEEE, 2015. p. 17–23.

SAREMI, R. L.; YANG, Y. Empirical analysis on parallel tasks in crowdsourcing software development. In: INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING WORKSHOP, 30., 2015, Washington. **Anais...** Washington: IEEE, 2015. p. 28–34.

SAREMI, R. L. et al. Leveraging crowdsourcing for team elasticity: an empirical evaluation at topcoder. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING IN PRACTICE TRACK, 39., 2017, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2017. p. 103–112.

SCHILLER, T. W.; ERNST, M. D. Reducing the barriers to writing verified specifications. In: INTERNATIONAL CONFERENCE ON OBJECT ORIENTED PROGRAMMING SYSTEMS LANGUAGES AND APPLICATIONS, 12., 2012, Tucson. **Anais...** Tucson: ACM, 2012. p. 95–112.

STOCK, P. **Ships, clocks and stars: the quest for longitude**. Nova York: Harper Design, 2014.

- STOL, K.-J.; FITZGERALD, B. Researching crowdsourcing software development: Perspectives and concerns. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 1., 2014, Hyderabad. **Anais...** Hyderabad: IEEE, 2014. p. 7–10.
- STOL, K.-J.; FITZGERALD, B. Two's company, three's a crowd: A case study of crowdsourcing software development. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 36., 2014, Hyderabad. **Anais...** Hyderabad: ACM, 2014. p. 187–198.
- STOL, K. J.; LATOZA, T. D.; BIRD, C. Crowdsourcing for software engineering. **IEEE Software**, Washington, v. 34, n. 2, p. 30–36, Mar. 2017. ISSN 0740-7459.
- SUGANTHY, A.; CHITHRALEKHA, T. Application of crowdsourcing in software development. In: INTERNATIONAL CONFERENCE ON RECENT TRENDS IN INFORMATION TECHNOLOGY, 5., 2016, Chennai. **Anais...** Chennai: IEEE, 2016. p. 1–6.
- TAJEDIN, H.; NEVO, D. Value-adding intermediaries in software crowdsourcing. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 47., 2014, Waikoloa. **Anais...** Waikoloa: IEEE, 2014. p. 1396–1405.
- TRANQUILLINI, S. et al. Modeling, enacting, and integrating custom crowdsourcing processes. **Transactions on the Web**, New York, v. 9, n. 2, p. 1–43, 2015.
- TUNIO, M. Z. et al. Impact of personality on task selection in crowdsourcing software development: A sorting approach. **IEEE Access**, Piscataway, v. 5, p. 18287–18294, Set. 2017.
- VUKOVIC, M. Crowdsourcing for enterprises. In: CONGRESS ON SERVICES, 1., 2009, Los Angeles. **Anais...** Los Angeles: IEEE, 2009. p. 686–692.
- WANG, J.; IPEIROTIS, P.; PROVOST, F. **Quality-based Pricing for Crowdsourced Workers**. Maryland: University of Maryland, 2013. 13–46 p. (DO&IT Seminar Series, 1).
- WANG, J. et al. Local-based active classification of test report to assist crowdsourced testing. In: INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING, 31., 2016, Singapore. **Anais...** Singapore: IEEE, 2016. p. 190–201.
- WEIDEMA, E. R. Q. et al. Toward microtask crowdsourcing software design work. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 3., 2016, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2016. p. 41–44.
- WINKLER, D. et al. Improving model inspection with crowdsourcing. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 4., 2017, Buenos Aires. **Anais...** Buenos Aires: IEEE, 2017. p. 30–34.
- WOHLIN, C. et al. **Experimentation in Software Engineering: An Introduction**. Norwell: Kluwer Academic Publishers, 2000. 236 p. ISBN 0-7923-8682-5.
- YANG, Y. et al. Who should take this task?: Dynamic decision support for crowd workers. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 10., 2016, Bolzano. **Anais...** Bolzano: ACM, 2016. p. 8:1–8:10.
- YANG, Y.; SAREMI, R. Award vs. worker behaviors in competitive crowdsourcing tasks. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 9., 2015, Beijing. **Anais...** Beijing: IEEE, 2015. p. 1–10.

YIN, R. K. **Estudo de Caso: Planejamento e Métodos**. Porto Alegre: Bookman editora, 2015. 271 p. ISBN 978-1-452-24256-9.

YUE, S.; PILON, P.; CAVADIAS, G. Power of the mann–kendall and spearman’s rho tests for detecting monotonic trends in hydrological series. **Journal of Hydrology**, Amsterdam, v. 259, n. 1, p. 254–271, Mar. 2002.

ZANATTA, A. L. et al. Barriers faced by newcomers to software-crowdsourcing projects. **IEEE Software**, v. 34, n. 2, p. 37–43, Mar. 2017.

ZHAO, M.; HOEK, A. v. d. A brief perspective on microtask crowdsourcing workflows for interface design. In: INTERNATIONAL WORKSHOP ON CROWDSOURCING IN SOFTWARE ENGINEERING, 2., 2015, Florença. **Anais...** Florença: IEEE, 2015. p. 45–46.

ZHAO, S. et al. Towards effective developer recommendation in software crowdsourcing. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, 27., 2015, Pittsburgh. **Anais...** Pittsburgh: IEEE, 2015. p. 326–329.

ZHU, J.; SHEN, B.; HU, F. A learning to rank framework for developer recommendation in software crowdsourcing. In: ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, 22., 2015, New Delhi. **Anais...** New Delhi: IEEE, 2015. p. 285–292.

APÊNDICE A – NOME DAS FONTES RECUPERADAS NO MAPEAMENTO SISTEMÁTICO

Tabela 27: Nome das fontes

Sigla	Nome	Tipo
AMECSE	Africa and Middle East Conference on Software Engineering	Conferência
APSEC	Asia-Pacific Software Engineering Conference	Conferência
ASE	International Conference on Automated Software Engineering	Conferência
AST	International Workshop on Automation of Software Testing	Workshop
CHASE	International Workshop on Cooperative and Human Aspects of Software Engineering	Workshop
CIC	International Conference on Collaboration and Internet Computing	Conferência
CISTI	Conferência Ibérica de Sistemas e Tecnologias de Informação	Conferência
CSI-SE	International Workshop on CrowdSourcing in Software Engineering	Workshop
ESEC	Joint Meeting on Foundations of Software Engineering	Simpósio
ESEM	International Symposium on Empirical Software Engineering and Measurement	Simpósio
HICSS	International Conference on System Sciences	Conferência
ICGSE	International Conference on Global Software Engineering	Conferência
ICSE	International Conference on Software Engineering	Conferência
ICSEK	International Conference on Software Engineering and Knowledge	Conferência
ICW	International Conference on Wirtschaftsinformatik	Conferência
IEEE-IC	IEEE Internet Computing	Jornal
IEEE-S	IEEE Software	Revista
IWSA	International Workshop on Software Analytics	Workshop
JSS	Journal of Systems and Software	Jornal
OOPSLA	International Conference on Object oriented Programming Systems Languages and Applications	Conferência
RCIS	Research Challenges in Information Science	Conferência
SAP	Symposium on Applied Computing	Simpósio
SIGKDD	International Conference on Knowledge Discovery and Data Mining	Conferência
SOSE	Symposium on Service-Oriented System Engineering	Simpósio
SVLHCC	Symposium on Visual Languages and Human-Centric Computing	Simpósio
TOSEM	Transactions on Software Engineering and Methodology	Jornal
TWEB	Transactions on the Web	Jornal
UIST	Symposium on User Interface Software and Technology	Simpósio
VI-HCC	Symposium on Visual Languages and Human-Centric Computing	Simpósio
WASHES	Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software	Workshop
WCS	World Conference on Services	Conferência
WI-IAT	International Conferences on Web Intelligence and Intelligent Agent Technology	Conferência

Fonte: Autoria própria.

APÊNDICE B – RELAÇÃO COMPLETA DAS MICROTASKS UTILIZADAS NO EXPERIMENTO CONTROLADO

Tabela 28: *Microtasks* usadas no experimento controlado

<p>#M001 - Tipo [Desenvolvimento] Atividade: Adicionar campo e-mail Descrição: A tabela usuários deve ser alterada e possuir uma coluna para adicionar o e-mail. Além disso, a página xhtml responsável por cadastrar e exibir os usuários deverá ter um novo campo para cadastrar o e-mail do usuário Regras: -Alterar o script de banco de dados -Alterar a classe index.xhtml -Alterar a classe bean.index.java -Alterar a classe mode.users.java -Preencha o formulário</p>
<p>#M002 - Tipo [Desenvolvimento/Bug] Atividade: nome da classe Descrição:A classe bean.index.java está com o nome errado. Para padronizar, deve-se adicionar a primeira letra em maiúsculo. Logo, o nome deve ser alterado para bean.Index.java. Além da alteração, todas dependências que apresentarem erro deverão ser refatoradas. Um exemplo de refatoração a ser feita é explicitado no test.TestUser.java Regras: -Renomear index.java para Index.java -Refatorar o código -Preencha o formulário</p>
<p>#M003 - Tipo [Protótipo] Atividade: tela de login Descrição: A aplicação ainda não possui um sistema de login, entretanto, para iniciar essa mudança deve-se ser criado um protótipo. O protótipo deverá ser enviado em imagem .png e ter um tamanho fixo de 1200px [largura] x 700px [altura]. Na construção do protótipo leve em consideração que a tela deverá ter os campos "usuário", "senha", botão "logar" e botão "cadastrar". Regras: Tamanho 1200x700 -Campos: usuario, senha, botão logar, botão cadastrar -Submeter na pasta /003 usando o nome do usuário -Usuários que submeterem duas ou mais submissões para essa atividade terão ambas desconsideradas. -Preencha o formulário</p>
<p>#M004 - Tipo [Desenvolvimento] Atividade: nova classe de teste Descrição: Não existem testes para a classe bean.Requisitions.java dificultado o processo de verificação! É necessário criar uma classe de teste que verifique se o método save() está funcionando corretamente. Para desenvolver a classe de teste de requisições utilize o conceito da classe test.TestUser.java Regras: -Criar a classe test.TestRequisition.java -Criar o método test01 e testar o save() da classe bean.Requisitions.java -Preencha o formulário</p>
<p>#M005 - Tipo [Desenvolvimento] Atividade: método save da classe bean.Requisitions.java Descrição: O método save() na classe bean.Requisitions.java deve ser alterado e retornar um boolean (true/false) na inserção Regras: -Criar try/catch na classe no método save() da bean.Requisitions.java -Retornar true se a inserção finalizar e retornar false se houver algum erro durante a persistência! -Preencha o formulário</p>
<p>#M006 - Tipo [Desenvolvimento] Atividade: nova classe de teste Descrição: Não existem testes para a classe bean.Services.java dificultado o processo de verificação! É necessário criar uma classe de teste que verifique se o método save() está funcionando corretamente. Para desenvolver a classe de teste de requisições utilize o conceito da classe test.TestUser.java Regras: -Criar a classe test.TestService.java -Criar o método test01 e testar o save() da classe bean.Requisitions.java -Preencha o formulário</p>
<p>#M007 - Tipo [Desenvolvimento] Atividade: método save da classe bean.Services.java Descrição: O método save() na classe bean.Services.java deve ser alterado e retornar um boolean (true/false) na inserção Regras: -Criar try/catch na classe no método save() da bean.Services.java -Retornar true se a inserção finalizar e retornar false se houver algum erro durante a persistência! -Preencha o formulário</p> <p><i>Continua na próxima página...</i></p>

#M008 - Tipo [Desenvolvimento/Bug]

Atividade: botões "salvar"**Descrição:** Os botões utilizados no cadastro de usuários, serviços e requisições estão com um erro severo. Deve-se clicar duas vezes para adicionar um registro. Por vezes, isso acaba duplicando o cadastro. **Regras:** -Detectar o motivo do erro -Fazer a alteração nas classes xhtml -Preencha o formulário

#M009 - Tipo [Desenvolvimento/Bug]

Atividade: cadastrar usuário **Descrição:** Ao cadastrar um usuário, o campo "nome" permanece preenchido ao clicar no botão de cadastro. Para evitar erros futuros, deve-se limpar o campo "nome" após o cadastro. **Regras:** -Detectar o motivo do erro -Fazer a alteração nas classes xhtml -Preencha o formulário

#M010 - Tipo [Protótipo]

Atividade: Interface mobile do sistema **Descrição:** O sistema será exportado para ambientes móveis. Entretanto, atualmente ele não apresenta características responsivas. Logo, para evitar transtornos futuros é necessário criar os protótipos das novas interfaces moveis! Serão necessário enviar duas telas nessa atividade. A primeira tela com 700px [largura] x 600px [altura]. A segunda tela com 350px [largura] x 500px [altura] **Regras:** -Uma imagem tamanho [700x600] -Uma imagem tamanho [350x500] -Ambas as imagens em extensão .png -Submeter na pasta /010 usando o nome do usuário e tela -ç para [700x600] usar padrão nomeusuario-telatablet.png -ç para [350x500] usar padrão nomeusuario-telacelular.png -Usuários que submeterem três ou mais submissões para essa atividade terão ambas desconsideradas. -Preencha o formulário

#M011 - Tipo [Teste]

Atividade: teste em todo o sistema **Descrição:** O sistema possui diversos erros de usabilidade e programação. Agora é hora de identificar erros ainda não percebidos no sistema. Nessa atividade, considere usar a branch /master para os testes. Serão considerados erros de usabilidade (respostas inexatas do sistema durante sua manipulação como usuário) e de programação (falhas arquiteturais do projeto). Cada comentário será nessa microtask e será analisado pelo(s) orquestrador(es) do projeto. Caso seja considerado bug uma nova microtask será aberta. **Regras:** -Utilizar a brach /master -Detectar erros de usabilidade e de programação -Comentar o motivo do erro usando o padrão "erro", "local", "entrada" e "saída"

#M012 - Tipo [Desenvolvimento]

Atividade: envio de e-mail **Descrição:** O sistema deverá enviar um e-mail ao usuário alertando que uma nova requisição foi cadastrada. Considere nesse e-mail as seguintes características: -ç destinatário: e-mail do usuário responsável pelo serviço -ç assunto: "Nova requisição-ç texto: "Uma nova requisição foi cadastrada no TecService". Nessa atividade considere criar um novo método de teste em bean.TestUser.java **Regras:** -Enviar e-mail ao cadastrar requisição -Criar caso de teste para validar o envio de e-mail -Preencha o formulário

#M013 - Tipo [Desenvolvimento]

Atividade: deleção em cascata **Descrição:** O sistema possui três tabelas principais na pasta /Documentação. Entretanto, não existe nenhum modo de deleção em cascata. Considere alterar o create_tables.sql e adicionar regras de remoção em cascata, além disso, crie um novo arquivo nomeado "drop_tables.sql" que delete usando a nova estrutura. **Regras:** -Atualizar o create_tables.sql para deleção em cascata -Criar um novo arquivo chamado "drop_tables.sql" na pasta script que execute o comando de deleção -Utilize a branch /development -Preencha o formulário

#M014 - Tipo [Desenvolvimento]

Atividade: Paginação de usuários **Descrição:** A tela de usuários não apresenta paginação de listagem, logo ao inserir muitos usuários a página poderá apresentar demora de exibição. Nessa atividade considere usar o dataTable para realizar a paginação. **Regras:** -Realizar paginação de usuários a cada 10 registros -Criar um teste que insira 15 usuários e verifique se ocorre a paginação -Preencha o formulário

#M015 - Tipo [Desenvolvimento]

Atividade: Remoção de Requisições **Descrição:** A tela de requisição não possui nenhum mecanismo para remoção de registros. Considere criar essa funcionalidade. **Regras:** -Adicione uma nova coluna denominada "Ações" na listagem de requisições -Adicione um botão de remoção -Ao clicar no botão de remoção o registro deverá ser apagado do banco de dados e a listagem de requisições atualizada -Preencha o formulário
