O IMPACTO DA DOCUMENTAÇÃO E DOS TESTES NO PROCESSO DE DESENVOLVIMENTO DE SISTEMAS

Natanael Rebelatto Comparsi[1]
Vinicius Zanchet de Lima[2]

RESUMO

Conhecimento é considerado um ativo muito importante dentro de uma empresa. Pessoas entram e saem destas empresas e muitas acabam levando consigo o conhecimento adquirido, e desta forma, estas informações acabam sendo perdidas e muitas vezes não podem ser recuperadas. Ao tratarmos de software, muitas vezes a única pessoa que tem conhecimento do funcionamento de um determinado software, é o programador que desenvolveu o mesmo. Isso inevitavelmente pode gerar transtornos. Relativo à realização de testes, podemos citar um fato muito discutido, que é a balança a qual quanto mais é investido em testes durante o processo de desenvolvimento, consequentemente menor será o gasto existente com a manutenção dele. Desta forma com o intuito de compreender e entender a importância da aplicação da documentação e dos testes de software, bem como as influências que eles causam no processo de desenvolvimento de sistemas como um todo, o presente artigo objetivou apresentar os impactos causados pelo uso correto da documentação e dos testes durante o processo de desenvolvimento de um sistema, outorgada pela pesquisa bibliográfica em meio a documentos, sites, livros e revistas, destes destacando artigos científicos, teses, dissertações e trabalhos de conclusão de curso. Apresentando como resultado que através do uso da documentação de software é possível tornar a dependência dos usuários menor e dar mais autonomia aos mesmos, uma vez que o conhecimento sobre o software é de fácil acesso e está compreensível para ele, já por meio da utilização dos testes se torna mais rápido a detecção dos erros ou anomalias existentes em um software, consequentemente poupando recursos às empresas.

Palavras-chave: Documentação; Testes; Sistemas; Impacto da documentação; Impacto dos testes.

1. INTRODUÇÃO

Os sistemas de informação começaram a ser desenvolvidos em torno da década de 1970, onde nesta época ocorreu uma crise devido à baixa qualidade com a qual os softwares eram entregues, os prazos nunca eram respeitados, havia um alto custo final e consequentemente os requisitos não eram atendidos no fim do desenvolvimento de um sistema. Consonante a isto podemos citar por exemplo o lançamento do foguete Ariane-5G, que acabou explodindo após 37 segundos de seu lançamento devido a uma falha que acabou ficando conhecida como “estouro de inteiros”, ou seja, quando uma variável especifica tenta realizar o armazenamento de um valor maior do que o definido disponível em um determinado espaço de memória existente para ela (KREBS, 2017).

Devido a isto, se fez necessário na época a criação de uma metodologia de testes que atestasse desta maneira o bom funcionamento da ferramenta desenvolvida, testes esses que podem ser divididos

em vários tipos como por exemplo usabilidade, performance, segurança, estabilidade, entre outros (PINHEIRO, 2022).

Conforme Cunha (2010), é a engenharia de software, que dentro da fase de testes acaba por fornecer informações sobre a qualidade de um sistema existente com a devida relação ao contexto em que se espera o que ele opere, e que consequentemente planeja como será realizado o desenvolvimento, para que assim todos os requisitos desejados pelo usuário sejam corretamente atendidos.

O desenvolvimento de software é um processo complexo executado normalmente em contextos com poucas certezas, onde neste contexto muito frequentemente existem mudanças de tecnologias e requisitos (OCHODEK; et al., 2018). Todo o ramo de conhecimento que é trabalhado pela engenharia de software busca sempre desenvolver estratégias que facilitem o desenvolvimento de sistemas de software como um todo. Dito isto neste contexto, destaca-se o processo de engenharia de requisitos o qual desde a década de 1990 é reconhecido por estabelecer as bases para seja desenvolvido software com qualidade (TENBERGEN; DAUN, 2019). Sommerville afirma que a engenharia de requisitos é um processo que possui como o seu principal objetivo: descobrir, analisar, documentar e consequentemente verificar os serviços que um software deve propiciar e as suas devidas restrições existentes (SOMMERVILLE, 2011).

O presente artigo se encontra estruturado da seguinte forma, primeiramente ele apresentará o método de pesquisa utilizado. Posteriormente ele é dividido em três seções e por último são apresentadas as considerações finais referentes ao mesmo. A primeira seção aborda sobre a documentação de software apresentando seus conceitos, esta seção também se decompõe em três temas menores que seriam a dívida técnica existente da documentação, o processo de gestão do conhecimento existente nas empresas realizada por meio da documentação de software e por último a importância da documentação como um todo no ambiente de desenvolvimento de sistemas. A segunda seção fala sobre os testes de software, desta forma sendo apresentado os principais testes existentes e os seus respectivos objetivos, esta seção é subdividida em dois temas, a importância dos testes de software no processo de desenvolvimento de sistemas e os principais testes de software existentes com as suas principais vantagens e desvantagens consequentes de sua utilização. A terceira seção mostra um framework conceitual exemplificando de maneira visual quais são os principais impactos do uso da documentação e dos testes no desenvolvimento de sistemas, também é citado tópicos de extrema importância que tem relação direta com os impactos gerados pela utilização deles. Tudo isto seguindo os princípios da pesquisa bibliográfica exploratória.

2. MÉTODO DE PESQUISA

O presente artigo utilizou como seu respectivo método a pesquisa bibliográfica exploratória com revisão na literatura, efetuando assim a análise em documentos, sites, livros e revistas, destes destacando-se como principais os artigos científicos, teses, dissertações e trabalhos de conclusão de curso com o seguinte objetivo de apresentar um conteúdo explicativo acerca do tema abordado.

A pesquisa bibliográfica, segundo o que é citado por Pizzani et. al. (2012, p. 54) é descrita da seguinte forma: “É a revisão da literatura sobre as principais teorias que norteiam o trabalho científico. Essa revisão é o que chamamos de levantamento bibliográfico ou revisão bibliográfica, a qual pode ser realizada em livros, periódicos, artigos científicos, sites da internet entre outras fontes”.

De acordo com o que é afirmado por Yin (2001), primeiramente deve ser efetuado o levantamento dos devidos dados encontrados com o intuito de posteriormente seguir o escopo proposto. Conhecer, analisar e esclarecer os problemas por meio de referências teóricas é o que denomina a pesquisa bibliográfica (TOFOLI, 2011).

A pesquisa sobre o tema teve início na base de dados da plataforma Google Acadêmico, e posteriormente a da Scielo, utilizando como palavras chaves para a busca impacto da documentação, impacto dos testes, importância da documentação e importância dos testes, dos quais resultaram vários trabalhos, onde foram selecionados os mais recentes e relevantes referentes ao tema abordado tendo como métrica a quantidade de citações, sem restrições de idiomas ou país de origem.

3. RESULTADOS E DISCUSSÃO

3.1 DOCUMENTAÇÃO DE SOFTWARE

De acordo com (TALITA; SOUZA, 2019) a realização da documentação de requisitos é uma prática que está relacionada diretamente à construção de um produto de software. Ela reflete as regras e os conceitos que o sistema deve conter respectivamente. No entanto, a construção da documentação necessária nem sempre é efetuada de uma maneira simples e outro ponto é que a diversidade de modelos de documentos existentes pode também ocasionar dúvidas a respeito de qual é a mais adequada a ser utilizada em um determinado projeto para a efetuação da padronização das documentações realizadas entre toda a equipe.

O objetivo central da documentação é o de compartilhar e transferir conhecimento acerca de um determinado software, todavia os processos ágeis procuram sempre valorizar a prática da comunicação face-a-face em detrimento da documentação em si. Desta maneira Chau e Maurer (2004) colocam que esta prática depende muito diretamente de interação sociais entre os respectivos indivíduos e de uma boa rede de relacionamento existente dentro da equipe. As limitações de comunicação face-a-face ficam evidentes quando ocorrem as seguintes situações demonstradas no quadro a seguir:

Quadro 1: Situações ocasionadoras de limitações de comunicação face-a-face.

SITUAÇÕES OCASIONADORAS DE LIMITAÇÕES DE COMUNICAÇÃO FACE-A-FACE
O software desenvolvido está dependendo muito da comunicação face-a-face, desta forma fazendo com que o software acabe tornando-se de difícil aplicação para grandes equipes.
A localização de integrantes do time referentes a sua situação no projeto (em caso de ausências) pode ser difícil de ser estipulada e a confiança em um compartilhamento informal de informações pode apresentar desafios em sua execução.
A prática só está facilitando o aprendizado dentro de uma determinada equipe, mas não de outras equipes existentes na organização. Assim acabam sendo utilizadas outras abordagens de transferência de conhecimento junto as práticas existentes de comunicação da respectiva metodologia ágil utilizada pela equipe o que não é o ideal.

Fonte: Elaborado pelo autor.

Dessa forma, Talita e Souza (2019) nos diz que quando estamos diante de um contexto em que existe uma dificuldade de se realizar a documentação de um documento identificado pela literatura fica notável desta forma que há um desafio em manter alinhada a evolução do sistema desenvolvido com a documentação que se é realizada, podendo assim comprometer além da comunicação entre as partes interessadas, a autonomia dos integrantes existentes na equipe e também as decisões de caráter gerencial que ficam sem nenhum suporte adequado referente ao gerenciamento dos requisitos existentes.

3.1.2 DÍVIDA TÉCNICA DA DOCUMENTAÇÃO

A dívida técnica é uma metáfora criada para objetos que não estejam finalizados, faltando partes ou que não sirvam para serem utilizados durante o ciclo de vida de desenvolvimento de um software como um todo, desta forma eles geram custos elevados e uma menor qualidade geral de um software produzido a longo prazo (SEAMAN; GUO, 2011). Ao passar dos anos, o interesse por este

tema se intensificou, aumentando consideravelmente desta forma a aplicação dos seus conceitos (LI; et al., 2015). No entanto, mesmo havendo um crescente número de estudos relativos a este assunto, as organizações ainda abordam muito pouco a respeito destes conceitos de dívida técnica e a sua respectiva aplicação de maneira direta nos processos existentes durante o desenvolvimento de um software (SILVA; et al., 2019).

Entre os inúmeros tipos de dívida técnica existentes, a documentação de software aborda acerca dos problemas que existem na documentação de projetos de softwares, desta forma estes problemas costumam ser encontrados ao serem realizadas pesquisas por documentações que não existam, que não possuam informações consistentes com o software desenvolvido, cujo as informações estejam desatualizadas ou até mesmo incompletas (SEAMAN; GUO, 2011). Um ponto extremamente crucial que deve ser fornecido a sua devida atenção necessária é o fato de que esta dívida ocorre durante a realização do gerenciamento do produto de um software, pois isto pode ser considerado como um fator-chave para o sucesso deste software (EBERT; BRINKKEMPER, 2014).

É fundamental que se faça a utilização da documentação não importa qual for o processo de desenvolvimento de software a ser realizado. Ultimamente profissionais da área de desenvolvimento de software e pesquisadores da mesma vem apresentando uma preocupação crescente sobre os benefícios gerados, os custos existentes e a qualidade referente a documentação de software na prática (ZHI; et al., 2015).

3.1.3 GESTÃO DO CONHECIMENTO

Segundo Silva (2002), o surgimento da gestão do conhecimento é datado da época de 1980 quando Nonaka e Takeuchi desenvolveram as teorias fundamentais referentes a gestão do conhecimento, assim como realizaram o início dos estudos com a finalidade de analisar qual a importância do conhecimento nas organizações. Conforme citado por Nonaka e Takeuchi (1997), a gestão do conhecimento é a capacidade de uma empresa criar e disseminar este conhecimento de uma forma a incorporar o mesmo em produtos, serviços e sistemas.

Com o intuito de melhorar o entendimento da gestão do conhecimento nas organizações Nonaka e Takeuchi (1997) desenvolveram um modelo de duas dimensões para a criação e a disseminação do conhecimento. O nome espiral vem do fato de neste respectivo modelo a criação de conhecimento ser originaria de um processo social entre as pessoas, de tal maneira que as

transformações de conhecimento são realizadas de formas interativas em forma de espiral ao invés de serem de forma unidirecionais.

Ainda segundo Nonaka e Takeuchi (1997), a primeira dimensão deste modelo é ontológica a qual possui como preocupação a criação do conhecimento organizacional em oposição ao conhecimento individual existente. Já a segunda dimensão do modelo citado é a epistemológica cujo parte da discussão da diferença entre o conhecimento tácito e explícito.

De acordo com Statdlober (2016), o conhecimento tácito é aquele que não possui nenhuma espécie de documentação formal, desta maneira acaba por somente constar na cabeça das pessoas de maneira individual e pessoal. Já o conhecimento explícito é todo o conhecimento que está documentado de forma formal, escrita, armazenada e consequentemente registrada, de forma que exista garantia que estas informações estejam retidas e que possam ser compartilhadas sem depender de certos indivíduos.

Na Figura 1, é possível se visualizar a espiral proposta por Nonaka e Takeuchi (1997) onde é destacado a ligação existente entre o conhecimento tácito e explícito. Nesta espiral proposta por eles, são utilizados quatro modos de conversão (Socialização, Externalização, Internalização Combinação), sendo assim a partir desta espiral é possível se explicar a criação do conhecimento dentro de uma organização.

Figura 1: Espiral do conhecimento.

Fonte: Adaptado de Nonaka e Takeuchi, 1997.

Desta forma Somensi (2020) afirma que a Socialização corresponde a comunicação e a troca de conhecimento existente entre os indivíduos, de forma formal ou informal. A Externalização corresponde ao processo de documentar o conhecimento que está impregnado na cabeça das pessoas. A Internalização pode ser considerada como a parte do aprendizado, quando estudamos determinados conteúdos e levamos este conhecimento para dentro da organização. E por fim, a Combinação é a situação quando se é realizada a utilização de conteúdos já registrados para se realizar a composição de novos.

Adentrando a gestão do conhecimento se faz necessário conceituarmos o que são ativos de conhecimento e o que é determinado na etapa de mapeamento deste conhecimento. Conforme Statdlober (2016), ativos de conhecimento, também são conhecidos como o capital intelectual das empresas, são respectivamente ativos intangíveis que fazem referência ao conhecimento como o "Know-how", ou seja, o saber-fazer, as melhores práticas, propriedade intelectual, entre outros. Os ativos de conhecimento podem ser definidos na prática como conhecimentos explícitos e tácitos de uma organização (SOMENSI, 2020).

O mapeamento do conhecimento é uma etapa fundamental e muito importante dentro da gestão do conhecimento, pois é nesta etapa que serão determinados quais serão os ativos de

informação existentes dentro de uma organização, assim como os fluxos de informação e os respectivos detentores destas informações (STATDLOBER, 2016).

Conforme Somensi (2020) este processo funciona como uma espécie de inventário dos conhecimentos existentes em uma determinada empresa, tanto os explícitos como os tácitos, de forma a se verificar quais os indivíduos que detém o conhecimento acerca dos processos existentes na organização. Para que esta etapa transcorra de maneira adequada e seja efetiva, algumas informações devem ser verificadas, como quais os conhecimentos importantes e desejáveis ao processo a ser realizado, as funções das pessoas envolvidas, os ativos de informações explícitos e tácitos, a criticidade destes ativos e quais as lacunas de conhecimento existentes.

3.1.4 A IMPORTÂNCIA DA DOCUMENTAÇÃO

A elaboração da documentação nos projetos de software é muito importante pois é nela que estão presentes os requisitos fundamentais na elaboração de um determinado software e é através destes requisitos que um desenvolvedor poderá realizar os devidos testes para concluir se eles foram atendidos ou não. Consonante a isto Espinha (2016) afirma que preservar algum registro de como cada item foi feito, quais serão as respectivas decisões que virão a ser escolhidas , como que isto tudo será realizado e o porquê isto irá ser feito referente a todas as fases de um determinado projeto é uma maneira de a equipe de desenvolvimento ser protegida a respeito de todo o desenvolvimento que irá ser realizado referente ao trabalho desenvolvido, todavia mesmo possuindo uma grande importância é muito comum vermos projetos com pouca ou até mesmo nenhuma documentação o que pode vir a gerar um impacto negativo no termino do mesmo.

Existem alguns passos conforme citado por Espinha (2016) para que exista desta forma uma boa documentação referente ao sistema que está sendo desenvolvido por determinada equipe o que desta forma faz com que a gestão do desenvolvimento deste sistema seja mais fácil do que seria normalmente sem a existência de nenhuma documentação:

Quadro 2: Passos para uma boa documentação por parte da equipe de desenvolvimento.

Passos para uma boa documentação	
1. A existência de uma comunicação clara e acertada deve estar registrada.	Durante toda a realização de uma documentação a comunicação existente entre a equipe a qual participou do desenvolvimento de um determinado projeto deve estar completamente clara com o intuito de não existirem erros na realização dela como por exemplo alguma brecha que poderá ocasionar a geração de alguma dúvida

	posteriormente por parte do usuário, ou até mesmo alguma interpretação que venha a ocorrer diferente da esperada por quem realizou a documentação deste determinado software.
2. Desenvolvimento de um histórico de ações para uma avaliação mais minuciosa.	Deverá ser realizado durante o desenvolvimento do projeto um histórico que possua todas as principais ações que foram realizadas, com o intuito de se poder realizar uma avaliação realmente minuciosa disto tudo, para que posteriormente seja preservado um histórico deste projeto desenvolvido, isto pode ser considerado um dos principais fatores existentes com que acabam por fazer com que a documentação seja deverás importante de ser realizada.
3. A existência de controle para o desenvolvimento do projeto com o um todo.	Durante o desenvolvimento de um determinado software é comum termos que o ambiente dele seja muito incerto e consequentemente bastante dinâmico, desta forma isto acaba por colocar a equipe responsável por determinado projeto como sempre a frente de muitas situações as quais podem muito facilmente fugir de controle de forma muito inesperada.
4. Realização de um a alinhamento sobre as informações existentes de um projeto cujo será realizado a inserção de novos membros no mesmo.	Se faz sempre necessário se realizar um alinhamento acerca das informações existentes em um determinado projeto no momento da inserção de novos membros em uma determinada equipe, por este motivo que se é tão necessário se possuir uma documentação abrangente e que esteja sempre fiel ao projeto que está sendo desenvolvido, pois assim durante a inserção de um novo membro ele poderá estar a par de todo este projeto conhecendo as funcionalidades já existentes no mesmo, desta forma está pessoa irá com toda certeza contribuir de maneira mais eficaz nas atividades que ele for destinado a realizar posteriormente.
5. A criação e manutenção de uma base para a realização de tomadas de decisões.	Durante a realização de um projeto as tomadas de decisões referentes ao mesmo nunca podem ser tomadas de maneira aleatória sendo assim se faz necessário sempre que estas decisões por parte do time de desenvolvimento sejam tomadas baseadas em fatos e preferencialmente que elas sejam comprováveis de alguma forma, desta maneira estes dados se fazem explicitamente necessários de estarem registrados na documentação realizada referente a este determinado projeto que está sendo desenvolvido.
6. Realização de um reporte de status referente ao projeto para todos os stakeholders participantes.	Por último mais não mesmo importante se faz extremamente necessário a realização da documentação, pelo fato dela manter sempre todos os stakeholders referentes a um projeto informados acerca de todas as alterações realizadas no mesmo, desta maneira eles sempre possuirão dados que sejam precisos referentes a este determinado sistema e também consequentemente frequentes devido ao fato de um sistema como um todo estar sempre em desenvolvimento seja pela manutenção ou a implementação de novas funcionalidades por exemplo. Dito isto assim isto acaba mantendo a equipe do projeto focada no projeto em si e não sempre procurando responder dúvidas e mal-entendidos por parte dos stakeholders existentes no projeto.

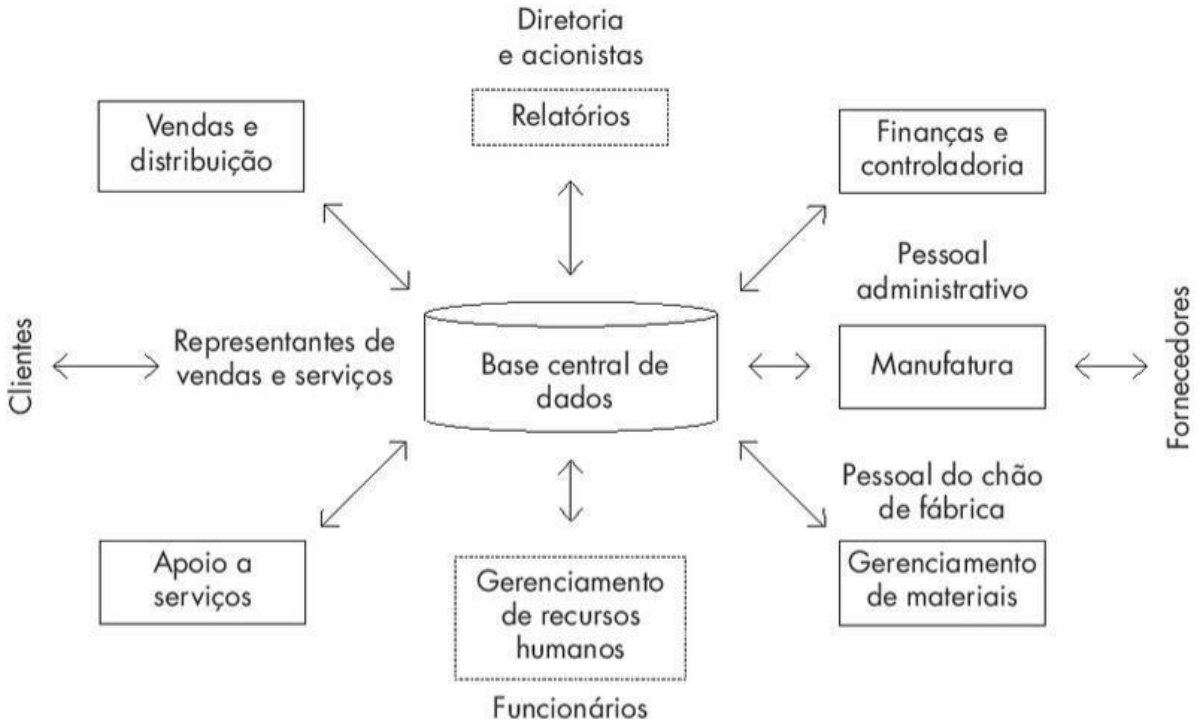
Fonte: Adaptado de Espinha, 2016.

Durante o processo de teste de um software como um sistema por exemplo, as falhas que forem apresentadas, ou seja, descobertas durante este processo de testes devem ser registradas com uma quantidade de informações suficientes com o intuito de que este respectivo defeito possa ser reproduzido posteriormente, para que assim ele seja analisado e por fim corrigido (IEEE, 1998).

Antes da apresentação dos resultados referentes a uma pesquisa realizada em uma empresa metalúrgica, situada na serra gaúcha se faz necessário a conceituação do que é um ERP para fins de complementação. De acordo com Caiçara Junior (2015), Enterprise Resource Planning (ERP) ou

conforme tradução Planejamento de Recursos Empresariais, é um sistema de gestão que permite a integração de todas as informações existentes em uma determinada organização. Já para Davenport (1998), ERP é um sistema composto por um conjunto de softwares cujo possuem como objetivo organizar, padronizar e integrar as informações referentes a uma determinada organização. Esta integração de dados possibilita o acesso a uma fonte realmente confiável de informações localizada em um banco de dados centralizado e em tempo real. Na Figura 2 é possível verificar, segundo o autor, a estrutura de funcionamento de um sistema ERP.

Figura 2: Estrutura de funcionamento de um sistema ERP.



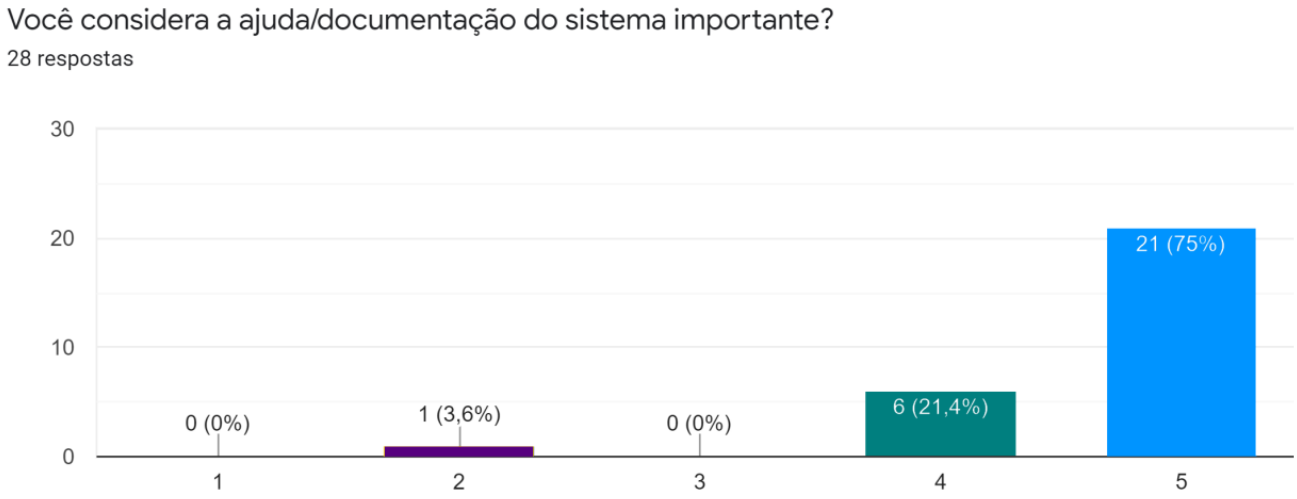
Fonte: Adaptado de Davenport, 1998.

Conforme pesquisa de caráter exploratório descritivo e implementação de uma solução em um sistema de ERP da empresa metalúrgica Tramontina, situada na serra gaúcha realizada por Somensi (2020) para padronização e aprimoramento da documentação, ajuda e gestão dos conhecimentos de um software ERP tanto para quem desenvolve quanto para quem tem dúvidas quanto ao seu uso, fica claro a necessidade e a importância de gerir o conhecimento da organização como um ativo importante e estratégico para a empresa.

De acordo com Somensi (2020) o ERP utilizado na empresa Tramontina é totalmente desenvolvido internamente, desta forma ele é completamente customizado para a realidade desta empresa e suas respectivas particularidades existentes. Sendo assim para novos usuários, a documentação do ERP é de extrema importância para facilitar a adaptação ao uso do sistema. No período de realização desta pesquisa foram constatados em torno de 853 tickets de dúvida no sistema de Help Desk da empresa totalizando 3059 horas despendidas pelos programadores para responder estas dúvidas entre os anos de 2016 e 2019.

Neste trabalho efetuado por Somensi (2020) o seu objetivo principal era melhorar a forma de como os softwares da empresa eram documentados por meio da utilização de uma plataforma mais moderna e atual com o intuito de estimular e melhorar consequentemente a experiencia da documentação a ser realizada no sistema. Desta forma conforme formulário realizado e aplicado por Somensi (2020) para o grupo de desenvolvedores da empresa Tramontina o qual era formado na época por 30 pessoas e destes, 28 responderam à pesquisa, a primeira pergunta tinha como objetivo verificar se os desenvolvedores acreditavam que a ajuda e documentação do sistema era importante. Conforme demonstrado na Figura 3, 94,4% dos desenvolvedores que responderam o formulário, acreditam que a ajuda é importante ou muito importante.

Figura 3: Importância da documentação do sistema.



Fonte: Adaptado de Somensi, 2020.

Foi constatado através das informações coletadas e a sua devida implementação da solução que 82,1% dos desenvolvedores aprovaram as mudanças realizadas e que isto gerou uma motivação extra por parte dos mesmos para a realização da documentação do sistema Tramontina. Dito isto através desta pesquisa realizada por Somensi (2020) podemos afirmar a tamanha importância que a documentação de software possui na gestão do conhecimento de uma determinada empresa nos dias de hoje referente aos sistemas que são desenvolvidos na mesma como um todo.

3.2 TESTES DE SOFTWARE

Um teste de software é um método que realiza a validação da execução de um programa de uma maneira controlada, com o principal objetivo de avaliar o seu devido comportamento durante a sua utilização. A atividade de um teste de software exige planejamento, conhecimento, projeto, execução, recursos que sejam necessários para a realização dele, o seu devido acompanhamento e uma grande interação entre as equipes existentes no projeto (CRESPO, 2004). Já de acordo com Barbosa (2009) um teste de software realiza a exibição da atividade que realiza a elaboração do processo de verificação e validação de um respectivo software, o qual é a técnica existente mais utilizada em todo o âmbito da confiabilidade e da garantia de um software.

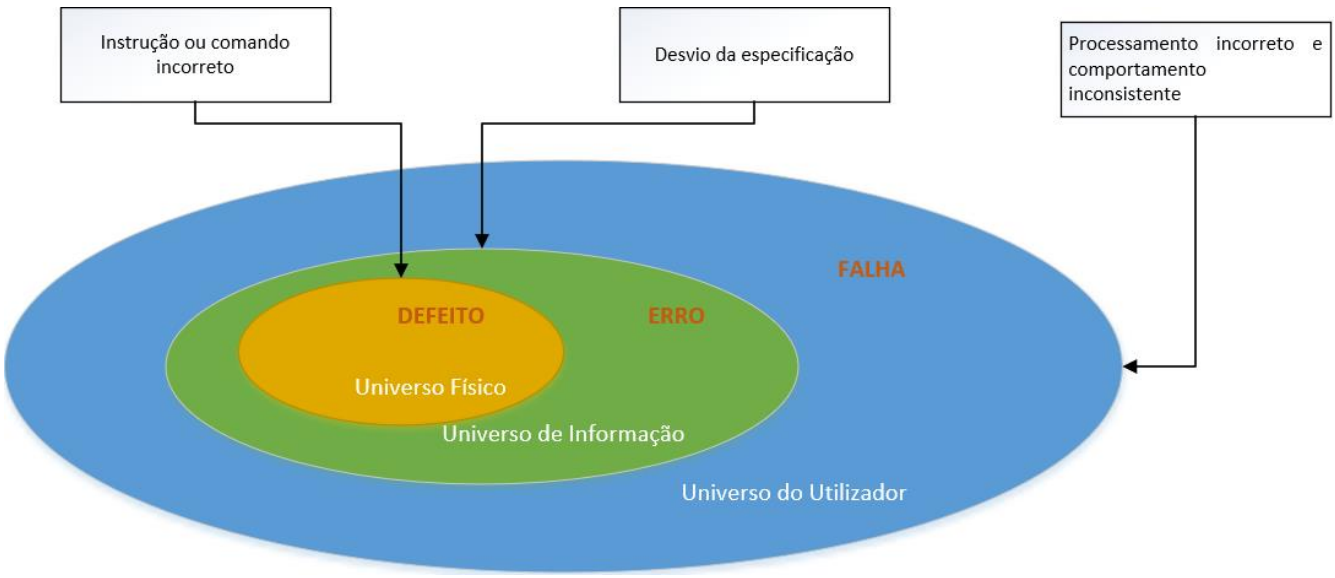
Ao realizar um teste de um software, podemos nos deparar com algumas dificuldades, tais como: ser um processo que possui um custo elevado, a existência de falta de conhecimento sobre a relação existente entre o custo e benefício do teste que se é realizado, a falta de profissionais na área de teste, a inexistência de conhecimento de técnicas de testes adequadas para cada determinada situação e a não preocupação com a atividade de teste na fase final de um projeto a ser realizado (CRESPO, 2004).

Quando abordamos sobre testes de software isto nos leva consequentemente a um debate referente a conceitos relacionados a esta atividade, sendo assim é preciso esclarecer a diferença entre defeitos, erros e falhas. O Defeito é resultante de um ato inconsciente, que é cometido ao se tentar entender determinada informação existente ou fazer o uso de um método ou uma ferramenta em específico para se resolver um problema por exemplo. Erro é uma revelação fundamentada de um determinado defeito em um software onde é notada a diferença entre o valor obtido no mesmo e o valor realmente esperado pelo utilizador no caso o usuário deste software. Já a Falha é um comportamento operacional de um software que acaba por ser diferente do esperado pelo seu usuário,

tal comportamento pode ser causado por múltiplos erros existentes neste software e entre eles alguns podem não causar uma falha (TRIGO, 2017).

Ao observar a Figura 4 é possível visualizar que há uma diferença entre os conceitos citados, defeitos podem efetuar o equilíbrio da revelação de erros que tenham ocorrido em um determinado software, ou seja, no caso o desenvolvimento de um software é feito de uma forma diferente ao que se foi especificado no projeto. Por fim, os erros tendem a gerar falhas, ou seja, comportamentos de um software que sejam inesperados pelo usuário do mesmo, acabam por afetar este utilizador final deste software o que pode desta maneira inviabilizar a utilização do mesmo (NETO, 2014).

Figura 4: Defeito, Erro e Falha.



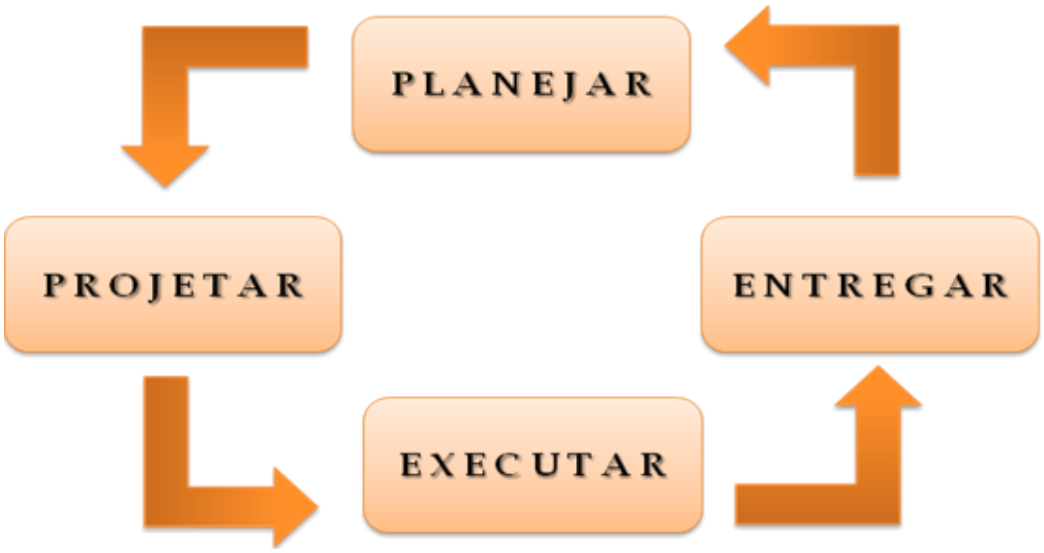
Fonte: Adaptado de Neto, 2014.

A realização da verificação das especificações é o proposito fundamental existente para a realização de testes. Damos o nome de especificação a descrição daquilo que o software deve realizar a partir do que foi analisado anteriormente, desta forma é nesta etapa que é apresentada uma solução para que os respectivos problemas levantados no processo de análise sejam resolvidos durante o processo de desenvolvimento. A especificação é uma forma de mensagem que é realizada diretamente entre a equipe de desenvolvimento e o cliente (TONSIG, 2008).

As atividades de teste que são realizadas em um software não se resumem somente na execução dele, é necessário efetuar um planejamento deste teste, escolher a condição de teste do mesmo e por fim elaborar os critérios de avaliação deste. Após este planejamento ter sido realizado é elaborado o respectivo teste e retiradas as devidas conclusões ou resultados acerca do mesmo. Desta forma é necessário que o processo de teste efetuado seja transparente, desde o seu planejamento até a apresentação dos resultados obtidos, sendo que o tester, ou seja, a pessoa responsável pelo teste deste software contribua para essa transparência, assim mantendo todas as partes envolvidas informadas sobre o processo de teste efetuado (TRIGO, 2017).

Um processo de teste de software são vários passos agrupados entre si que são constituídos por atividades, métodos e práticas que são utilizadas para se testar um software. Um processo de teste de software possui os seguintes subprocessos conforme Figura 5: planejamento, projeto, execução e registro, ou seja, a entrega do resultado do teste efetuado (JUNIOR, 2007).

Figura 5: Processo de Teste de Software.



Fonte: Adaptado de Eliza e Lagares, 2012.

Os testes de software podem ser aplicados a várias áreas existentes dentro do desenvolvimento de software como por exemplo acessibilidade, segurança, usabilidade, entre outros. O ideal é que os projetos a serem executados realizem pelo menos alguns testes entre os existentes dentro de cada

respectiva área citada acima de forma a garantir um funcionamento adequado do produto a ser entregue para o cliente (PINHEIRO, 2022).

3.2.1 A IMPORTÂNCIA DOS TESTES DE SOFTWARE

Existem vários fatores que justificam a realização de testes de software, apesar disso, os fatores principais que acabam por levar os gestores de tecnologia de informação à realização de testes de software de acordo com Carvalho (2010) são: diminuir o risco, diminuir os custos que estejam associados à produção e a manutenção do software e por fim garantir que todas as expectativas e especificações definidas e esperadas pelo cliente sejam cumpridas de acordo com o que foi preestabelecido anteriormente, para que assim não haja nenhuma insatisfação por parte do cliente.

Os testes de software asseguram um equilíbrio na entrega do software desenvolvido, garantindo desta forma não somente o funcionamento correto da ferramenta, mas também o respectivo atendimento aos princípios básicos de segurança e aos requisitos contratados (ROCHA, 2011). Ainda podem ser realizados testes de usabilidade, performance, estabilidade, entre outros, de forma a se atestar quais os limites existentes de desempenho em um determinado sistema.

Dessa forma, o conhecimento acerca dos tipos de testes existentes, como eles são feitos e a sua abordagem realizada permite que os desenvolvedores ou o time de testadores caso exista ganhe tempo e, conforme a complexidade e risco identificados no sistema, sejam realizados os testes mais apropriados (JUNCKES; MORGADO, 2013).

O teste de software é uma fase deveras importante existente no ciclo de vida de desenvolvimento de um software. Desta forma ele é um elemento vital e pode fornecer excelentes resultados caso o teste seja feito de maneira adequada e eficaz. Infelizmente, o teste de software geralmente é muito menos formal e rigoroso do que ele deveria ser e a principal razão para isto é porque tem-se lutado ao longo do tempo para se definir as melhores práticas, metodologias, princípios e padrões para a realização de um teste de software ideal. Sendo assim para se realizar testes de software de maneira eficaz e consequentemente eficiente, todos os envolvidos na realização destes testes devem estar familiarizados com os objetivos básicos de teste de software, a sua importância, tipos, limitações e conceitos existentes (PINHEIRO, 2022).

Um fator muito importante para a realização de testes de software está cada vez mais ligado com o fato de as organizações dependerem explicitamente de mais sistemas que suportem os seus respectivos processos de negócio existentes, pois consonante a isto os sistemas de informação estão

cada vez mais complexos, incluindo os muitos componentes utilizados em todo o seu processo de desenvolvimento (CARVALHO, 2010).

De acordo com Pinheiro (2022) a realização de testes de software para identificar falhas no código desenvolvido é uma tarefa muito importante no desenvolvimento de software, todavia ele tem recebido pouca atenção nos dias de hoje. Consonante a isto ele também afirma que no âmbito de testes de software existem diversos tipos de testes que são feitos para diferentes propósitos para que desta forma um produto de qualidade seja desenvolvido, sendo assim o principal objetivo por trás disso é o desenvolvimento de um software livre de erros.

Além dos benefícios gerados através do uso de testes de software, outros benefícios práticos importantes são (CARVALHO, 2010): a otimização de todo o processo de desenvolvimento existente, a continua melhoria da qualidade no desenvolvimento, ou seja, a criação de documentação específica para o software desenvolvido juntamente com existência de requisitos para a sua correta implementação e documentação posteriormente, a garantia de que os requisitos serão entregues, a imposição de critérios de qualidade que estejam bem definidos e consequentemente bem esclarecidos para os fornecedores deste software e por último a satisfação do cliente.

A verificação adequada de um software levará a menos problemas presentes durante a fase de validação deste software desenvolvido, pois a maioria dos problemas já foi descoberto e consequentemente corrigido durante a verificação e a sua validação adequada o que desta maneira irá garantir que o sistema seja desenvolvido de acordo com os artefatos de software que foram finalizados durante o processo de verificação. Portanto, é importante conhecer todas as atividades de testes, pois o processo de validação e verificação dele andam de mãos dadas e reduz desta forma o retrabalho que possa vir a existir e consequentemente os custos futuros (PINHEIRO, 2022).

3.2.2 PRINCIPAIS TIPOS DE TESTES DE SOFTWARE

Quando falamos sobre testes de software devemos ter em conta que eles estão divididos em tipos variados, cada um com a sua divisão ocorrendo de acordo com o seu objetivo particular. Para clarificar estes conceitos irão ser apresentados no quadro a seguir os principais tipos de testes de software existentes (CATARINA, 2015):

Quadro 3: Principais tipos de testes de software.

TESTE DE SOFTWARE	CONCEITO
1. Teste de configuração	Este teste é bem simples ele praticamente representa um processo de instalação a ser realizado para um usuário deste sistema, para que ele possa possuir este software instalado em seu computador.
2. Teste de instalação	Durante a realização deste teste é verificado se o software que foi desenvolvido está cumprindo com o que foi planejado no que diz respeito a instalação do mesmo em vários tipos de hardwares diferentes, por exemplo são realizados testes para se verificar se ele funciona corretamente em cenários como pouco espaço de memória na máquina a qual este software foi instalado, ou se alguma interrupção venha a existir durante a instalação deste software, como por exemplo a de acesso à internet que pode vir a ocasionar algum erro.
3. Teste de integridade	Este teste é realizado com o intuito de se testar se o respectivo software que está sendo desenvolvido realmente é resistente as falhas, ou seja, em outras palavras é verificado se ele é considerado um software robusto na prática.
4. Teste de segurança	Realiza um teste para verificar se um determinado sistema e os dados existentes no mesmo como um todo, estão sendo realmente acessados de uma maneira considerada segura hoje em dia pelo autor destas determinadas ações que estão sendo realizadas neste sistema.
5. Teste funcional	Este teste verifica se os requisitos funcionais existentes realmente estão de acordo com o que foi definido, isto inclui as funções existentes no mesmo e os seus respectivos casos de uso desenvolvidos.
6. Teste unitário	Verifica por meio da realização de testes se um respectivo componente quando isolado do restante do sistema desempenha suas funções corretamente, isto também se aplica a uma classe de um sistema por exemplo.
7. Teste de integração	Este teste realiza um teste com o intuito de se realizar a verificação de um ou mais componentes existentes em um determinado sistema, desta maneira é testado se quando combinados realmente executam as suas funções de maneira satisfatória sem nenhum problema sendo ocasionado durante este processo.
8. Teste de volume	Verifica se o comportamento existente para um determinado sistema com a sua operação consistindo em um volume considerado normal de transações, sendo efetuadas neste ambiente, juntamente a mesmo volume de de dados envolvendo o banco de dados durante um período considerado extenso.
9. Teste de desempenho/ performance	Este teste é composto de três tipos distintos: teste de carga, teste de estresse e teste de estabilidade cada um com sua funcionalidade específica.
10. Teste de usabilidade	Este teste é realizado com o objetivo de ser focado na utilização do usuário, ou seja, a experiencia que ele irá ter com o seu uso, é verificado se a interface está realmente consistente como um todo, em seu layout, e nos aspectos que dizem respeito ao acesso às funcionalidades existentes no mesmo.
11. Testes de caixa branca e caixa preta	O teste de caixa branca consiste no teste de partes isoladas existentes de um código já o teste de caixa preta consiste em que sejam realizados os testes das funcionalidades sem que seja considerado a sua implementação ou seja sem que seja considerada a parte referente ao código em si.
12. Teste de regressão	É realizado um novo teste referente aos componentes do software de modo que seja verificado se existe alguma modificação que tenha sido realizada recentemente a qual tenha ocasionado algum efeito indesejado no software.
13. Teste de manutenção	Realiza o teste se a respectiva mudança referente ao ambiente não ocasionou nenhuma interferência no funcionamento do sistema.

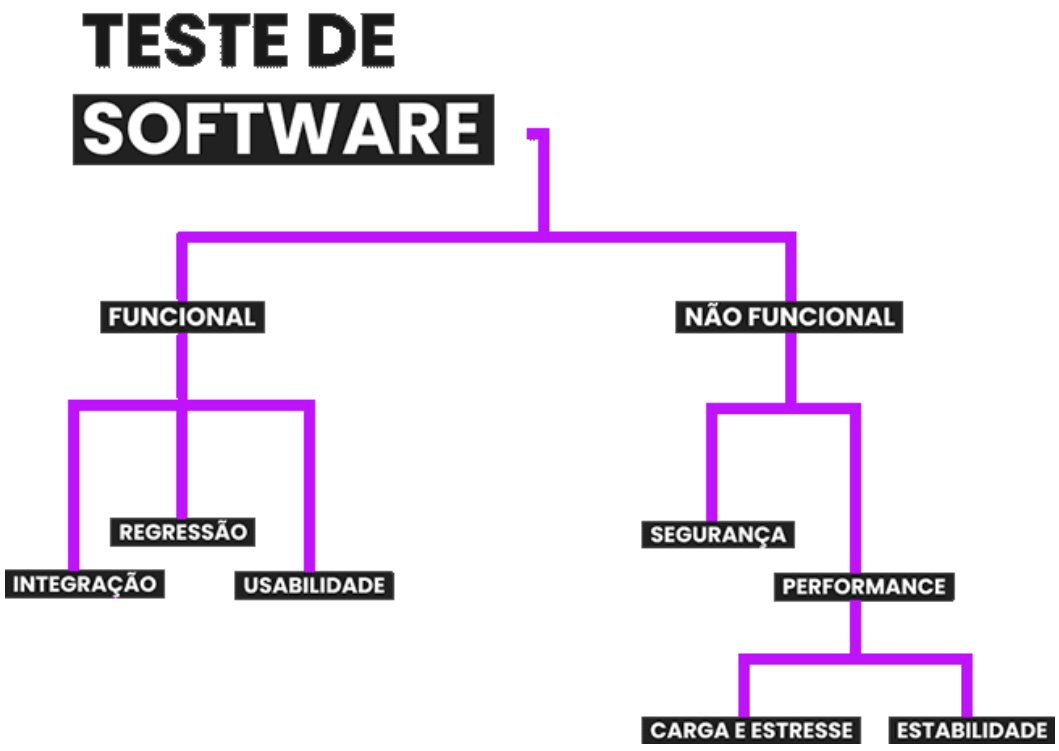
Fonte: Adaptado de Catarina, 2015.

De acordo com Pinheiro (2022) os testes de software podem ser aplicados a várias áreas do desenvolvimento como um todo, entre eles podemos citar segurança, acessibilidade, usabilidade,

entre outros. O ideal é que os projetos em realização façam pelo menos alguns testes dentro de cada área de forma a ser assegurado o funcionamento adequado do produto que será entregue ao cliente.

A Figura 5 resume os principais tipos de testes funcionais e não funcionais, e cabe a respectiva equipe existente no projeto verificar em quais testes deverão ser focados com uma maior ênfase, conforme os riscos identificados no projeto, pois uma execução de maneira aprofundada de todos os testes demandaria um custo e um tempo que poucos clientes estariam dispostos a pagar além de um maior tempo de desenvolvimento recorrente (PINHEIRO, 2022).

Figura 6: Principais testes de software funcionais e não funcionais.



Fonte: Adaptado de Portal Ivory IT, 2021.

Após a apresentação destes testes se faz necessário a realização de um maior aprofundamento acerca dos testes funcionais e não funcionais. Através dos testes de usabilidade é possível entender o comportamento por parte do usuário do sistema. Por meio dos testes de performance é possível avaliarmos como a aplicação vai se comportar, principalmente em casos extremos. Já através dos testes de segurança é possível encontrarmos falhas nas aplicações desenvolvidas antes que algum

usuário mal-intencionado faça isso, e consequentemente cause danos a organização (PINHEIRO, 2022).

A seguir no quadro 4 é apresentado uma comparação entre os testes de performance, que englobam alguns itens mencionados até o momento, e os testes funcionais, tal qual também realizam os testes do comportamento da aplicação, todavia são voltados somente a um usuário (CAMPOS, 2015).

Quadro 4: Comparativo entre os testes de performance e funcionais.

TESTE DE PERFORMANCE	TESTE FUNCIONAL
Não realiza o teste do front-end da aplicação desenvolvida, ou seja, somente realiza o teste das suas funcionalidades existentes.	Testa o front-end da aplicação, bem como a sua usabilidade e respectivas funcionalidades existentes.
Testa a escalabilidade da aplicação desenvolvida e monitora o uso dos recursos de hardware.	Não testa a escalabilidade da aplicação e monitora o uso dos recursos de hardware.
Projetado com o intuito de determinar como uma aplicação/sistema irá realizar as suas atividades definidas ao longo do tempo.	Não possui a capacidade de determinar como uma aplicação/sistema irá realizar as suas atividades definidas ao longo do tempo.
Requer uma aplicação totalmente funcional para que seja possível a sua execução adequada em cenários ideais.	Não requer uma aplicação totalmente funcional para que os cenários existentes sejam executados adequadamente.
Vários usuários.	Um usuário.

Fonte: Adaptado de Campos, 2015.

De acordo com Campos (2015) as principais vantagens existentes destes testes são: a redução do custo de mudanças no software e de sistema, aumento dos lucros, aumento da satisfação do usuário do sistema, controle total do desempenho que um sistema pode proporcionar, entre outros. Já como restrições, tem-se também que existe uma grande complexidade na elaboração destes testes, o que acaba por requerer uma simulação de uma grande variedade de cenários para a realização dos mesmos, consonante a isto as ferramentas de automação muitas das vezes acabam que por possuir um custo elevado, em contrapartida a isto ainda é citado que o ambiente real de produção nunca pode ser completamente simulado, e se faz necessário sempre ter um ambiente adequado para simulação (CAMPOS, 2015).

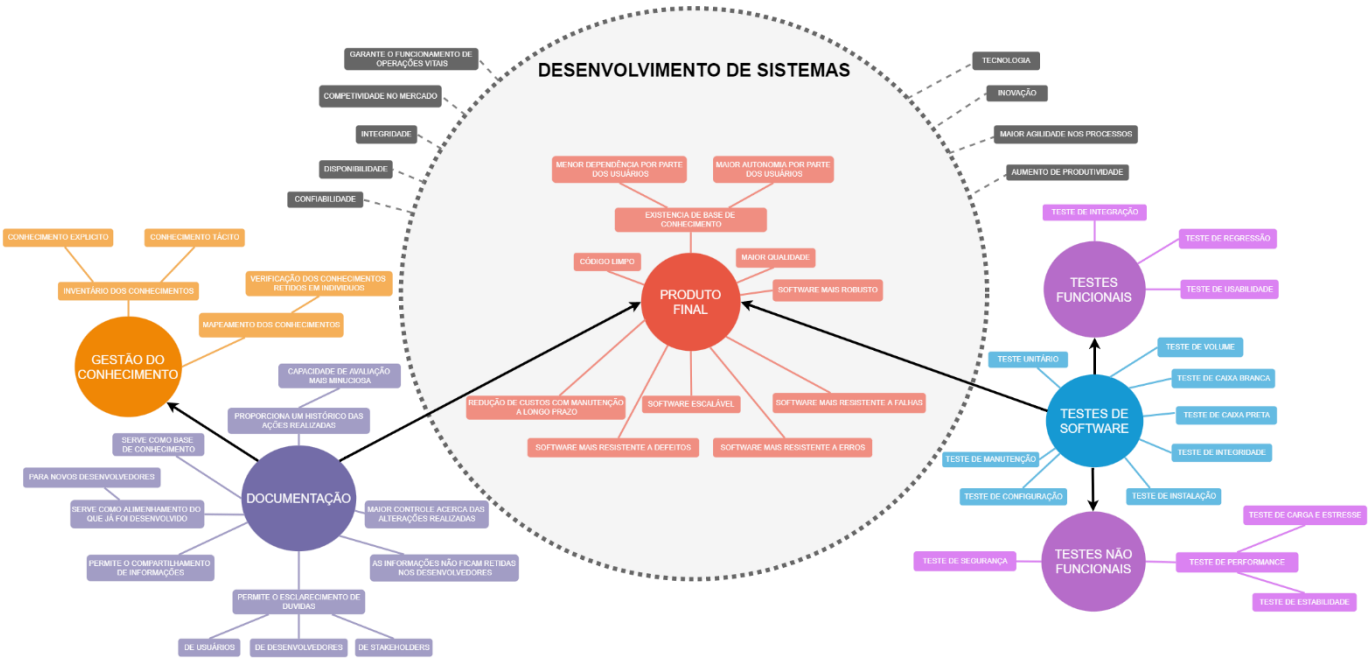
3.3 FRAMEWORK CONCEITUAL

O modelo teórico desenvolvido explica a maneira de como a utilização da documentação e dos testes durante o processo de desenvolvimento de sistemas contribui diretamente no produto a ser desenvolvido. Este está composto por constructos denominados como produto final, documentação e testes de software, seguido das suas respectivas dimensões existentes denominadas gestão do conhecimento, testes funcionais e testes não funcionais, cada uma abordando de maneira mais profunda os constructos existentes no framework desenvolvido.

O fluxo desenvolvido começa a partir das duas dimensões existentes de maneira externa ao desenvolvimento de sistemas em si onde é tratado especificamente de codificação neste caso abordado no framework, denominadas testes de software e documentação, que impactam de maneira direta no constructo produto final. Nos testes de software existem diversos tipos de testes que podem ser realizados para diferentes propósitos, para que desta maneira um produto de qualidade seja entregue ao final de todo o processo de desenvolvimento de um respectivo software, desta forma sendo o principal objetivo por trás disto a elaboração de um software livre de erros conforme afirmado por Pinheiro (2010). Já referente a documentação se faz deveras importante citarmos que o seu principal objetivo é realizar a transferência e o compartilhamento de conhecimento conforme Silva (2020), sendo assim este objetivo vai realmente em consonância com os resultados apontados por meio deste framework desenvolvido.

O constructo testes de software resulta em mais duas dimensões sendo uma delas, testes funcionais, que conforme Janssen um teste funcional é um modo de se realizar a verificação de um software de tal forma a garantir que este software possui todas as funcionalidades necessárias especificadas nos seus requisitos funcionais definidos. A outra dimensão se denomina testes não funcionais, que conforme aborda o Portal BayMetrics (2022), sobre os principais testes não funcionais existentes, como por exemplo, os testes de segurança onde o mesmo cita que se referem ao processo existente de verificação de vulnerabilidades nos sistemas desenvolvidos, ou a existência de alguma brecha que venha a servir como uma porta de entrada a um usuário mal intencionado que possua acesso ao sistema como um usuário registrado, ou até mesmo o servidor ou a base de dados referente ao mesmo. A seguir na Figura 7 é apresentado um modelo teórico dos impactos causados pelo uso da documentação e dos testes no processo de desenvolvimento de sistemas.

Figura 7: Framework conceitual.



Fonte: Desenvolvido pelo autor.

4. CONSIDERAÇÕES FINAIS

De acordo com o objetivo de apresentar os impactos causados pelo uso correto da documentação e dos testes durante o processo de desenvolvimento de um sistema, este artigo demonstrou que muitos times de desenvolvimento que realizam seu trabalho utilizando processos ágeis acabam que por menosprezar a importância da documentação de um software como uma ferramenta de apoio a capacidade de escalabilidade de suas devidas soluções e de seu respectivo processo produtivo. Deixando desta forma de ser observado, de acordo com os valores ágeis utilizados, que embora um software em funcionamento seja bastante valorizado no âmbito da área de desenvolvimento, possuir uma documentação abrangente que contenha muitas informações uteis também pode ser muito necessário.

Atualmente na era em que vivemos, em que tudo evolui de maneira muito veloz, o conhecimento acaba que por se tornar obsoleto de forma muito rápida, podendo desta maneira ser considerado um produto perecível. Desta forma, a atualização dele é tão fundamental quanto sua identificação, captura e armazenamento. Conhecimento sem a sua devida atualização tende a se tornar obsoleto com o tempo.

Através do uso da gestão do conhecimento por meio da realização da documentação de software, essas informações vitais e importantes para estas empresas não ficam retidas nos desenvolvedores e são sistematicamente compartilhadas para toda a organização. Dito isto podemos afirmar que se tratando de documentação e ajuda de software, através de uma boa base de conhecimento, é possível tornar a dependência dos usuários menor e dar mais autonomia aos mesmos, uma vez que o conhecimento sobre o software é de fácil acesso e está compreensível para ele, desta forma afirmando a tamanha importância que a documentação de software possui na gestão do conhecimento de uma empresa nos dias de hoje referente aos sistemas que são desenvolvidos também é importante citar que para novos usuários, a documentação de um sistema é importante para facilitar a adaptação ao uso do sistema pelo mesmo.

Os testes de software têm-se revelado cada vez mais importantes no âmbito de desenvolvimento de software para as empresas e o seu progresso tem seguido no sentido de cada vez mais ser uma tarefa independente e imparcial em relação a tarefa de programação de um software devido a sua tamanha importância. Sendo assim considera-se que fica reforçado a importância dos testes no desenvolvimento de um sistema, pois por meio da utilização deles se torna mais rápido a detecção dos erros ou anomalias existentes em um software, consequentemente poupando recursos às empresas. Consonante a isto através dos conceitos de testes aprofundados neste artigo, fica comprovado o quão importante é a documentação dentro do seio da equipe, pois se ter tudo documentado pode ser uma ferramenta de salvação quando alguma coisa não ocorre como o planejado.

Como implicação de trabalhos de pesquisa realizados no referido conceito bibliográfico, o presente artigo possui algumas limitações especificas, a principal delas a ser citada é a de que o tema abordado não costuma ser aprofundado de maneira tão direta quanto a que foi buscada ser apresentada neste artigo, desta forma isto acaba por trazer como resultado das buscas efetuadas poucos estudos sobre o assunto em específico normalmente sendo abordados de uma maneira indireta muitas vezes, e além disto a pesquisa somente foi realizada na base de dados do Google acadêmico e da Scielo tornando assim ainda mais escassa a obtenção de complementações, ideias e os respectivos pensamentos existentes sobre o assunto conforme a opinião de outros autores.

Por fim, como indicação de trabalhos futuros a serem realizados, sugere-se a realização de buscas para um maior aprofundamento acerca dos temas aqui abordados, intensificação de pesquisas além das plataformas Scielo e Google acadêmico como por exemplo o portal de periódicos da Capes, a Biblioteca Digital de Teses e Dissertações da Universidade de São Paulo e Lume - Repositório

Digital da UFRGS, bem como buscar demonstrar outros impactos da documentação e de outros tipos de testes que estão em alta hoje em dia no processo de desenvolvimento de sistemas, como por exemplo, os testes de software automatizados que não necessitam de tanto tempo e trabalho a serem despendidos em sua realização e execução, e que podem ser muito vantajosos dependendo o contexto empregado. Se faz também de suma importância, a aplicação do modelo framework de maneira empírica, a fim de se evidenciar na pratica as relações entre as dimensões, correlações e seus respectivos constructos existentes.

REFERÊNCIAS

MENDES, Leonardo; CERDEIRAL, Cristina; SANTOS, Gleison. Documentation Technical Debt: A Qualitative Study in a Software Development Organization. In: **Proceedings of the XXXIII Brazilian Symposium on Software Engineering**. 2019. p. 447-451.

KREBS, Gunter D. “Ariane-5G”. Gunter's Space Page. Disponível em: <https://space.skyrocket.de/doc_lau_det/ariane-5g.htm>. Acesso em: 07 de nov. 2022.

LI, Zengyang; AVGERIOU, Paris; LIANG, Peng. A systematic mapping study on technical debt and its management. **Journal of Systems and Software**, v. 101, p. 193-220, 2015.

SEAMAN, Carolyn; GUO, Yuepu. Measuring and monitoring technical debt. In: **Advances in Computers**. Elsevier, 2011. p. 25-46.

SILVA, Victor Machado; JUNIOR, Helvio Jeronimo; TRAVASSOS, Guilherme Horta. A taste of the software industry perception of technical debt and its management in Brazil. **Journal of Software Engineering Research and Development**, v. 7, p. 1: 1, 2019.

EBERT, Christof; BRINKKEMPER, Sjaak. Software product management–An industry evaluation. **Journal of Systems and Software**, v. 95, p. 10-18, 2014.

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.

TRIGO, Cláudia Filipa Ferreira. Impacto na Realização de Testes de Software. **PQDT-Global**, 2017.

CARVALHO, M. F. **Automatização de Testes de Software - Dashboard QMSanalyser**. Coimbra. (2010).

TONSIG, Sérgio Luiz. **Engenharia de software: análise e projeto de sistemas**. Ciência Moderna, 2008.

BARBOSA, E. F. Introdução ao teste de software. 2009.

CRESPO, Adalberto Nobiato et al. Uma metodologia para teste de Software no Contexto da Melhoria de Processo. In: **Anais do III Simpósio Brasileiro de Qualidade de Software**. SBC, 2004. p. 204-218.

CATARINA, R. Os 13 principais tipos de Testes de Software. **targettrust**, 2015.

JUNIOR, Elio Trevisan. Processos de Teste de Software. 2007.

PIZZANI, L.; DA SILVA, R. C.; BELLO, S. F.; HAYASHI, M. C. P. I. A arte da pesquisa bibliográfica na busca do conhecimento. RDBCI: **Revista Digital de Biblioteconomia e Ciência da Informação**, v. 10, n. 2, p. 53-66, 2012.

TOFOLI, E. T. Proposta de um modelo de alinhamento da metodologia seis sigma com o gerenciamento matricial de receitas. Piracicaba: UNIMEP, 2011. Tese (Doutorado em Engenharia de Produção). Universidade Metodista de Piracicaba - Campus Santa Bárbara d'Oeste, 2011.

YIN, R. K. **Estudo de caso** – planejamento e métodos. (2. ed.). Porto Alegre: Bookman. 2001.

SILVA, André Santiago da Fonseca. "Documentação de software: uma análise comparativa entre documentação tradicional e living documentation." ("Portal de Programas de Pós-Graduação (UFRN)") 2020.

YIN, OCHODEK, M.; KOPCZYNSKA, S. Perceived importance of agile requirements engineering practices – A survey. ("A Systematic Literature Review on Factors Impacting Agile Adaptation in ...") **Journal of Systems and Software**, Elsevier Inc., v. 143, p. 29–43, 9 2018. ISSN 01641212.

TALITA, L.; SOUZA, M. D. Universidade Federal do Rio Grande do Norte - Centro de Ciências Exatas e da Terra - Departamento de Informática e Matemática aplicada programa de pós-graduação em Sistemas e Computação - PPgSC **Documentação de Requisitos e Compartilhamento de Conhecimento: Uma .** [S.l.], 2019.

TA CHAU, T.; MAURER, F. Knowledge sharing in agile software teams. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 3075, p. 173–183, 2004. ISSN 16113349.

TENBERGEN, B.; DAUN, M. **Is Requirements-Engineering Research Delivering What It Promised?: A Review of Its Accomplishments and Opportunities After 10 Years.** [S.l.]: IEEE Computer Society, 2019. 6–11 p.

SOMMERVILLE, I. **Engenharia de software**. PEARSON BRASIL, 2011. ISBN 9788579361081.

SOMENSI, Douglas. Gestão do conhecimento: padronização e melhoria de ajuda e documentação de software. 2020.

DAVENPORT, Thomas H. **Conhecimento Empresarial: Como as organizações gerenciam o seu capital intelectual**. 14.ed. Rio de Janeiro: Elsevier, 1998.

CAIÇARA JÚNIOR, Cícero. **Sistemas Integrados de Gestão ERP: uma abordagem gerencial**. 2.ed. Curitiba: Intersaberes, 2015.

SILVA, Sergio Luis da. Informação e competitividade: a contextualização da gestão do conhecimento nos processos organizacionais. **Ciência da Informação**, v. 31, n. 2, 2002.

TAKEUCHI, Hirotaka; NONAKA, Ikujiro. **Gestão do conhecimento**. [s. l.]: Bookman, [s. d.]. ISBN 9788577801916.

TAKEUCHI, Hirotaka; NONAKA, Ikujiro. **Criação do Conhecimento na Empresa: como as empresas geram a dinâmica da inovação**. Rio de Janeiro: Campus,1997.

STATDLOBER, Juliano. **Gestão do conhecimento em serviços de TI: guia prático**. **Ciência da Informação**, 1.ed. Rio de Janeiro: Brasport, 2016.

ELIZA, F., & Lagares, V. **Processo de Teste de Software.** DevMedia, 2012.

PINHEIRO, William Richard Everton. Testes de software: estudo exploratório dos tipos de teste de software. 2022.

ROCHA, Camila. **Estudo da qualidade de software na Metodologia V-model e sua interação com metodologias ágeis (SCRUM).** Faculdade de Tecnologia de São Paulo. 2011.

JUNCKES, Gabriel Dias e MORGADO, Paulo. **Gerência de riscos em desenvolvimento de software**. Universidade do Sul de Santa Catarina. 2013.

CUNHA, Simone Moreira. **Engenharia de Software: uma abordagem à fase de testes**. Monografia de Especialização. Universidade Federal de Minas Gerais, MG. 2010.

CAMPOS, Fábio Martinho. **Teste de desempenho: Conceitos, Objetivos e Aplicação - Parte 1. linhadecódigo.** 2015.

Janssen, D., & Janssen, C. (2017). **Mobile Application Testing.** Disponível em: <https://www.techopedia.com/definition/30672/mobile-application-testing>. Acesso em 13 de nov. 2022.

PORTAL BAYMETRICS. **Introdução aos testes de segurança: um guia prático para startups.** Disponível em: <https://www.baymetrics.com.br/introducao-aos-testes-de-seguranca-um-guia-pratico-para-startups/>. Acesso em 15 de nov. 2022.