Documentação de Software:
Exploração dos problemas mais frequentes e qualidades mais relevantes

Kelvin Williams Neves Bernardo

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, 2022

Kelvin Williams Neves Bernardo

Documentação de Software

Monografia apresentada ao curso Engenharia de Computação do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Derzu Omaia

Dezembro de 2022

Catalogação na publicação
Seção de Catalogação e Classificação

B523d Bernardo, Kelvin Williams Neves.
Documentação de Software: Exploração dos problemas mais frequentes e qualidades mais relevantes / Kelvin Williams Neves Bernardo. - João Pessoa, 2022.
36 f. : il.

Orientação: Derzu Omaia.
TCC (Graduação) - UFPB/CI.

1. Documentação de software. 2. Problemas de documentação. 3. Qualidades da documentação. 4. Software. I. Omaia, Derzu. II. Título.

UFPB/CI CDU 004.415.2.01

Elaborado por ROGERIO FERREIRA MARQUES - CRB-15/690

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos **22** dias do mês de **Dezembro** de **2022**, às **14:00** horas, em sessão pública, na presença da banca examinadora presidida pelo professor orientador **Derzu Omaia** e pela professora **Danielle Rousy Dias Ricarte** e **Luiz Fernando Fonsêca Pinheiro de Lima**, o aluno **Kelvin Williams Neves Bernardo** apresentou o trabalho de conclusão de curso intitulado: **Documentação de Software: Exploração dos problemas mais frequentes e qualidades mais relevantes**, como requisito curricular indispensável para a integralização do Curso de **Engenharia da Computação**.

Após a exposição oral, o candidato foi arguido pelos componentes da banca que reuniram-se reservadamente, e decidiram, **APROVAR** a monografia, com nota **8,5.** Divulgando o resultado formalmente ao aluno e demais presentes, eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

João Pessoa, 22 de Dezembro de 2022

Derzu Omaia

Derzu Omaia

Danielle Rousy Dias Ricarte

Danielle Rousy Dias Ricarte

Luiz Fernando F. P. de Lima

Luiz Fernando Fonsêca Pinheiro de Lima

Kelvin Williams Neves Bernardo

Kelvin Williams Neves Bernardo

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que fizeram parte da minha vida e me ajudaram chegar onde estou hoje, meu pais Edilson e Adailza, que me criaram da melhor maneira que puderam e me deram amor e apoio incondicional, meus irmãos, Karol, Kaline e Klinton, que sempre me fizeram companhia, a mulher da minha vida e minha namorada, Laís, você é a responsável por grande parte da minha motivação de lutar para ser melhor todo dia além de ser a mulher mais incrível desse mundo, aos meus amigos, Fleig, Itallo, Caio, Rafael(Duarte), Luiz, Rafael(Mala) e Pedro, Vinícius, Leo e David, tudo fica mais fácil quando você tem alguém pra zoar junto, meus gatos, Zoro, Kanna, Gyro, Johnny e Mai e ao meu falecido cachorro Rony, você estará pra sempre dentro do meu coração meu guerreirinho.

Também agradeço à instituição UFPB e ao CI pela oportunidade de cursar Engenharia da Computação e agora estar finalizando esta etapa da minha vida, agradeço aos professores Derzu e Raoni, pelo apoio e compreensão imensos durante todas as tentativas de desenvolvimento deste trabalho.

RESUMO

O objetivo deste trabalho foi realizar uma pesquisa exploratória sobre o tema de documentação de *software*, buscar artigos científicos que ajudem a conhecer o tema e determinar quais são os problemas mais frenquentemente enfrentados por desenvolvedores durante as atividades de Documentação. A motivação para realização deste trabalho foi determinar quais problemas devem ser focados para tornar o desenvolvimento da Documentação menos trabalhoso, bem como determinar quais são os atributos de qualidade mais importantes de um documento de *software*. Foi feita uma pesquisa bibliográfica para eleger artigos relevantes sobre o tema e em seguida comparar e compilar os resultados dos artigos mais relevantes. Foram determinados os problemas mais enfrentados (incompletude, ambiguidade, incorreção, obsolescência e falta de organização), os problemas considerados mais importantes (ambiguidade, incompletude, incorreção, falta de exemplos e obsolescência), e as qualidades mais importantes (integridade, atualidade, legibilidade, usabilidade e disponibilidade). Também foram elaboradas cinco hipóteses para serem exploradas em trabalhos futuros.

Palavras-chave: Documentação de Software, Problemas de Documentação, Qualidades da Documentação.

ABSTRACT

The objective of this work was to carry out an exploratory research on the *software* documentation theme, to look for scientific articles that help getting to know the theme and to determine which are the most frequently faced problems by developers during the documentation activities. The motivation for carrying out this work was to determine which problems should be focused to make documentation development less laborious, as well as to determine which are the most important quality attributes of a software document. A bibliographic search was carried out to select relevant articles on the subject and then compare and compile the results of the most relevant articles. The most faced problems (incompleteness, ambiguity, incorrectness, obsolescence and lack of organization), the problems considered most important (ambiguity, incompleteness, incorrectness, lack of examples and obsolescence), and the most important qualities (integrity, up-to-dateness, legibility, usability and availability). Five hypotheses were also elaborated to be explored in future works.

Keywords: Software Documentation, Documentation Issues, Documentation Qualities.

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

1 INTRODUÇÃO

"Quando falamos de engenharia de software, não se trata apenas do programa em si, mas de toda a documentação associada e dados de configurações necessários para fazer esse programa operar corretamente. Um sistema de software desenvolvido profissionalmente é, com frequência, mais do que apenas um programa; ele normalmente consiste em uma série de programas separados e arquivos de configuração que são usados para configurar esses programas. Isso pode incluir documentação do sistema, que descreve a sua estrutura;"[11], como citado por Sommerville, desde o início dos estudos e teorizações sobre desenvolvimento de *software*, a documentação de *software* é uma das partes deste processo. Além do *software* em si, se faz necessário o desenvolvimento de documentos auxiliares, para guiar o desenvolvedor na hora de realizar manutenções ou adições no seu *software*.

A documentação de *software* voltada para o desenvolvedor se chama documentação técnica e o seu intuito é descrever e explicar o funcionamento interno do *software*, as interações entre os seus componentes, a lógica por trás de suas funções, casos de uso e etc. Apesar da documentação existir desde a criação do *software*, quanto mais tempo se passa, mais ela está sendo deixada de lado pelos desenvolvedores e pesquisadores, por motivos de falta de tempo para desenvolvê-la ou falta de visão da utilidade que esta documentação pode vir a ter [3].

Esta pesquisa trabalha com a hipótese de que o motivo da falta de interesse dos praticantes (desenvolvedores, engenheiros e pesquisadores) da área de desenvolvimento de software se dá pelo motivo de que desenvolver a documentação exige muito trabalho e nem sempre ela chega a ser utilizada o bastante para compensar este trabalho. Parte desse esforço citado se dá pelos problemas que são enfrentados durante e depois do desenvolvimento da documentação, problemas que causam retrabalho ou atrasos neste desenvolvimento.

O intuito deste trabalho é explorar o estado da arte das pesquisas sobre documentação de *software*, a procura de quais são os problemas mais pertinentes que ocorrem durante seu desenvolvimento, explorar também quais os benefícios e utilidades da Documentação, bem como quais qualidades deve-se almejar ao desenvolvê-la para que ela possa ter o seu valor utilitário elevado.

1.1 Definição do Problema

O problema a tratar neste trabalho é que a área de documentação de *software* é muito pouco explorada no meio científico, principalmente no Brasil. Desta forma o intuito deste trabalho é realizar tal exploração, para que novas pesquisas e trabalhos possam ser

desenvolvidos dentro deste tema.

1.2 Premissas e Hipóteses

A premissa que este trabalho trata é de que a área de documentação de *software* é pouco explorada no meio acadêmico, principalmente no Brasil, este trabalho é realizado no intuito de averiguar o estado da arte das pesquisas feitas nesta área e determinar quais são os problemas mais pertinentes desta área.

A hipótese abordada neste trabalho é que muitos dos problemas enfrentados por desenvolvedores ao produzir e utilizar a documentação de um *software*, além de serem recorrentes e comuns, são conhecidos no meio científico, mas soluções para os mesmos são pouco exploradas.

1.3 Objetivo geral

Realizar uma pesquisa bibliográfica exploratória da área de documentação de *software*, buscando determinar quais são os problemas mais frequentemente enfrentados por praticantes no desenvolvimento de atividades relacionadas ao tema.

1.4 Objetivos específicos

- Analisar o estado da arte das pesquisas e estudos sobre documentação de *software*;
- Conhecer e listar quais os problemas mais relevantes enfrentados por praticantes em atividades de documentação de *software*;
- Conhecer e listar quais as qualidades mais valorizadas por praticantes em documentos de *software*;
- Realizar uma comparação e compilação dos dados e conhecimentos obtidos sobre o tema;

1.5 Estrutura da monografia

Este trabalho está dividido em: Referencial Teórico, Metodologia, Resultados e Conclusões e Trabalhos Futuros.

No **Referencial Teórico**, é conceituado o tema documentação de *software*, são explorados artigos e estudos sobre o tema. Também são explorados os problemas mais enfrentados por desenvolvedores durante o desenvolvimento da documentação e os atributos mais valorizados em um documento de *software*.

Na **Metodologia** é explicada qual foi a metodologia empregada para realizar a pesquisa bibliográfica sobre o tema, fazer a comparação dos resultados das pesquisas escolhidas para se basear e também fazer a junção destes resultados, a fim de obter uma visão mais geral sobre os temas explorados.

Nos **Resultados** são mostrados quais os resultados foram encontrados depois que a pesquisa foi completada, são criadas hipóteses sobre os resultados encontrados.

Nas **Conclusões e Trabalhos Futuros** é discorrido sobre o que foi aprendido e descoberto durante o desenvolvimento do trabalho, quais resultados foram obtidos e quais hipóteses foram criadas e também quais trabalhos podem ser desenvolvidos a partir dos conhecimentos obtidos neste.

2 REFERENCIAL TEÓRICO

2.1 Documentação de Software

Desde o início dos estudos sobre desenvolvimento de software, foi-se constatado que além do software em si, fazia-se também necessária a criação de um documento o qual pudesse ajudar o usuário do software a conhecer e fazer uso do mesmo, bem como seria necessário um documento descrevendo a estrutura e funcionamento interno deste software, o segundo sendo destinado à pessoa ou equipe responsável pela manutenção dele [1]. Lomax[1] descreve a documentação como "... a ponte entre fornecedores e usuários de produtos de software".

A documentação de um software é, de maneira simples, um ou mais documentos ou diagramas que devem relatar de forma clara e explicativa o funcionamento e/ou a estrutura do Software ao qual se refere e, portanto, "tem um *status* ou autoridade oficial e pode ser usado como evidência"[2], "Espera-se que um documento seja correto ou preciso, ou seja, espera-se que as informações que alguém possa obter do documento sejam realmente verdadeiras para o sistema que está sendo descrito. Quando este não for o caso, há um erro: ou o sistema está com defeito, o documento está incorreto ou ambos estão errados"[2].

Retornando ao conceito de Documentação de Software, Zhi et al.[4], após mencionar que documentação pode possuir diversos significados, faz uso de uma lista de conceitos para definir a mesma:

- Documentação é uma descrição escrita de sistemas de software;
- Espera-se que a documentação forneça informações precisas sobre os sistemas;
- A documentação pode se referir ao manual do produto que os desenvolvedores criaram para usuários;
- A documentação de desenvolvimento (ou técnica) é criada para fins de comunicação entre engenheiros de software;
- A documentação pode se referir a diferentes artefatos, incluindo requisitos, *design*, comentário de código, casos de teste, etc;
- A documentação pode ser apresentada em diferentes formatos, variando desde o tradicional texto escrito até modelos gráficos (ex., aqueles que usam UML), de texto estático a sistemas dinâmicos de hipertexto.

Tendo em vista esta variedade de significados, é de se esperar que diversas formas de descrição, conjuntos de instruções, representações e etc. sejam considerados tipos

válidos de documentação de um código. No estudo feito por Aghajani [6], foram entrevistados 146 praticantes (pesquisadores e desenvolvedores de software) com o intuito de determinar "(i) quais os problemas de documentação que eles enfrentam com mais frequência e quais soluções são mais comumente aplicadas aos mesmos, e (ii) quais os tipos de documentação são considerados mais importantes para cada tarefa relacionada à documentação", são definidos pelos autores, ao todo, 13 tipos diferentes de documentação de software, sendo estes: Comentário de código, Guia de Contribuição, Conhecimentos da Comunidade, Manual de Usuário, Guia de Implantação, Perguntas Frequentes, Guia de Migração, Tutorial, Notas de Lançamento/*Change Log*, Guia de Instalação, Tutoriais em Vídeo, Referência da API e *Getting Started*.

Em resumo, a Documentação de Software pode ser definida como: todo e qualquer texto, documento, imagem ou combinação dos três que possa de alguma maneira auxiliar o leitor a melhor entender o funcionamento, estrutura e/ou uso de um determinado software.

2.1.1 Documentação Técnica e Não-Técnica

Mesmo possuindo tal abrangência de significados, uma divisão se faz clara nas fontes referenciadas sobre o tema [3][4][5][7], a divisão entre os grupos documentação não-técnica e documentação técnica. A não-técnica, como o nome indica, é aquela destinada ao usuário final do software, podendo ser, por exemplo, um manual ou uma página de "ajuda", ela guia o usuário e tira dúvidas sobre como fazer uso do software à qual se refere, sem necessariamente entrar em detalhes técnicos de funcionamento. Já a documentação técnica é destinada aos desenvolvedores do software, deve servir como fonte de informação tanto durante o desenvolvimento quanto após o lançamento do software, para que seja feito o desenvolvimento, a manutenção e o incremento do software da melhor maneira possível. Ela deve descrever com precisão e clareza a estrutura do software, detalhando o funcionamento de todos (ou dos mais importantes, caso o software seja muito complexo) os componentes internos deste, como eles realizam suas tarefas e como se comunicam.

Segundo Aghajani [5]"Documentação técnica (ex.: Referência de API, guias) descreve informação sobre o design, código, interfaces e funcionalidade do software para auxiliar desenvolvedores nas suas tarefas". Já Coelho [7], define a documentação técnica como "Voltada ao desenvolvedor, compreende dicionários e modelos de dados, fluxogramas de processos e regras de negócios, dicionários de funções e comentários de códigos". Documentação técnica engloba tudo aquilo que é desenvolvido com o intuito de ser utilizado pelo desenvolvedor, a pessoa que tem acesso ao código fonte do software ou partes dele e que irá fazer uso da mesma para obter informações necessárias para realizar a manutenção, incremento ou até mesmo para aprender como aquele software funciona.

Tendo ciência da definição geral de documentação de *software*, e da separação entre

técnica e não-técnica, o próximo tema a ser abordado são os usos da documentação técnica nas diferentes atividades de desenvolvimento de *software*.

2.1.2 Utilidade e Qualidade da Documentação de Software

A documentação é desenvolvida e utilizada em diversas diferentes atividades que ocorrem antes, durante e depois do desenvolvimento de um software, como por exemplo: diagramas UML (*Unified Modelling Language*) representando a estrutura do sistema podem ser criados durante o processo de criação do projeto do software, no entanto podem ser utilizados tanto durante o desenvolvimento do mesmo, para entender e desenvolver componentes e funções que se comportem da maneira desejada, bem como durante tarefas de manutenção, para facilitar a compreensão do comportamento desejado do software. Outro forte exemplo é o comentário de código, desenvolvido durante a implementação do software, é utilizado durante tarefas de manutenção(corretiva e aditiva), bem como pode ajudar no desenvolvimento e execução de testes unitários, funcionais e de regressão [3].

Garousi et al. [3], em 2015, diz o seguinte sobre a utilidade da documentação: "Geralmente, engenheiros de software se apoiam na documentação técnica como uma ajuda no entendimento do sistema e na compreensão do programa. Sem documentação, a única fonte confiável de informação é o código-fonte. Por consequência, muitas vezes se toma muito tempo e esforço explorando o código-fonte para encontrar a informação relevante e obter um entendimento da funcionalidade do sistema". Vale ressaltar que ao citar a leitura do código-fonte como a única fonte de conhecimento além da documentação, não está sendo falado sobre a leitura de comentários de código, pois estes também são considerados documentação, como citado anteriormente.

Em sua pesquisa, Aghajani [6] classifica quais os tipos de documentação são vistos como mais úteis no contexto de cada atividade dentro da Engenharia de Software:

- **Comentários de Código**: *Debugging*, Compreensão do programa.
- **Diretrizes de Contribuição**: Testes de Software/*Quality Assurance*(QA), Programação de Software, Design da Estrutura e da Arquitetura do Software.
- **Conhecimento da Comunidade**: Engenharia de Requisitos, Design da Interface do Usuário, *Debugging*, QA.
- **Guia de *Deployment***: Gerenciamento de *Release*.
- **Perguntas Frequentes**: Programação do Software e Aprendizado de um novo *framework*/tecnologia.
- **Guia de Migração**: Gerenciamento de *Release*.

- **Notas de lançamento(*Change Log*)**: Gerenciamento de *Release*.
- **Tutorial/Tutorial em Vídeo**: QA.
- **Guia de Instalação**: QA, Migração de Software/Dados, Gerenciamento de *Release*.
- **Referência de API**: Programação do Software.
- **Manual do Usuário e *Getting Started***: Não foram nomeadas como muito relevantes em nenhuma atividade.

Ainda na pesquisa de Garousi [3], também é possível perceber que os dois tipos de documentação considerados mais utilizados pelos desenvolvedores entrevistados são o Comentário de Código e as Diretrizes de Contribuição, ambos mais presentes nas atividades relacionadas ao desenvolvimento.

A percepção de utilidade de uma documentação está fortemente relacionada com a qualidade desta, como descreve Forward [8]: "A utilidade é uma medida de quão bem o conteúdo de um documento pode prover conhecimento, informação e/ou *insight* para o seu leitor com respeito a outros artefatos disponíveis considerando ambos a atividade e o valor do artefato", por haver o fator humano da capacidade de interpretação de informação e nível de experiência de quem está desenvolvendo a documentação, e também a sua habilidade de repassar e descrever esta informação da melhor maneira, a qualidade do documento não depende só do método adotado para documentar ou da experiência técnica do documentador.

Mesmo tendo ciência da importância da documentação, tanto Lomax [1], quanto Parnas [2] e consequentemente Garousi et al. [3], mencionam a dificuldade que desenvolvedores de software têm em decidir qual nível de importância deve-se dar para a atividade de documentação do software, dando-se como justificativa para essa dúvida a falta de estudos mais concretos sobre o quanto é seguro investir tempo e recursos na atividade de documentação, por esse motivo são usados argumentos como "O código-fonte já pode ser usado como documentação"ou "Não vale a pena alocar tempo e recursos demais para desenvolvimento da documentação, pois ela nunca será usada com frequência o bastante para justificar este esforço"[3]. Porém, assim como Parnas afirma: "...na prática, a maioria dos programas são tão complexos que considerá-los como sua própria documentação é uma ilusão ingênua ou falsa"[2], o estudo feito por Garousi et al.[3] mostra que, apesar de que para realizar tarefas de manutenção o código fonte seja o mais considerado, para atividades de desenvolvimento (planejamento de melhorias e adição de novas funcionalidades) a documentação técnica é consultada com maior frequência como fonte de informação sobre o software.

Mas afinal, como classificar uma documentação como tendo boa ou má qualidade?

A pesquisa feita por Zhi et al.[4], por meio de um mapeamento sistemático de artigos abordando o tema de Documentação de Software, classifica, por meio de ordem de evidência, quais atributos de qualidade da Documentação de Software recebem o maior foco dentro dos artigos consultados. Como pode ser visto na Figura 1, os 5 atributos mais relevantes são, respectivamente, Integridade (*Completeness*), Consistência, Acessibilidade, Formato e Atualidade (*Up-to-dateness*).

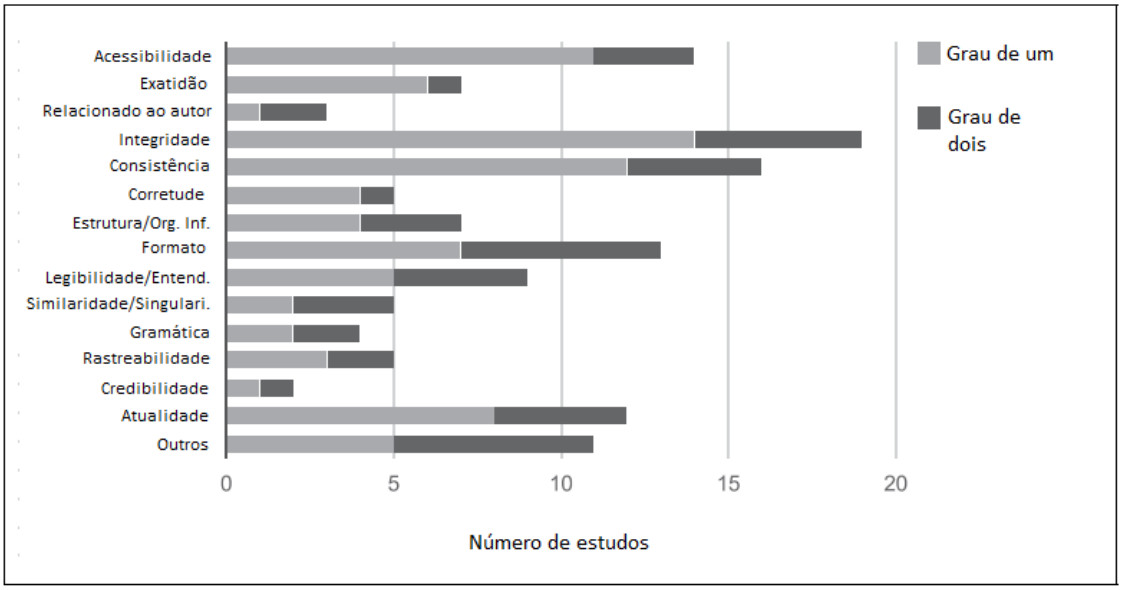

Figura 1: Atributos de Qualidade de Documentação, traduzido e adaptado de Zhi et al. [4]

Sobre estes resultados, Zhi fez a seguinte interpretação relacionada ao atributo "Integridade" ter sido classificado como mais relevante: "... os atributos de documentação que atraem mais atenção de pesquisadores anteriores é "**Integridade**" (17 artigos, 28%). Isto pode indicar que documentação incompleta tem sido uma das dificuldades que a maioria dos pesquisadores ou profissionais tentaram abordar" [4]. Também é possível destacar que na pesquisa feita por Aghajani et al.[6] os entrevistados classificaram como mais importantes em uma Documentação os atributos "**Legibilidade (Clareza)**", "Atualidade" e "Integridade".

Ding et al.[9] também realizaram uma pesquisa bibliográfica sobre o tema e constataram que, dos atibutos de qualidade avaliados, "**Compreensibilidade**" foi o mais abordado nos artigos analisados, seguido de "Rastreabilidade" e "Consistência".

Analisando os atributos apresentados nas três pesquisas citadas, podemos ver que "Integridade", "Consistência" e "Atualidade" foram os mais citados, demonstrando assim, que o que os entrevistados mais valorizam em uma documentação é que a informação contida nele seja completa, inequívoca e atual. Assim como cita Parnas [2]: "Espera-se que os documentos sejam precisos e inequívocos. Em outras palavras, não deve haver dúvida sobre o que eles significam", e ainda "Deve-se notar que não se espera que os documentos

individuais sejam completos no sentido de fornecer todas as informações que possam ser fornecidas sobre o sistema". O importante é que a informação sobre o funcionamento do sistema seja passada de forma clara e objetiva, não é necessário descrever toda e qualquer possibilidade de funcionamento para que a documentação seja considerada completa.

Classificar quais atributos de qualidade são considerados mais importantes na documentação é uma tarefa muito importante para determinar onde se deve manter o maior foco durante e após seu desenvolvimento, para que a documentação mantenha seu valor informativo ao mesmo tempo que o software é desenvolvido e incrementado.

2.2 Problemas mais frequentemente enfrentados

Retornando brevemente à hipótese de Zhi et al. [4] citada anteriormente, a qual menciona que a falta de integridade pode ser um problema extensamente explorado, a seguir serão abordados os problemas mais enfrentados por desenvolvedores relacionados à Documentação de Software.

Aghajani et al. publicaram dois estudos nos anos de 2019[5] e 2020[6], o primeiro sendo um estudo exploratório, buscando em fontes diversas disponíveis na internet (Perguntas feitas no Stack Overflow, *Issues* em projetos *open-source* no Github, Listas de email e etc.) *reports* ou pedidos de ajuda para solucionar problemas relacionados a Documentação de Software e a partir desta compilação de 824 artefatos, conseguiu classificá-los em 162 diferentes tipos de problemas relacionados à documentação, estes tendo sido agrupados em quatro categorias principais: (i)**Conteúdo da informação (O quê)**, (ii) **Conteúdo da informação (Como)** , (iii) **Relacionados a Processos** e (iv) **Relacionados a ferramentas**. Dentro de cada uma destas categorias também foi feita a divisão em subcategorias: (i) Corretude (*Correctness*), Integridade e Atualidade, (ii) Manutenibilidade, Legibilidade, Usabilidade e Utilidade, (iii) Internacionalização, Contribuição na documentação, Configuração de gerador de documento, Problemas de Desenvolvimento Causados pela Documentação e Rastreabilidade, (iv) *Bug/Issue*, Suporte/Expectativas, Ajuda Requerida e Migração de Ferramenta. No Anexo A está detalha a taxonomia completa de problemas criada por Aghajani et al., retirada do artigo.

Utilizando as informações obtidas no estudo publicado em 2019, Aghajani realizou uma segunda pesquisa [6], publicada em 2020, entrevistando praticantes (Desenvolvedores e Pesquisadores de Sofware), para definir, a partir da taxonomia obtida anteriormente, quais os problemas de documentação que eles enfrentam com mais frequência e quais soluções são mais comumente aplicadas aos mesmos, os resultados completos desta pesquisa estão localizados no Anexo B.

Dos resultados apresentados por Aghajani et al.[6], podemos destacar os problemas enfrentados com mais frequência e também considerados mais importantes pelos entrevis-

tados na Tabela 1. Todos os problemas citados na Tabela 1 se localizavam entre os 25% mais frequentemente enfrentados pelos entrevistados.

Tabela 1: Problemas enfrentados com maior frequência e considerados mais importantes, taduzido pelo autor

Problema	% de importância
Clareza	88
Falta Documentação para uma nova funcionalidade/componente	69
Documentação do usuário ausente	65
Acessibilidade	65
Falta de tempo para escrever a documentação	65
Instruções de instalação inapropriadas	63
Inconsistência Código-Documentação	59
Conteúdo não é útil na prática	59
Organização da Informação	49
Reportando problemas encontrados na documentação	44
Organização dos arquivos da documentação	40
Rastreabilidade	35
Formato/Apresentação	33
Pobre/Falta de Automatização	29

Outra pesquisa que lida com este tema é a feita por Uddin e Robillard [10], também entrevistando Desenvolvedores, realizaram pesquisas dentro do tema Documentação de APIs e reuníram as informações demonstradas na Figura 2.

Na Figura 2, lendo o campo "E"como sendo o número de exemplos dados pelos entrevistados de documentações problemáticas e o campo "D"sendo o número de Desenvolvedores que reportaram o problema, podemos perceber que "Incompletude"é o problema mais reportado, seguido por "Ambiguidade"e "Inchaço"(As descrições eram excessivamente extensas, verbosas).

Ainda na pesquisa feita por Uddin e Robillard[10], foi feito outro questionário com diferentes entrevistados, que rendeu os resultados mostrados na Figura 3, nela o eixo horizontal representa a frequência em que os problemas são encontrados pelos entrevistados, o eixo vertical representa a severidade e a necessidade de resolvê-lo e o tamanho do círculo indica quantos entrevistados deram prioridade máxima para tal problema.

Como pode-se averiguar, "Incompletude"novamente foi tido como problema mais

Categoria	Problema	Descrição	E*	D*
Conteúdo	Incompletude	A descrição de um elemento da API ou tópico não estava onde é esperado que esteja.	20	20
	Ambiguidade	A descrição de um elemento da API estava na maior parte completa mas pouco clara.	16	15
	Exemplos não explicados	Um exemplo de código foi explicado insuficientemente.	10	8
	Obsolescência	A documentação de um tópico se referia a uma versão prévia da API	6	6
	Inconsistência	A documentação de elementos que deveriam ser combinados não correspondeu.	5	4
	Incorreção	Alguma informação estava incorreta.	4	4
		Total	**61**	**57**
Apresentação	Inchasso	A descrição de um elemento da API ou tópico era verbosa ou excessivamente extensa.	12	11
	Fragmentação	A informação relacionada a um elemento ou tópico foi fragmentada ou espalhada em muitas páginas ou seções.	5	5
	Informação estrutural em excesso	A descrição de um elemento continha informação redundante sobre a sintaxe ou estrutura do elemento, que poderia ser facilmente obtida através de IDEs modernas.	4	3
	Informação embaralhada	A descrição de um elemento da API ou tópico foi misturada com informação que o correspondente não necessitava.	4	3
		Total	**25**	**22**

** E é o número de exemplos que mencionavam um problema; D é o número de desenvolvedores que reportaram um problema.*

Figura 2: Problemas de documentação de API reportados no questionário exploratório, traduzido pelo autor [10]

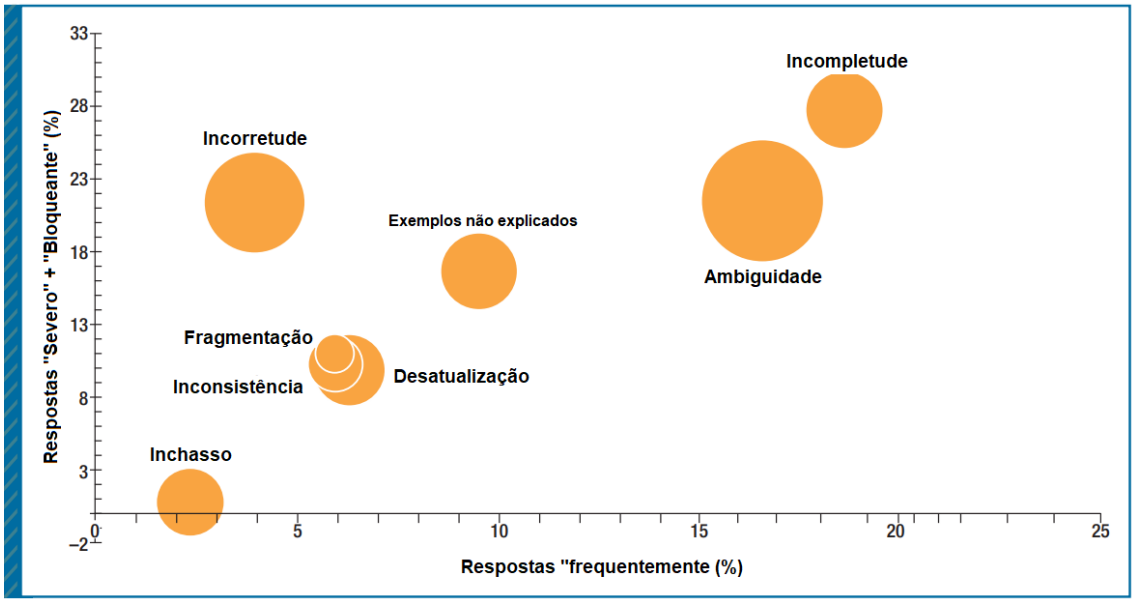

Figura 3: Frequência, severidade e necessidade de resolução de problemas em Documentação de APIs, traduzida pelo autor [10]

enfrentado e de maior severidade, no entanto "Ambiguidade"é o problema considerado como mais prioritário pela maioria dos entrevistados, "Incorreção", apesar de não ser enfrentado com muita frequência, também foi tido como severo e sua resolução mais

prioritária.

3 METODOLOGIA

A metodologia empregada neste trabalho foi a de uma revisão integrativa simplificada, baseada nas instruções de Souza et al. [12], com o intuito de pesquisar, analisar e entender sobre o tema Documentação de Software, focando principalmente nos problemas enfrentados por desenvolvedores e demais praticantes da área, com o intuito de compilar os resultados de pesquisas e eleger quais são os problemas mais relevantes dentro da área de Documentação de Software, bem como quais são os atributos mais valorizados em um Documento.

A metodologia empregada dispõe de 6 fases, listadas a seguir:

- **1ª Fase**: elaboração da pergunta norteadora;
- **2ª Fase**: busca ou amostragem na literatura;
- **3ª Fase**: coleta de dados;
- **4ª Fase**: análise crítica dos estudos incluídos;
- **5ª Fase**: discussão dos resultados;
- **6ª Fase**: apresentação da revisão integrativa.

3.1 Elaboração da pergunta norteadora

A pergunta norteadora foi definida como: "Quais são os problemas enfrentados com maior frequência por desenvolvedores durante a realização de atividades relacionadas à documentação de software? E também quais são os atributos de qualidade mais importantes em um documento de software?"

3.2 Busca na literatura

Foram utilizados os sites Google Scholar[1] e IEEE Xplore [2] para pesquisar por Artigos, Livros, Dissertações e Teses que abordassem o tema de Documentação de Software, utilizando as palavras-chave "Documentação de Software", "Documentação de Código", "Documentação", "Documentação Engenharia de Software"e todas as suas traduções para o inglês.

Após esta pesquisa foram reunidos 11 artefatos, dentre eles artigos, livros e trabalhos, que possuíam algum foco teórico no tema Documentação de Software, em uma

[1]Acessível em: https://scholar.google.com.br/
[2]Acessível em: https://ieeexplore.ieee.org/

tentativa de aumentar a quantidade de artefatos obtidos, foi observada a lista de referência destes 11 artefatos, e a partir dela foram encontrados mais 3 artigos que não haviam sido encontrados anteriormente, totalizando 14 artefatos.

3.3 Coleta de dados

Em seguida foi feita a leitura de cada um dos artefatos, no caso de livros, da sessão que aborda o tema em questão. Durante esta leitura foi averiguado quais dos artefatos constatavam pesquisas que focavam nos problemas enfrentados por desenvolvedores dentro do tema Documentação de Software, bem como quais são as qualidades consideradas mais importantes por desenvolvedores em uma Documentação de Software.

3.4 Comparação dos resultados encontrados

Após leitura e interpretação dos artefatos, foram feitas comparações dos resultados de pesquisas similares dentro do tema. Foram encontrados três diferentes tipos de resultados dentre os artefatos os quais poderia ser feita uma comparação:

1. **Problemas listados por frequência de ocorrência**

   Em artefatos onde praticantes foram questionados e, em alguma parte da entrevista ou questionário foi pedido para que fossem nomeados ou votados, a partir de uma lista pré-fabricada, quais problemas eram enfrentados com maior frequência, ao desenvolver atividades de Documentação de Software.

2. **Problemas listados por ordem de importância**

   Assim como no anterior, em artefatos nos quais os entrevistados foram estimulados a escolher, dentre uma lista de problemas enfrentados em atividades de Documentação de Software, quais seriam os de maior importância, ou seja, que deviam ser investigados e resolvidos com maior prioridade.

3. **Atributos de qualidade em ordem de importância**

   Neste caso, foram averiguados artefatos onde, também por meio de entrevistas ou questionários, os participantes foram levados a nomear quais seriam as qualidades mais importantes em uma Documentação de Software.

Como esperado, cada artefato analisado nomeou um lista própria de Atributos e/ou Problemas de Documentação para usar na sua pesquisa, e por isso, haviam muitos elementos presentes em um artefato mas não em outros, mas também havia um número

significante de elementos iguais ou semelhantes, e por isso, com o intuito de aumentar o leque de comparações em cada um dos casos, elementos que são sinônimos ou muito similares foram considerados como sendo o mesmo, com o intuito de aumentar o número de categorias comparadas.

Exemplo: No artigo **A**, foram usados os atributos: Legibilidade, Atualidade e **Usabilidade**. E no artigo **B**, foram usados os atributos: Legibilidade, Atualidade e **Navegação**. Para que a comparação pudesse ser mais completa "Navegação"e "Usabilidade"foram considerados como a mesma categoria.

A lista completa das equivalências utilizadas está descrita a seguir:

- "Uso de modelos visuais"e "uso de diagramas";
- "Falta de relevância"e "conteúdo supérfluo";
- "Organização da informação"e "Organização estrutural";
- "Compreensibilidade", "legibilidade"e "entendimento".

Para cada um dos tipos de resultados citados anteriormente, foi feita uma comparação entre todas as categorias equivalentes, utilizando os valores percentuais de cada resultado, para que a diferença no tamanho da amostra de cada pesquisa não interefe-risse. O intuito desta comparação é obter o maior número de amostras dos resultados escolhidos, para então analisar e criar hipóteses sobre estes resultados.

3.5 Junção dos resultados

Em seguida, em cada um dos casos em que fosse aplicável, foi feita uma junção de todos os resultados em um só para cada categoria, esta junção se deu da seguinte maneira:

Primeiramente, para poder acumular os resultados, foi necessário obter o valor de "votos"únicos em cada uma das categorias, ou seja, exatamente quantas pessoas nomearam cada um dos Atributos ou Problemas. Para obter tais valores, foi necessário multiplicar o valor percentual de cada categoria pelo número total de entrevistados do artefato, obtendo assim o número de pessoas que nomearam tal categoria.

Exemplo: No artigo **A**, foram entrevistadas 50 pessoas e estes foram os resultados dos atributos mais importantes na Documentação: "Legibilidade"com 10%, "Atualidade"tendo 20% e "Usabilidade"com 30%. Para obter o número de pessoas que escolheram "Atualidade"como mais importante, precisamos calcular:

$$Resultado_{Atualidade} * Total_{Entrevistados} = 10\% * 50 = 0,10 * 50 = 5$$

Tendo os valores em número de voto individuais em cada Qualidade ou Problema, é possível somar os resultados similares em cada uma delas. Para assim, após analisar esta perspectiva, formular novamente hipóteses e teorias sobre estes resultados.

4 RESULTADOS

Neste capítulo, como mencionado na metodologia, serão mostrados os resultados da pesquisa, Comparação e Junção de resultados obtidos.

4.1 Problemas mais enfrentados por entrevistados

Neste quesito, os resultados de dois artigos foram utilizados, a pesquisa feita por Aghajani et al.[6] e a pesquisa feita por Uddin et al.[10]. Em ambos os artigos, os entrevistados(em ambos praticantes da área) foram estimulados a escolher, dentre os problemas listados, quais aconteciam com mais frequência durante suas atividades de trabalho, os resultados comparados estão demonstrados na tabela 2.

Tabela 2: Problemas enfrentados com mais frequência[6][10]

	Aghajani	**Uddin**
Ambiguidade	52,56%	23,19%
Incompletude	70,51%	28,99%
Exemplos	7,69%	14,49%
Obsolescência	32,05%	8,70%
Inconsistência	16,67%	7,25%
Incorreção	43,59%	5,80%
Extensividade	7,69%	17,39%
Falta de organização	15,38%	13,04%
Informação Supérflua	7,69%	5,80%

Como pode-se ver na Tabela 2, apesar da diferença nas grandezas entre as duas pesquisas, é possível notar que "Ambiguidade"e "Incompletude"são os mais nomeados em ambos estudos. Isto pode dar a entender que, mesmo havendo discrepâncias nas experiências outros problemas enfrentados, os mais frequentes são estes dois para a maioria dos entrevistados em ambos os estudos.

A seguir foi feita a conversão dos valores das pesquisas de Aghajani[6] e Uddin[10], para que os valores pudessem ser lidos como votos individuais, e assim podendo acumular os dois resultados em um único, o resultado da conversão é ilustrado na Figura 4.

Analisando os resultados acumulados mostrados na Figura 4, podemos afirmar com mais certeza que "Incompletude"e "Ambiguidade"são os problemas mais enfrentados pelos desenvolvedores entrevistados, com "Incorreção", "Obsolescência"e "Falta de Organização"seguindo logo após. Estes resultados reforçam que os problemas enfrentados mais frequentemente dentro da Documentação de Software são problemas relacionados à qualidade da informação apresentada, e que na grande maioria das vezes a Documentação

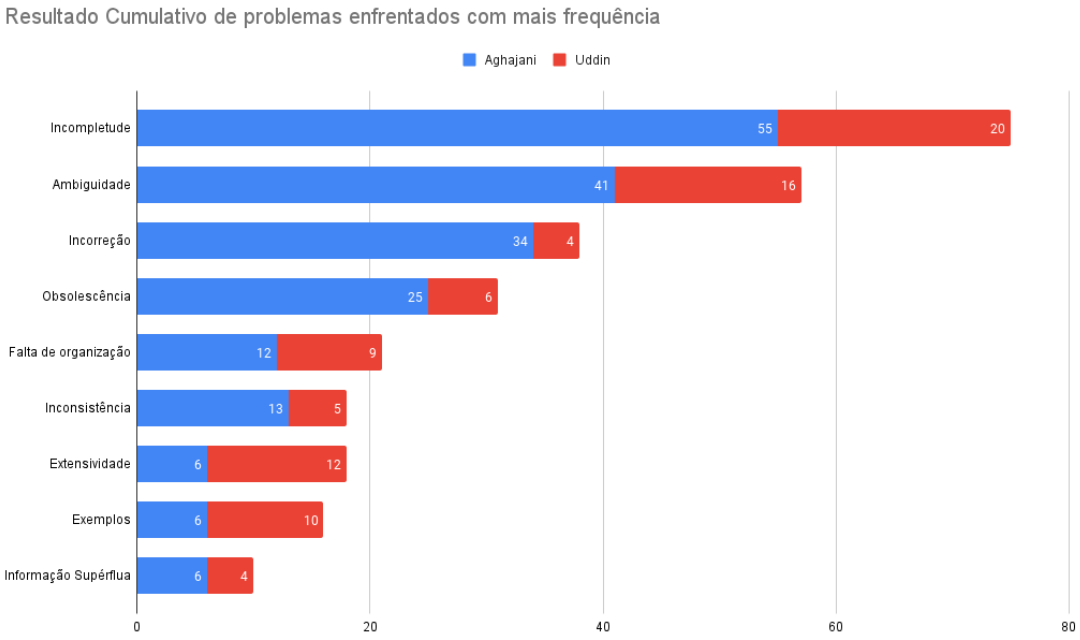

Figura 4: Problemas mais frequentemente enfrentados acumulados [6][10]

tem informações incompletas e/ou explicações ambíguas. Uma hipótese que pode ser criada a partir desta análise é de que, em média, **(i) não é dedicado tempo o bastante para o desenvolvimento da documentação, fazendo com que ela fique incompleta**. Em contrapartida, também pode-se hipotetizar que, **(ii) por mais que se trabalhe extensivamente no desenvolvimento da documentação, é muito difícil que todas as informações necessárias sejam descritas**. Mais estudos precisam ser feitos para determinar qual destas hipóteses é a mais correta.

Sobre o problema de ambiguidade, podemos hipotetizar que **(iii) a ambiguidade pode ocorrer por: o desenvolvedor da documentação não possui o conhecimento técnico geral profundo o bastante do código para conseguir explicar com precisão o funcionamento dele, não possui o nível desejado de habilidade didática para poder explicar com precisão o funcionamento do código, ou ambos.**

4.2 Problemas em ordem de importância

Nesta sessão, foram utilizadas novamente as pesquisas feitas por Aghajani[6] e Uddin[10], mas desta vez focando em outro aspecto do seus resultados, tendo em mãos a lista de problemas mais frequentemente enfrentados, ambos solicitaram que os entrevistados escolhessem quais problemas tinham maior importância, ou seja, os problemas que deveriam ter prioridade máxima para ser solucionados. Os resultados são comparados a seguir no gráfico da Figura 5.

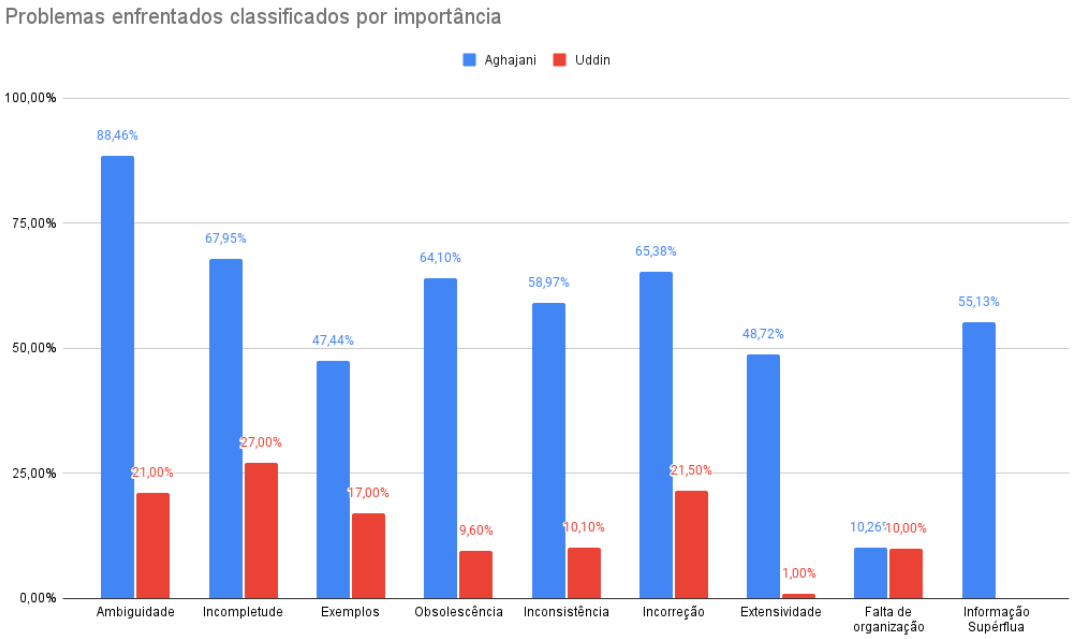

Figura 5: Problemas enfrentados classificados por importância [6][10]

Nestes resultados mostrados na Figura 5 podemos ver que, em média, os entrevistados dão maior importância aos mesmos problemas que ocorrem com mais frequência, mas na pesquisa de Aghajani[6], os problemas de "Ambiguidade"receberam mais destaque do que todos os outros, ao invés da "Incompletude"que liderava nos resultados anteriores.

Na Figura 6, são demonstrados os resultados acumulados de ambas as pesquisas no quesito importância dos problemas enfrentados.

Analisando os resultados demonstrados nas Figuras 5 e 6, vendo como Ambiguidade e Incompletude seguem sendo os problemas considerados mais importantes, podemos hipotetizar que segundo os praticantes entrevistados em ambos os estudos **(iv) Acima de qualquer outra qualidade, deve sempre se prezar pela Integridade (contrário de Incompletude) e pela Exatidão, quando lidando com Documentação de Software, pois estes são os dois atributos que mais contribuem para que o conhecimento seja absorvido da melhor maneira pelo leitor da mesma.** E **(v) mesmo que uma documentação esteja correta, consistente, atualizada e etc., se ela for incompleta e/ou ambígua, seu valor informativo será considerado baixo.**

4.3 Atributos de Qualidade em ordem de importância

Para esta comparação foram usados os seguintes artigos: Forward et al.[8], que fez uma entrevista com 48 participantes, e em uma de suas perguntas pediu para que os

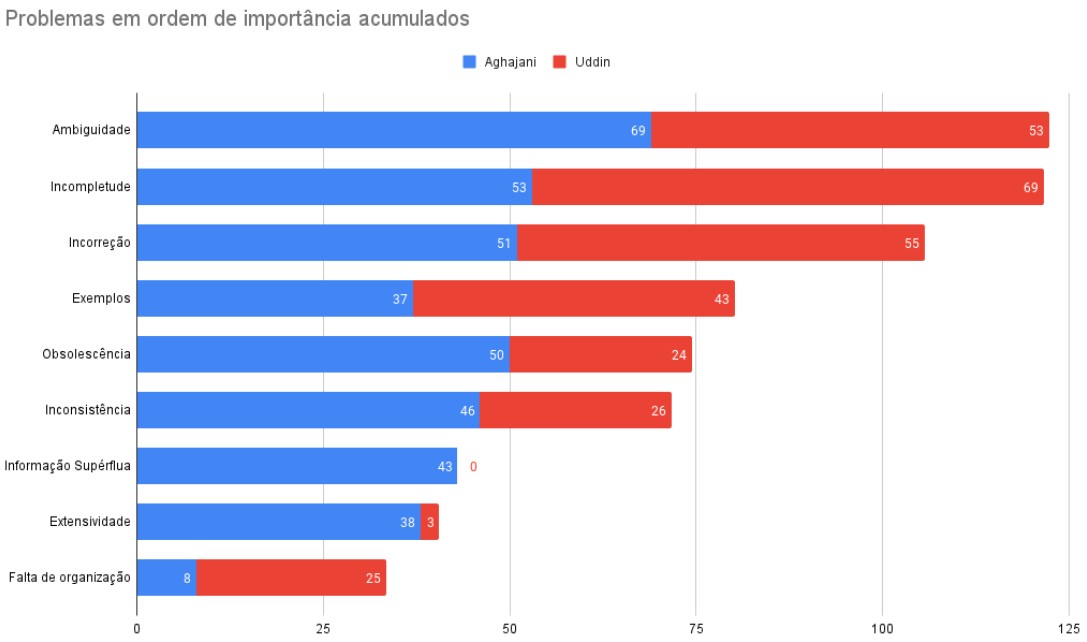

Figura 6: Problemas classificados por importância acumulados [6][10]

entrevistados avaliassem de 1 a 5, o nível de importância de Atributos de Qualidade de Documentação de Software. Para esta comparação, para cada atributo foi resgatada a porcentagem de quantos dos entrevistados deram nível de importância máximo(5); Garousi[3], que entrevistou um total de vinte e cinco participantes, e da mesma maneira, estimulou os participantes a classificar cada atributo de qualidade de documentação de *software* em um nível de importância de 1 a 4, os valores percentuais utilizados para esta comparação foram os que deram importância máxima(4) para os atributos; e novamente Aghajani[6], que questionando 78 participantes, os pediu que marcassem livremente dentro de uma lista quais atributos são mais importantes em uma Documentação de Software.

Os resultados são comparados a seguir na Tabela 3, nos campos em que não há valor, significa que no artigo em questão não foi consultado o atributo e não havia equivalência que pudesse ser feita com atributos de outros artigos. A Figura 7 demonstra um gráfico para melhor visualização destes dados.

Em seguida foi feito o cálculo de votos individuais para cada atibuto de acordo com o número de participantes de cada pesquisa, os resultados acumulados são demonstrados no gráfico da Figura 8.

A partir da interpretação dos resultados vistos na Tabela 3 e nas Figuras 7 e 8, podemos perceber que, novamente, "**Integridade(Completeness)**"é o atributo considerado mais importante pelos desenvolvedores e demais praticantes, reforçando a hipótese (iv), seguida de "Atualidade"e "Legibilidade", que também se posicionam muito bem como indicadores importantes de qualidade do Documento.

Tabela 3: Comparação de Importância de Atributos de Qualidade da Documentação [3][6][8]

	Forward	Garousi	Aghajani
Integridade	85,00%	40,00%	68,00%
Corretude	-	48,00%	65,00%
Atualidade	46,00%	52,00%	69,00%
Manutenibilidade	-	-	55,00%
Legibilidade	-	32,00%	88,00%
Usabilidade	30,00%	24,00%	65,00%
Utilidade	-	-	59,00%
Uso de diagramas	15,00%	40,00%	-
Relevância	-	48,00%	55,00%
Consistência	-	24,00%	59,00%
Tecnologia	-	4,00%	-
Exemplos	37,00%	24,00%	59,00%
Disponibilidade	41,00%	-	65,00%
Tipo	26,00%	-	-
Estrutura	11,00%	-	49,00%
Estilo de Escrita	7,00%	-	-
Tamanho	7,00%	-	-
Gramática	0,00%	-	23,00%
Rastreabilidade	7,00%	-	35,00%
Influência para usar	12,00%	-	-
Formato	0,00%	-	33,00%

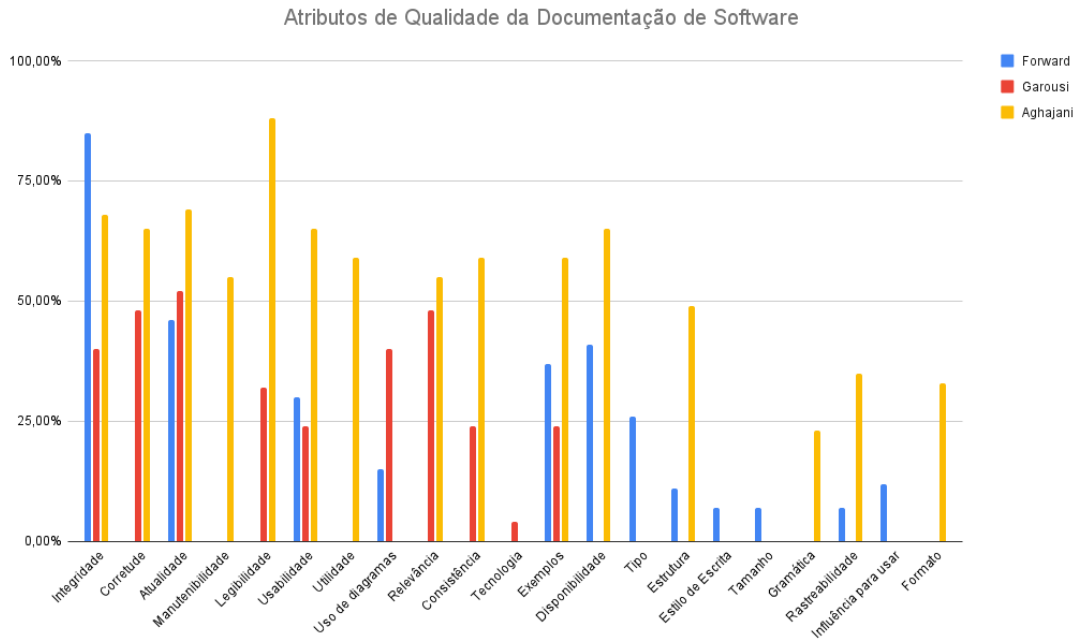

Figura 7: Atributos de Qualidade classificados por importância [3][6][8]

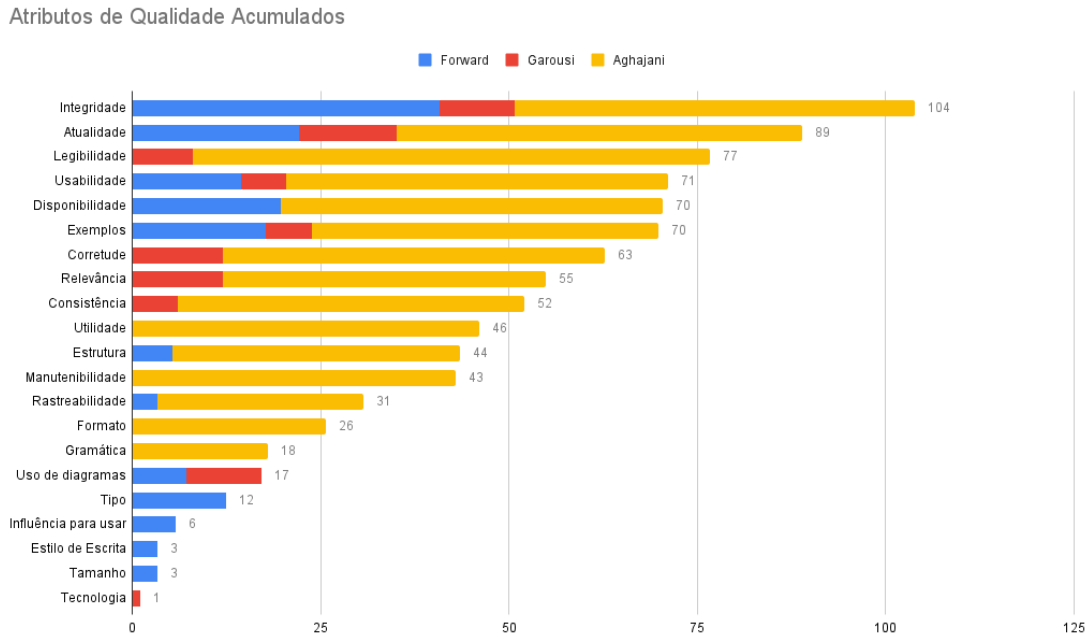

Figura 8: Atributos de Qualidade classificados por importância acumulados [3][6][8]

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho tem como objetivo realizar uma pesquisa sobre documentação de *software* e a partir desta pesquisa definir quais são os problemas mais enfrentados por praticantes da área em atividades relacionadas à mesma, bem como definir quais atributos de qualidade são considerados mais relevantes em uma documentação. Uma revisão integrativa simplificada foi realizada, os resultados dos artefatos encontrados foram analisados e comparados e, por fim, foram levantadas hipóteses e reforçados conhecimentos sobre os temas abordados. Os resultados obtidos são demonstrados a seguir.

Após finalizar esta pesquisa e a análise dos resultados obtidos, foram levantadas as seguintes hipóteses:

(i) : "Não é dedicado tempo o bastante para o desenvolvimento da documentação, fazendo com que ela fique incompleta";

(ii) : "Por mais que se trabalhe extensivamente no desenvolvimento da documentação, é muito difícil que todas as informações necessárias sejam descritas";

(iii) : "A ambiguidade pode ocorrer por: o desenvolvedor da documentação não possui o conhecimento técnico geral profundo o bastante do código para conseguir explicar com precisão o funcionamento dele, não possui o nível desejado de habilidade didática para poder explicar com precisão o funcionamento do código, ou ambos";

(iv) : "Acima de qualquer outra qualidade, deve sempre se prezar pela Integridade (contrário de Incompletude) e pela Exatidão, quando lidando com Documentação de Software, pois estes são os dois atributos que mais contribuem para que o conhecimento seja absorvido da melhor maneira pelo leitor da mesma";

(v) : "Mesmo que uma documentação esteja correta, consistente, atualizada e etc., se ela for incompleta e/ou ambígua, seu valor informativo será considerado baixo".

Para todas as 5 hipóteses criadas, é necessário que sejam feitas pesquisas mais aprofundadas para investigar sua validade.

Além da criação de hipóteses, foi possível obter conhecimentos da literatura sobre os problemas mais enfrentados por praticantes da área dentro do contexto de Documentação de Software, compilando todos os resultados em um único, foi determinado que, segundo as pesquisas consultadas, os 5 problemas mais enfrentados por praticantes foram respectivamente: (1)Incompletude, (2)Ambiguidade, (3)Incorreção, (4)Obsolescência e (5)Falta de Organização.

Também foi possível determinar quais problemas são considerados mais importantes por praticantes, segundo as pesquisas consultadas, em ordem de importãncia, os 5 melhores posicionados foram: (1)Ambiguidade, (2)Incompletude, (3)Incorreção, (4)Falta de Exemplos ou Exemplos equivocados e (5)Obsolescência.

E por fim, foram determinados quais os Atributos de Qualidade são considerados mais importantes para determinar a qualidade de um Documento, segundo os entrevistados das pesquisas consultadas, os 5 atributos de qualidade mais importantes em uma documentação são: (1)Integridade, (2)Atualidade, (3)Legibilidade, (4)Usabilidade e (5)Disponibilidade.

5.1 Trabalhos Futuros

A partir das descobertas desta pesquisa é possível realizar diversos trabalhos de continuidade, como citado anteriormente, para as cinco hipóteses criadas, é possível realizar novas pesquisas que possam investigar a validade delas, com uma pesquisa quantitativa, pode-se entrevistar praticantes da área sobre os temas aqui descorridos e aprofundar-se mais sobre as causas e efeitos dos problemas encontrados, bem como as justificativas para a escolha das qualidades como mais importantes.

Também é possível realizar pesquisas quanto aos diferentes tipos de documentação, buscando entender quais são os atributos mais desejados em diferentes contextos do desenvolvimento de software.

Outro possível trabalho futuro seria, a partir dos dados obtidos de problemas mais enfrentados, aprofundar mais ainda este conhecimento fazendo uma pesquisa local no Brasil, procurar determinar quais são as causas e melhores soluções para os principais problemas listados, e em posse destas informações elaborar um documento de requisitos para uma aplicação de documentação, que busque atender os problemas mais relevantes neste contexto.

REFERÊNCIAS

[1] LOMAX, J. D.; Tradução de Isabel Cristina Medeiros de Gouvêa **Documentação de Software**. Rio de Janeiro: Campus, 1983.

[2] PARNAS, Daniel Lorge. Precise Documentation: The Key to Better Software. **The Future of Software Engineering**, 125-148, Springer, Berlin, Heidelberg, 2011.

[3] Golara Garousi, Vahid Garousi-Yusifoğlu, Guenther Ruhe, Junji Zhi, Mahmoud Moussavi, Brian Smith. Usage and usefulness of technical software documentation: An industrial case study. **Information and Software Technology**, v. 57, p. 664-682, 2015.

[4] ZHI, Junji et al. Cost, benefits and quality of software development documentation: A systematic mapping. **Journal of Systems and Software**, v. 99, p. 175-198, 2015.

[5] AGHAJANI, Emad et al. Software documentation issues unveiled. In: **2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)**. IEEE, 2019. p 1199-1210.

[6] AGHAJANI, Emad et al. Software documentation: the practitioners' perspective. In: **2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)**. IEEE, 2020. p. 590-601.

[7] COELHO, Hilda Simone. Documentação de software: uma necessidade. **Texto Livre: linguagem e tecnologia**, v. 2, n. 1, 2009, p. 17-21.

[8] FORWARD, Andrew; LETHBRIDGE, Timothy C. The relevance of software documentation, tools and technologies: a survey. **Proceedings of the 2002 ACM symposium on Document engineering**. 2002. p. 26-33.

[9] DING, Wei et al. Knowledge-based approaches in software documentation: A systematic literature review. **Information and Software Technology**, v. 56, n. 6, p. 545-567, 2014.

[10] UDDIN, Gias; ROBILLARD, Martin P. How API documentation fails. **Ieee software**, v. 32, n. 4, p. 68-75, 2015.

[11] SOMMERVILLE, Ian. **Engenharia de software**, 9a. São Palo, SP, Brasil, p. 63, 2011.

[12] SOUZA, Marcela Tavares de; SILVA, Michelly Dias da; CARVALHO, Rachel de. **Revisão integrativa: o que é e como fazer**. Einstein (São Paulo), v. 8, p. 102-106, 2010.

ANEXO A – Taxonomia Completa de Problemas de Documentação

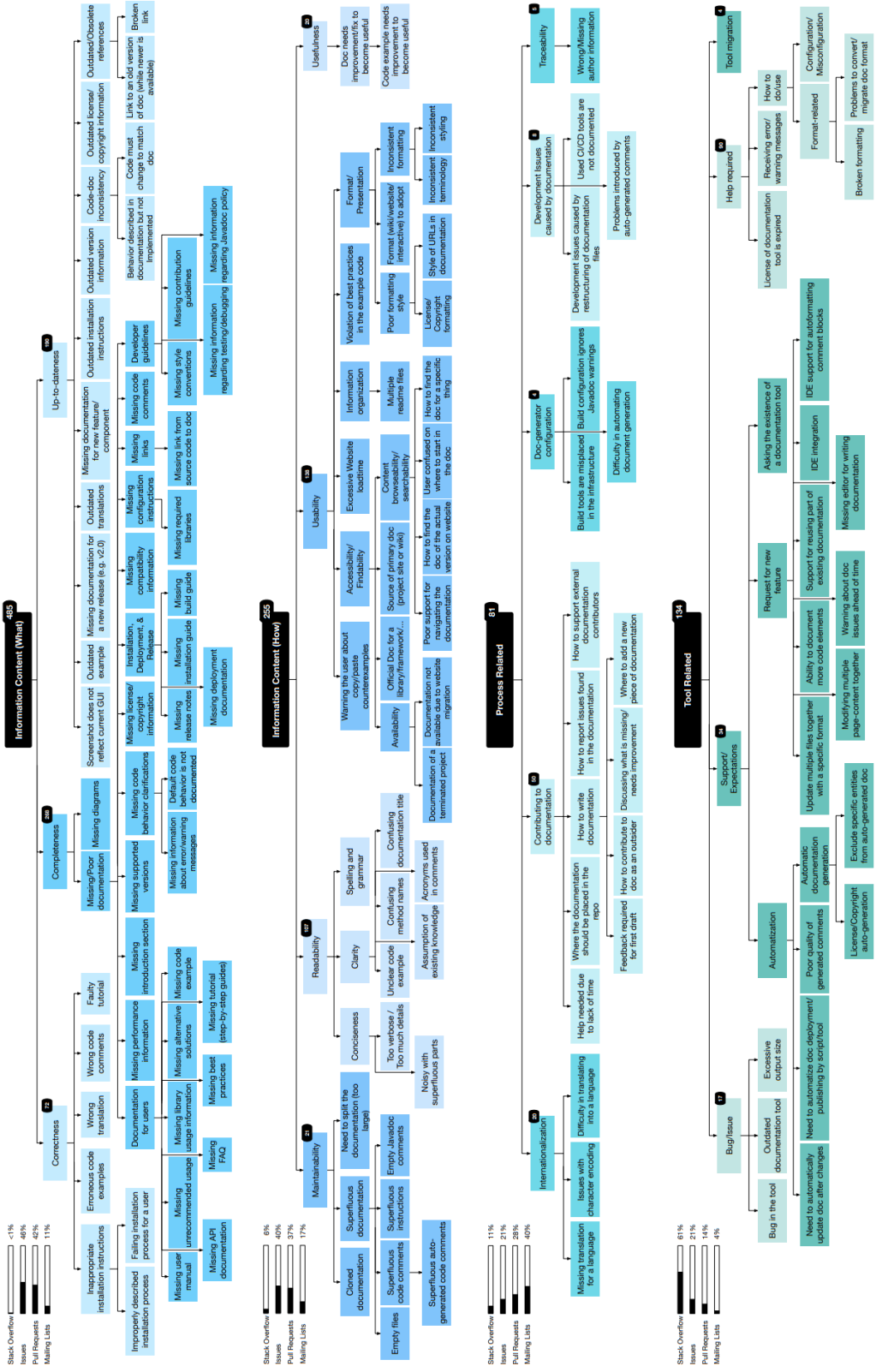

Fig. 1. Documentation Issues Taxonomy

FONTE: Aghajani et al.[5]

ANEXO B – Problemas de Documentação que são relevantes para praticantes

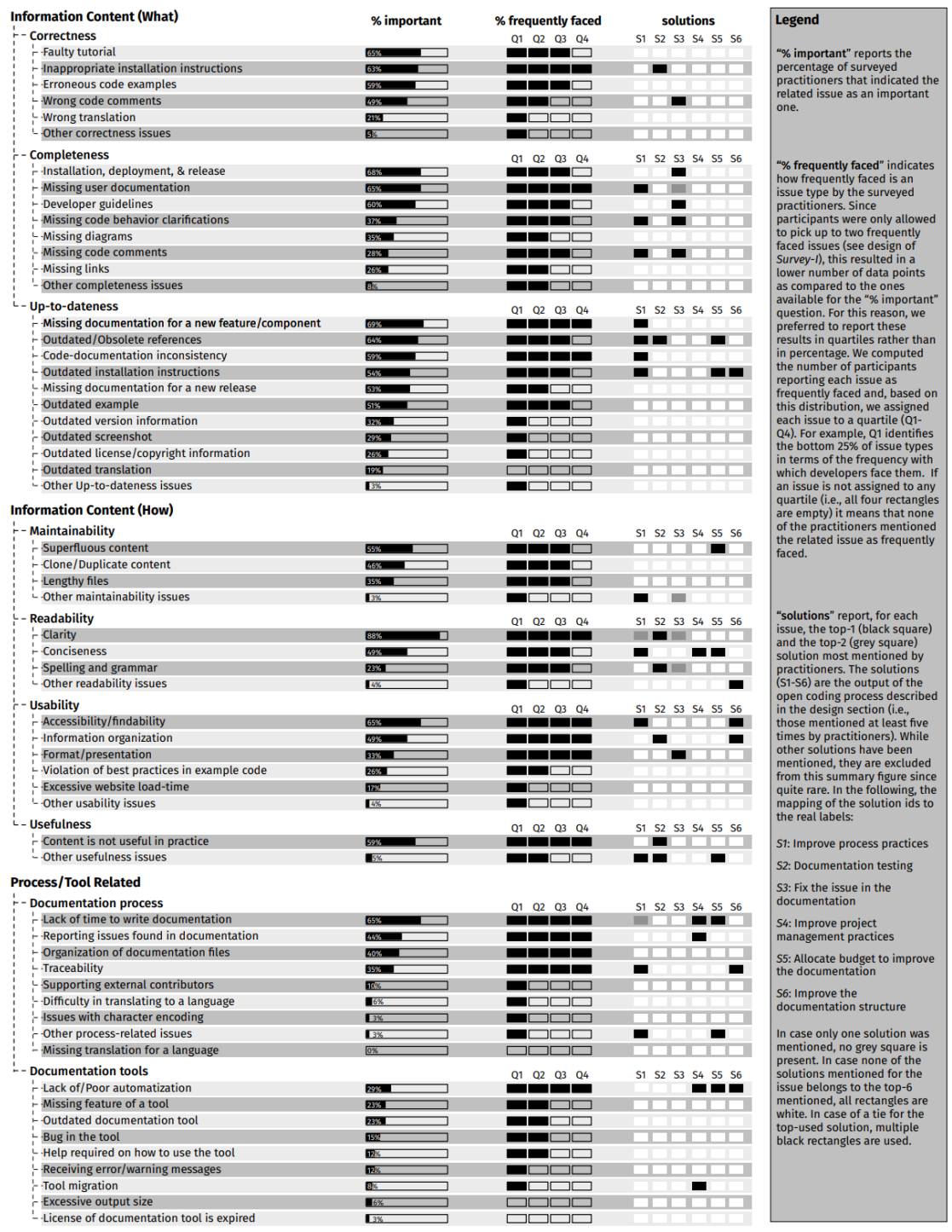

Figure 2: Documentation issues that are relevant to practitioners (RQ_1), according to the results of Survey-I.

FONTE: Aghajani et al.[6]