As fases, técnicas e ferramentas do Design Thinking podem motivar o desenvolvedor de software a identificar e tratar de forma mais eficaz os requisitos de Segurança?

Mariana Borges de Sampaio
[1] *Universidade de Brasília, UnB, Brasília, Distrito Federal, Brasil*
180046926@aluno.unb.br

Keywords: Design thinking, técnicas, fases, método, sistema de software, requisitos de segurança, requisitos de software, requisitos funcionais ,requisitos não funcionais, partes interessadas, stakeholders, métodos ágeis, recursos, desenvolvimento, equipe, tecnologia, TI.

Abstract: This article aims to demonstrate the use of design thinking during the development of a software system. in the software system the problems that are found in a way that the interested parties also understand, that is, without necessarily taking the problem to the code, programming level. This article comes as an approach to define design thinking, its phases, techniques and tools that are used and how this method influences the security requirements of the software system.

1 INTRODUÇÃO

Ao desenvolver um sistema de software podem ser utilizados diversos métodos para que sejam resolvidos os problemas que são encontrados ao longo do desenvolvimento do sistema de software. No contexto deste artigo será abordada a utilização do design thinking, de forma que possa ser estabelecido como seu uso auxilia para definir e especificar o requisito de segurança do sistema de software, sendo este um dos requisitos de software do sistema de software. Tendo como finalidade de que o artigo aborde todas as definições necessárias no contexto deste artigo, serão abordadas as definições sobre o que é design thinking, quais são as suas fases, suas técnicas e as ferramentas que são utilizadas. Também será definido o que são os requisitos de um sistema de software e com maior especificação o que são os requisitos de segurança visto que esse será o requisito analisado com maior ênfase nesse artigo. O design thinking é um método que permite utilizar uma abordagem que envolve as partes interessadas no sistema de software que será desenvolvido a passarem por um momento para combinar a empatia e o contexto do projeto. Dessa forma, todos os envolvidos estariam cientes do que é necessário para o desenvolvimento do projeto. Para isso, o design thinking segue um processo de fases e técnicas que são realizadas de maneira iterativa. O objetivo desse artigo é analisar se as fases, técnicas e ferramentas do Design Thinking que podem motivar o desenvolvedor a identificar e tratar de forma mais eficaz os requisitos de segurança. A partir dessa análise pode se afirmar ou não se esses resultados auxiliam nos requisitos de segurança(Anke Buhl, 2019)(Sommerville, 2011).

2 CONTEXTO

Para compreender este artigo de forma completa é necessário ter algumas definições em mente, visto que ao decorrer desse artigo esses termos serão abordados continuamente. Os termos e definições que serão abordados são os seguintes:

- Sistema de software;
- Partes interessadas;
- Requisitos do sistema de software;
- Requisitos de segurança;
- Design Thinking;

O sistema de software é definido como o sistema para qual o software é primordial importância para as partes interessadas no sistema(IEEE, 2022). Sendo assim, o sistema de software é o produto que será desenvolvido pelo desenvolvedor de software e esse sistema de software tem como objetivo sanar alguma necessidade das partes interessadas no sistema de software.

As partes interessadas no sistema de software são todos aqueles que estão envolvidos no desenvolvimento do sistema de software e têm algum interesse seja de forma direta ou indireta. Dentre as partes interessadas que podem ser citadas como exemplo, tem-se(IEEE, 2022):

- Desenvolvedor;
- Organização interessada no sistema de software;
- Usuário final.

Tendo essas definições é possível entrar nos requisitos de sistema de software. Os requisitos de um sistema referem-se as descrições que o sistema deve fazer, os serviços oferecem e as restrições do seu funcionamento. Esses requisitos devem refletir as necessidades que as partes interessadas têm sobre o sistema de software. O sistema de software deve suprir uma necessidade do usuário final, sendo assim o sistema de software pode ter diversas finalidades. Como exemplo de finalidade, o sistema de software pode realizar o monitoramento de consumo de energia, pode realizar a solicitação de um pedido ou até mesmo pode encontrar informações, notícias da atualidade. O processo que permite descobrir, identificar, documentar e verificar esses serviços é a engenharia de requisitos(Sommerville, 2011). O termo requisito não é utilizado de forma consiste na indústria de software. O termo requisito em alguns casos, é utilizado como uma declaração em que é descrito o que o sistema de software oferece. Quando isso ocorre, o requisito é dividido em dois tipos(Sommerville, 2011):

- Requisitos de usuário;
- Requisitos de sistema.

Os requisitos de usuário referem-se a declarações que descrevem através de linguagem natural ou através de diagramas, o que o sistema de software oferece ao usuário final. Também são fornecidos aos usuários quais as restrições do sistema de software e como este sistema de software deve se comportar através das restrições(Sommerville, 2011).

Já os requisitos de sistema referem-se as descrições que são mais detalhadas das funções, sendo assim, são descritos quais os serviços e quais as restrições operacionais do sistema de software. Esse documento de requisitos do sistema deve definir com exatidão o que deve ser implementado. Sendo assim, esse documento deve ser definido entre todas as partes interessadas no sistema de software(Sommerville, 2011).

No entanto, neste artigo a definição de requisitos que será utilizada refere-se a visão tradicional e que é seguida e descrita no (Sommerville, 2011).

No (Sommerville, 2011) os requisitos são divididos em dois tipos:

- Requisitos funcionais;
- Requisitos não funcionais.

Os requisitos funcionais são declarações de serviços que o sistema deve fornecer sobre como o sistema deve reagir diante de uma entrada específica e como o sistema deve se comportar em diferentes situações. Esse comportamento está relacionado aos diferentes cenários que o sistema de software possui. Além disso, os requisitos funcionais também podem declarar o que o sistema não deve fazer(Sommerville, 2011).

Os requisitos não funcionais são os serviços ou funções que são oferecidos pelo sistema de software. Também podem são incluídas as restrições no sistema de software, sendo essas restrições de timing, restrições no processo de desenvolvimento e restrições que são impostas pelas normas. Os requisitos não funcionais aplicam-se ao sistema como um todo e não apenas a um comportamento no sistema de software, como é o que acontece com os requisitos funcionais(Sommerville, 2011).

Existem vários requisitos não funcionais, como exemplo tem-se os seguintes(Sommerville, 2011):

- Requisitos de produto;
  - Requisitos de eficiência;
  - Requisitos de confiança;
  - Requisitos de proteção.
- Requisitos organizacionais;
  - Requisitos ambientais;
  - Requisitos operacionais;
  - Requisitos de desenvolvimento.
- Requisitos externos;
  - Requisitos reguladores;
  - Requisitos éticos;
  - Requisitos legais;
  - Requisitos de segurança/proteção.

Para especificar os requisitos não funcionais, existem métricas que podem ser utilizadas para analisar cada propriedade(Sommerville, 2011):

- Velocidade;
  - Transações processadas/segundo;
  - Tempo de resposta de usuário/evento;
  - Tempo de atualização da tela.
- Tamanho;
  - Megabytes;
  - Número de chips de memória ROM.
- Facilidade de uso;
  - Tempo de treinamento;

  - Número de frames de ajuda.
- Confiabilidade;
  - Tempo médio para falha;
  - Probabilidade de indisponibilidade;
  - Taxa de ocorrência de falhas;
  - Disponibilidade.
- Robustez;
  - Tempo de reinício após falha;
  - Percentual de eventos que causam falhas;
  - Probabilidade de corrupção de dados em caso de falha.
- Portabilidade;
  - Percentual de declarações dependentes do sistema-alvo;
  - Número de sistema-alvo.

Com isso, pode-se notar que as definições podem ser realizas de forma especificas, mas nem sempre são fáceis de identificar no momento do desenvolvimento do sistema de software. Visto que os requisitos de sistema não apenas especificam os serviços, mas também deve detalhar, especificar os serviços do sistema de software(Sommerville, 2011).

Ter essa definição bem clara é necessário pois será analisado especificamente um requisito não funcional, que é o requisito de segurança e como ele se relaciona com o design thinking. O requisito de segurança pode ser relacionado ao acesso que um usuário tem no sistema de software, por exemplo , o seu acesso através de senha. Esse requisito é visto com frequência em diversas aplicações, como acesso ao aplicativo de banco, e-mail, entre outras formas de acesso. Com isso, existem formas de realizar a especificação de segurança do sistema de software seguindo as seguintes especificações(Sommerville, 2011):

- Identificação de riscos;
- Análise de riscos;
- Decomposição de riscos;
- Redução de riscos.

O processo de identificação de risco consiste na identificação de perigos que identificam os riscos que podem ameaçar os sistemas.

O processo de análise de riscos devem classificar os riscos que foram identificados anteriormente e devem classificar quais os riscos mais perigosos e/ou mais prováveis. Esses riscos mais prováveis ou mais perigosos devem ser priorizados em relação aos demais.

O processo de decomposição de riscos pretende descobrir os eventos que podem gerar algum risco ao sistema de software. Na especificação de segurança esse processo também pode ser visto na análise de riscos.

Por último, o processo de redução de riscos é baseado no resultado da análise de riscos e conduz a identificação de requisitos de segurança. Estes podem estar preocupados em garantir que os riscos não surjam ou que não conduzam a algum comportamento inesperado do sistema de software.

Com isso, passa-se para a definição do método do design thinking. Esse método tem como objetivo fornecer soluções para os potenciais problemas que podem ocorrer durante o desenvolvimento de um sistema de software, a fim de fornecer melhores produtos e serviços além de aumentar a produtividade e possíveis melhorias para o sistema de software. A ideia por trás do design thinking é conseguir acompanhar como ocorre o progresso durante o desenvolvimento e como podem ser descobertas novas ideias e inovações que podem agregar o sistema de software. O design thinking é um método interativo entre as partes interessadas no desenvolvimento de software e é um método centrado em obter informações que sejam relevantes para o usuário. Por isso esse é um método que necessita de interação, colaboração e requer abordagens práticas a fim de encontrar soluções apropriadas para o problema que está sendo analisado. O processo do design thinking é abordado em três fases(Anke Buhl, 2019)(Martins et al., 2019):

- Imersão;
- Ideação;
- Prototipação.

A imersão analisa o problema que foi encontrado no ao longo do desenvolvimento do sistema de software e busca entendê-lo. De forma que seja possível compreender o problema. A imersão pode ser dividida em duas etapas. A etapa preliminar tem como objetivo definir qual o objetivo do projeto e quais são as limitações do projeto. Também são identificados os usuários e demais autores chaves que devem ser considerados. A segunda etapa é a de profundidade. Nessa etapa é realizada uma elaboração de um plano de pesquisa que deve ser alinhado a pesquisa inicial do problema. Nessa etapa devem ser listados perfis de usuários e autores chaves do estudo, visando que os perfis e os autores listados podem auxiliar no contexto do projeto. A comunicação com os usuários pode ser realizada através de algumas técnicas, dentre essas técnicas tem-se as entrevistas, sessões entre outras formas. A finalidade de utilizar essas técnicas como forma de comunicação com os usuários é para auxiliar os usuários no entendimento do contexto que o sistema de software está enquadrado, visto que esse

sistema pode ser um produto ou pode oferecer um serviço(Martins et al., 2019).

A ideação é a fase que permite o surgimento de novas ideias para o sistema de software através de ferramentas e de pessoas que estimulem a criatividade dentre as pessoas que estão relacionadas com o desenvolvimento do sistema de software. A ideação é o momento de interação que tem como finalidade através de formas que envolvam mais as partes interessadas a gerar soluções que são relevantes dentro do contexto do sistema de software. A fase de ideação inicia com um brainstorming em relação a um assunto específico e de acordo com que as ideias surgem o assunto é discutido, documentado e validado. Para realizar essa fase de ideação são necessárias reuniões com as partes interessadas a fim de obter as soluções pertinentes para o sistema de software(Martins et al., 2019).

A fase de prototipagem permite auxiliar na validação das ideias que foram levantadas para serem implementas no sistema de software. Essa fase tem como objetivo identificar os pontos fortes e fracos da ideia que está sendo avaliada além de identificar novos caminhos para o protótipo do sistema de software. Com isso, pode ser identificado que nessa fase antes de realizar a implementação e depois avaliar é realizado o processo inverso. Pois apenas com as soluções estando em comum acordo entre as partes interessadas, levando em consideração a necessidade do usuário e com as soluções bem definidas é que é realizada a implementação.

Além das fases que foram apresentadas do design thinking, existem diversas técnicas que podem ser aplicadas no momento de realizar a elicitação de requisitos do sistema de software. Tendo como referência as fases que foram apresentadas anteriormente, cada fase do design thinking possui fontes de requisitos que são focadas durante o desenvolvimento do sistema de software(Souza et al., 2020).

No (Souza et al., 2020) foram listadas uma sequência de novas técnicas que podem ser utilizadas no design thinking para determinar os requisitos funcionais e não funcionais do sistema de software. As técnicas que foram listadas têm como embasamento que as fases que são seguidas são as de inspiração, ideação e implementação. Dentre as técnicas listadas, tem-se as seguintes:

- Arqueologia Comportamental;
- Plano de Serviço;
- Bodystorming;
- Brainstorming;
- Canvas do Modelo de negócios;
- Cartões de Insight;
- Diagrama de Afinidade;
- Entrevistas;
- Etnografia Rápida;
- Voe na parede;
- Desenho em grupo;
- Mapa Cognitivo;
- Mapa Conceitual;
- Mapa da Jornada do Cliente;
- Mapa das partes interessadas;
- Mapa mental;
- Matriz de Motivação;
- Matriz de Pontos de Contato;
- Personas;
- Pesquisa exploratória;
- Prototipagem;
- Questionários;
- Storyboard;
- Narrativas;
- Experimente você mesmo.

As técnicas apresentadas correspondem a alguns exemplos de como podem ser utilizadas durante o design thinking. Com isso, é permitido que essas técnicas sejam implementadas e facilitem a comunicação entre a equipe. Através dessas técnicas é possível responder alguns questionamentos que são levantados no momento da elicitação de requisitos. Sendo assim, as técnicas que são implementadas no design thinking têm intenção de tornar mais fluído o conteúdo que é apresentado.

Por fim, como o design thinking trabalha de acordo com o que é esperado pelas práticas de elicitação de requisitos, conforme tem-se a evolução do desenvolvimento do sistema de software é possível obter o uso de uma metodologia consistente em nível de documentação e de gerenciamento de equipes. Essa forma como o design thinking trabalha é um dos principais focos de uma metodologia ágil, visto que permite esse envolvimento entre as partes interessadas. Por fim, as ideias que possuem uma melhor classificação analisando os pontos fortes e fracos é que é de fato implementada no sistema de software(Martins et al., 2019).

3 CONFIGURAÇÃO DE ESTUDO

Para realizar o estudo apresentado neste artigo, o que foi identificado logo de início foi a necessidade de realizar a especificação de todos os termos

e todo o contexto necessário para fazer com que a leitura fosse fluída e dinâmica. Tendo essas definições claras a metodologia foi de identificar artigos que relacionassem os temas analisados que são o design thinking e os requisitos de segurança.

Com isso, o que tem-se como principal objetivo é identificar se a utilização do método do design thinking torna mais eficaz os requisitos de segurança. Para definir isso é necessário estabelecer se as fases, as técnicas e as ferramentas de fato auxiliam e caso auxiliem como essas atividades são eficazes no próprio processo. Também deve ser determinado como essas fases foram avaliadas para determinar a eficiência ao utilizar o design thinking. Com isso, para este artigo foram estabelecidas as seguintes perguntas para pesquisa:

1. RQ1: Existem preocupações sobre a definição sobre os requisitos não funcionais no sistema de software?
2. RQ2: O uso de recursos juntamente com metodologia ágil auxilia na definição dos requisitos não funcionais?
3. RQ3: O design thinking permite tratar de forma mais eficaz os requisitos de segurança?

Para isso, foram identificadas fontes que permitissem apresentar as definições dos principais temas que eram necessários para entender do que o artigo se trata. Também foram identificados artigos que permitissem responder esses questionamentos que foram elencados anteriormente.

4 DISCUSSÃO

Diante do que foi abordado até o momento nota-se que o design thinking pode ser inserido dentro de um desenvolvimento de projeto a fim de permitir uma interação entre as partes interessadas. Com isso, foi visto que em (Martins et al., 2019) é abordado que o uso desse método poderia ser realizado durante as sprints que ocorrem durante o desenvolvimento do projeto. Geralmente as sprints ocorrem entre o time que desenvolve o sistema de software, nesse caso, um possível usuário que entende sobre o problema que o sistema de software queira sanar, é convidado a participar da sprint. Dessa forma, ele daria o feedback mais rápido para a equipe que desenvolve o sistema de software, permitindo a integração entre todos os interessados no sistema de software. As sprints são reuniões que ocorrem durante a semana a fim de que ocorra a interação entre o time que está realizando o desenvolvimento do sistema de software. Como esse time pode ser multidisciplinar e envolver diversas áreas essa interação permite gerar novas ideias, permite um momento que integra a equipe. As sprints fazem parte da metodologia ágil chamada de scrum. Essa metodologia é amplamente utilizada nas empresas visando essa interação entre as equipes.

O que pode ser notado é que com a interação entre as equipes as soluções de problemas podem ser levantadas dentro da sprint e todos os envolvidos podem participar. Supondo que a sprint que seja realizada é feita pelo time de tecnologia da empresa, e que nesse time tenham pessoas de diversos times que fazem parte do projeto. Para isso, e para demonstrar esse exemplo, serão definidos que os times que estão desenvolvendo o projeto são os times:

- Dados;
- Desenvolvimento;
- UX/UI;
- Gestão de projeto.

Tendo esses times como exemplo, pode-se supor que problema que precisa ser resolvido por um determinado time, como exemplo o time de desenvolvimento, esse problema pode estar relacionado a uma necessidade que o time de dados já aborda. Entre diversas outras possibilidades de interação entre os times.

Tendo esse embasamento, pode-se notar que possíveis problemas que sejam relacionados aos requisitos não funcionais, ou especificamente relacionado ao requisito de segurança pode ser abordado durante as sprints.

5 VALIDAÇÃO

No (Martins et al., 2019) são abordados alguns questionamentos que são respondidos ao longo do artigo em questão. Dentre um dos seus questionamentos foi abordado o seguinte questionamento:

1. Quais são os desafios enfrentados na elicitação de requisitos de software ágil projetos de desenvolvimento e quais são as técnicas de design thinking utilizadas?

Ao responder esse questionamento foram levantas três atividades que são diretamente relacionadas com a elicitação de requisitos, foram eles:

- Estimativa de custos e cronograma;
- Atenção aos requisitos não funcionais;
- Participação dos clientes.

Dentre essas atividades, a atividade sobre a atenção aos requisitos não funcionais pode ser aplicada como uma das respostas ao questionamento que foi realizado pela RQ1 neste artigo. O (Martins et al., 2019) demonstra que muitas vezes ao desenvolver um sistema de software a equipe está mais voltada a desenvolver a funcionalidade e não se atenta aos requisitos não funcionais. Esses requisitos não funcionais por vezes não estão bem definidos e são ignorados em prol da entrega da funcionalidade.E nesses requisitos não funcionais são encontradas partes técnicas que são essenciais para o desenvolvimento de software, dentre as partes técnicas tem-se:

- Escalabilidade;
- Manutenção;
- Portabilidade;
- Segurança;
- Desempenho.

Seguindo os estudos, focando em responder ao questionamento levantado na RQ2, foi encontrado no artigo (Pereira et al., 2021) que demonstra quais as dificuldades que são encontradas no momento de realizar a elicitação dos requisitos e como o design thinking é visto como uma forma de superar esses desafios. Nesse artigo foram abordadas as dificuldades que foram encontradas e que poderiam ser superadas com a implementação de algum método para melhorar o desenvolvimento do sistema de software.

Tem-se que a elicitação de requisitos é a atividade inicial para compreender a fim de determinar as necessidades que as partes interessadas necessitam no sistema de software. Sendo assim, na elicitação de requisitos são definidas as principais características que o sistema de software deve possuir, levantando assim quais as funcionalidades que o sistema de software deve entregar para as partes interessadas. Segundo (Pereira et al., 2021) para definir as características do sistema de software pode ser analisado em relação a três perspectivas o que o sistema precisa:

- Por que o sistema é necessário (análise de contexto);
- Quais os recursos do sistema servirão para satisfazer esse contexto (análise funcional);
- Como o sistema deve ser construído (requisitos não funcionais).

As tarefas para elicitar os requisitos podem variar, mas podem ser categorizadas da seguinte forma:

- Entendimento;
- Domínio;
- Identificando fontes relevantes;
- Analisando as partes interessadas;
- Selecionando técnicas;
- Obtenção de requisitos.

Seguindo o estudo que foi realizado em (Pereira et al., 2021) foram identificados os principais desafios no momento de realizar a elicitação de requisitos, analisando o processo de deve ser realizado para desenvolver o sistema de software, a comunicação entre as partes interessadas e as qualidades dos requisitos. Para o estudo deste artigo serão listadas apenas alguns dos desafios que foram encontrados, segue a lista com alguns dos desafios que foram encontrados (Pereira et al., 2021):

- Orientação de processo:
  - Falta de orientação de processo;
  - Falta de criatividade no processo de elicitação de requisito.
- Comunicação entre as partes interessadas:
  - O conhecimento prático é difícil de ser entendido pelas partes interessadas.
- Qualidade dos requisitos:
  - Requisitos são inventados e não elicitados;
  - Volatilidade dos requisitos.

A falta de orientação de um processo que determine um método que permita elicitar os requisitos é constante. A necessidade de ter um método é necessário pois existem diferentes situações que devem ser lidadas durante o desenvolvimento de um sistema de software. Com um método seria possível determinar se o processo foi feito de forma correta. A falta de criatividade no processo de elicitação também foi destacado pois esse é um momento em que permitiria ter inovação no sistema de software que deve ser desenvolvido. A criatividade, criação e surgimento de novas ideias são características básicas do design thinking(Pereira et al., 2021)(Anke Buhl, 2019).

A comunicação entre as partes interessadas também foi inserida como como um dos desafios pois, no artigo (Pereira et al., 2021) foi levantada a dificuldade que as partes interessadas podem encontrar para conseguir estabelecer quais as suas necessidades e quais os requisitos que são interessantes para o sistema de software diante do ponto de vista das partes interessadas. Além desse desafio na comunicação entre as partes interessadas tem-se a dificuldade que as partes interessadas têm para entender o que os desenvolvedores fazem em nível de código. Deve existir uma comunicação que permita que todas as partes interessadas se comuniquem de forma que com uma linha de comunicação aberta seja possível estabelecer um melhor entendimento do que está sendo desenvolvido para todas as partes(Pereira et al., 2021).

Já a qualidade de requisitos foi observado que um dos desafios é que os requisitos são inventados e não elicitados. Isso pode ser um dos pontos cruciais dentro do desenvolvimento de software. Visto que, a elicitação ela é feita com foco no sistema de software que deve ser desenvolvido e para isso devem ser levantados nesse momento tudo o que as partes interessadas desejam no sistema de software. Por isso o seguinte desafio também é válido para este artigo, a volatilidade de requisitos tem como consequência um retrabalho para os desenvolvedores devido a reestruturação que o código possui para atender a esses novos requisitos que são levantados(Pereira et al., 2021).

Com isso, diante do que foi abordado tem-se que ao permitir a identificação de um problema, de uma falha no sistema de software de forma mais interativa é possível obter respostas mais rápidas e mais simples em relação a evolução do sistema de software. Com essa identificação e com a comunicação entre as partes interessadas no sistema é possível responder a RQ3. Que é justamente determinar que ao utilizar o design thinking é possível tratar de forma mais eficaz os requisitos de segurança. Pode ser notado que em alguns estudos tiveram como tema a necessidade da utilização do design thinking , isso ocorre pois é vista a necessidade de integrar a equipe e permitir uma geração de ideais com pessoas de diferentes áreas, a inovação de muitos aplicativos têm sua origem em conversas sem tons de seriedade e sim de forma mais sutil(Vetterli et al., 2013).

Sendo assim, utilizando recursos como o design thinking é possível determinar de forma exata quais os problemas que são encontrados no sistema de software. No entanto, além de encontrar é possível determinar como esses problemas podem ser resolvidos, não necessariamente obtendo logo de início uma resposta. Mas sim obtendo alternativas de como o problema pode ser tratado no sistema de software. Por isso, o uso de design thinking é útil para equipes de tecnologia, visto que dessa forma é possível determinar e tratar problemas do sistema de software referente aos requisitos tanto funcionais como não funcionais do sistema de software.

Tendo todas essas necessidades levantadas, tem-se que o design thinking vem como um método que permite resolver todos os desafios que foram abordados. Consequentemente validando o que foi colocado como uma das perguntas de pesquisa, que é justamente se o uso da metodologia ágil poderia auxiliar na definição de requisitos não funcionais. Não limitando apenas aos requisitos de segurança, mas sim nos demais tipos de requisitos não funcionais existes no sistema de software que será desenvolvido.

6 CONCLUSÃO

Por fim, durante o estudo pode-se perceber que a utilização de um método, como o design thinking permite a interação entre as partes interessadas do sistema. Permitindo assim, reunir todos aqueles que são interessados no sistema e possuem um ponto de vista e uma necessidade a serem levadas no momento do desenvolvimento do sistema de software que podem auxiliar durante o projeto. A utilização desse recurso permite um entrosamento entre as partes interessadas, algo que é relevante no contexto em que é necessário o usuário para dar seu ponto de vista. Essa iniciativa permite uma comunicação mais clara entre todos os envolvidos o que pode resultar em um ponto de vista válido ao avaliar o sistema e o uso de uma metodologia no desenvolvimento do projeto.

Sendo assim, pode-se perceber que a utilização do design thinking como recurso que ao ser utilizado em conjunto com uma metodologia ágil permite fornecer a equipe uma forma mais empática e até mesmo mais fácil de lidar com problemas mais complexos. A necessidade de permitir uma comunicação entre todos os interessados no sistema de software permite uma integração de equipe mais conectada e por vezes mais engajada. No momento em que a equipe demonstra crescer em prol de um sistema de software é incentivador para todos aqueles que estão desenvolvendo o sistema de software. Além de permitir que problemas simples sejam sanados entre si.

O scrum é uma metodologia ágil que pode ser utilizada em conjunto com o design thinking. As técnicas e as fases do design thinking podem ser utilizadas continuamente nas sprints, que são reuniões do scrum. E com essas técnicas é que pode ser notado a forma mais descontraída para lidar com problemas de grande porte durante o desenvolvimento de um sistema de software.

Diante dos estudos que foram realizados pode ser observado que a utilização de um recurso como o design thinking permite um ambiente mais leve para resolver problemas durante o desenvolvimento do sistema de software. A necessidade de realizar a elicitação de requisitos e de usar recursos que permitam uma comunicação mais clara é de extrema relevância para o contexto do mercado tecnologia. Visto que esse mercado possui um amplo crescimento e uma alternativa de trabalhar home office a aplicação de design thinking durante as reuniões permite a integração da equipe que está alocada em diferentes ambientes.

Sendo possível resolver questões relacionadas as diversas funcionalidades e necessidades que o sistema de software precisa para ter o seu funcionamento cor-

reto. Dentre essas necessidades tem-se os requisitos funcionais e não funcionais. Dentro dos requisitos não funcionais e tratando especificamente do requisito de segurança é possível utilizar o design thinking a fim de determinar o que dele ser levantado nesse requisito para que o sistema funcione corretamente, evitando falhas durante a pequenos acessos, como a autenticação. O uso de recursos facilita a encontrar as possíveis falhas e consequentemente elas podem ser tratadas, impedindo que o sistema dê algum erro, ou apresente alguma falha ao usuário final. E essa é uma das necessidades que se tem em determinar quais as necessidades das partes interessadas, dos usuários para que os desenvolvedores consigam tratar todas as necessidades funcionais e não funcionais do sistema de software.

Como trabalhos futuros diante do que foi estudado e encontrado sobre o design thinking o que é pensado é em estruturas técnicas que possam ser utilizadas no design thinking e implementa-las em uma equipe de tecnologia. As empresas que estão em uma crescente por vezes precisam determinar como gerenciar as equipes, principalmente empresas que utilizam home office continuamente. Tendo esse cenário em mente, a aplicação do design thinking em conjunto com uma metodologia ágil para acompanhar e gerenciar a evolução da equipe é o momento que pode ser trabalhado para dar sequência ao presente artigo. Para isso, seriam seguidos os seguintes passos que seriam divididos em:

- Análise de mercado;
- Desenvolver a metodologia;
- Aplicação da metodologia.

Inicialmente é necessário explorar e analisar o mercado, de forma que seja possível:

- Encontrar a equipe para ser gerenciada;
- Entender as necessidades da equipe.

Para desenvolver a metodologia, seria iniciado um processo para determinar:

- Definir as técnicas a serem implementadas para a equipe;
- Validar as técnicas com a equipe;
- Definir reuniões para acompanhamento da equipe.

Por fim, a aplicação da metodologia. Nessa fase seria o momento de execução e para acompanhar a implementação seria necessário levantar métricas a fim de estimar se a empresa está conseguindo se desenvolver ou não. E conforme essas métricas seria possível readequar a metodologia de acordo com a necessidade da equipe.

Isso seria feito tendo como principal objetivo estabelecer se melhorou ou não a definição dos requisitos não funcionais do sistema , enfocando nos requisitos de segurança. Podendo posteriormente ser inserida ou até mesmo desenvolvida alguma métrica que permita estimar apenas os requisitos de segurança do sistema de software. Como os requisitos não funcionais são requisitos que podem ser analisados por métricas como(Sommerville, 2011):

- Velocidade;
- Tamanho;
- Facilidade de uso;
- Confiabilidade;
- Robustez;
- Portabilidade.

Pode ser escolhida uma das métricas a ser avaliada. Tratando do requisito de segurança pode ser escolhida a métrica de confiabilidade do sistema de software que será desenvolvido pela equipe escolhida. Dessa forma pode ser avaliado através de métricas e de técnicas do design thinking que permita essa interação entre a equipe.

REFERENCES

Anke Buhl, e. (2019). Design thinking for sustainability: Why and how desig.

IEEE (2022). Sevocab:software and systems engineering vocabulary.

Martins, H. F., de Oliveira Junior, A. C., Canedo, E. D., Kosloski, R. A. D., Paldês, R. Á., and Oliveira, E. C. (2019). Design thinking: Challenges for software requirements elicitation. *Inf.*, 10(12):371.

Pereira, L., Parizi, R., Prestes, M., Marczak, S., and Conte, T. (2021). Towards an understanding of benefits and challenges in the use of design thinking in requirements engineering. In Hung, C., Hong, J., Bechini, A., and Song, E., editors, *SAC '21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*, pages 1338–1345. ACM.

Sommerville, I. (2011). *Engenharia de Software*. Pearson Education,São Paulo. 9.ed.

Souza, A. F., Ferreira, B., Valentim, N. M. C., Correa, L., Marczak, S., and Conte, T. (2020). Supporting the teaching of design thinking techniques for requirements elicitation through a recommendation tool. *IET Softw.*, 14(6):693–701.

Vetterli, C., Brenner, W., Uebernickel, F., and Petrie, C. J. (2013). From palaces to yurts: Why requirements engineering needs design thinking. *IEEE Internet Comput.*, 17(2):91–94.