Uma perspectiva macro acerca da análise de requisitos de software

Jonathan R. Silva, Silvio M. Silva, João V. P. Domingos, Gustavo A. Pereira

[1]Universidade Federal do Pampa (UNIPAMPA)
Alegrete – Rio Grande do Sul – Brasil

Resumo. *A análise de requisitos é uma fase-chave para compreender os problemas a serem resolvidos e validar se os requisitos elicitados corroboram para o propósito do software, buscando garantir a qualidade do sistema a ser entregue. A fase em questão do processo de concepção de um software é descrita sob perspectiva geral no presente artigo, que busca elucidar sobre o contexto em que a análise de requisitos surgiu e se desenvolveu, do que ela é constituída, quais são as suas etapas, como e por que aplicá-la, e acerca do impacto em não aplicá-la – ou não aplicá-la de forma eficaz. Demonstra-se também ferramentas que podem contribuir para a realização das etapas do processo e artigos relacionados.*

Palavras-chave: análise de requisitos, engenharia de requisitos, levantamento de requisitos, técnicas de análise de requisitos.

1. Introdução

Um software, como produto normalmente originado a partir de um determinado problema – ou de um conjunto de problemas –, e cujas necessidades são pautadas em uma solução para este(s) problema(s), possui requisitos. Atualmente, não mostra-se prudente, ou igualmente eficaz, conceber um software sem realizar previamente a elicitação e análise de seus requisitos de forma eficiente, como sugere [Sommerville 2011], dado que a presente atividade atua como um guia aos processos seguintes. No entanto, ao atentar-se à perspectiva histórica da análise de requisitos, houveram épocas onde a etapa em questão da engenharia de software não possuía o ênfase de hoje.

Baseado em exemplares de diferentes épocas e autores da literatura voltada à ciência da computação, o artigo "*A Historical Perspective on Requirements*" (em tradução literal, "Uma Perspectiva Histórica dos Requisitos") [Alexander 1997] propõe que, entre 1962 e 1996 houve uma mudança gradual de atenção na área de desenvolvimento de software, que caminhou em direção ao usuário e, principalmente, às especificações de seus problemas e necessidades. A popularização dos computadores, aliada ao expressivo aumento de acessibilidade e interatividade dos sistemas baseados nestes, e o crescimento de complexidade dos softwares foram alguns dos fatores que contribuíram para que essa se tornasse uma disciplina completa da engenharia [Alexander 1997].

A etapa de análise de requisitos de um software trata, sobretudo, do ato de compreender os problemas a serem resolvidos e validar se os requisitos elicitados corroboram para o que fora estabelecido como propósito do software. O Engenheiro de Software, geralmente acompanhado de outros *stakeholders*, exerce a análise requisitos com a finalidade de conduzir o desenvolvimento do projeto e garantir a qualidade do sistema a ser

entregue, pois são os requisitos elicitados que fornecem a referência para validar o produto final. Em um cenário onde o processo não seja concretizado, o projeto poderá divergir de seu objetivo, originando uma entrega que não atende às expectativas do cliente. Neste caso, pode ser necessário retroceder alguns passos, o que resultaria em gastos de recursos além do que fora inicialmente previsto e retrabalho para a equipe de desenvolvimento.

A seção 2, relativa à Definição, elucida os principais tipos de requisitos e os métodos que auxiliam no processo de análise e documentação destes. Na seção 3, apresenta-se um estudo de caso da aplicação da análise de requisitos em um projeto real. A seção 4 demonstra algumas das ferramentas que podem ser utilizadas para a realização do processo de análise de requisitos. A seção 5 descreve as principais vantagens e desvantagens de se aplicar o processo em questão. Por fim, são apresentados Artigos Relacionados (seção 6) e Considerações Finais (seção 7) acerca dos paradigmas que cercam a análise de requisitos.

2. Definição

Esta seção tem como objetivos definir a análise de requisitos, apresentar os tipos de requisitos, elucidar acerca do processo de interação com os *stakeholders*, demonstrar os métodos que auxiliam durante o processo e a documentação de requisitos.

A atividade de análise de requisitos é uma das primeiras a serem abordadas dentro do processo de desenvolvimento de software. Ela lida com a investigação, definição e escopo de melhorias ou criação de sistemas [Nuseibeh 2000]. Portanto, a mesma é uma etapa crucial do processo, já que é nesse momento em que o *Product Manager* e o Engenheiro de Software identificam as necessidades e requisitos de um cliente. É nessa fase que acontecem as primeiras reuniões com os clientes e/ou usuários do software, buscando conhecer as funcionalidades que o sistema deve possuir.

2.1. Tipos de Requisitos

Existem dois tipos de requisitos.

2.1.1. Requisitos Funcionais

Requisitos Funcionais são necessidades funcionais (serviços) que um software deve atender. De forma simplificada, eles descrevem o que o sistema deve fazer, e em alguns caso até o que o sistema não deve fazer [Sommerville 2011].

Exemplos de requisitos funcionais são:

- Um usuário deve ser capaz de realizar seu cadastro;
- O usuário deve ser capaz de efetuar o login;
- O sistema deve apresentar uma oferta diferente ao usuário de acordo com seu perfil;
- O sistema deve conseguir identificar a localização do usuário.

2.1.2. Requisitos Não Funcionais

Requisitos Não Funcionais, são usados para descrever as restrições de serviços ou funções que um sistema oferece [Sommerville 2011]. De forma simplificada, ele foca em outras

questões importantes para a sustentação, segurança do sistema e uma boa experiência com o uso do software.

Tipos de requisitos não funcionais:

- Segurança;
- Aspectos Legais;
- Desempenho;
- Disponibilidade;
- Usabilidade;
- Compatibilidade;
- Confiabilidade.

Exemplos de requisitos não funcionais:

- As pesquisas realizadas na aplicação não devem demorar mais do que 10 segundos para receber a resposta do servidor;
- O sistema deve ser desenvolvido de forma que seja compatível com as versões mais recentes do *Android*;
- As comunicações entre os servidores e *clients* devem ser criptografadas;
- Informações sensíveis, como documentos pessoais, não devem ficar expostas ao público em geral;
- A aplicação deve estar disponível 24 horas por dia, e 7 dias por semana;
- O sistema deve validar todos os dados que o usuário fornecer, a fim de garantir que não sejam executados comandos indesejados a partir do servidor ou *browser*.

2.2. Interpretação do problema/da solução/dos requisitos

"If I had asked people what they wanted, they would have said faster horse."

— Henry Ford

"Se eu perguntasse às pessoas o que elas queriam, elas teriam dito que era um cavalo mais rápido"

— Henry Ford

Ainda no processo de análise, é comum que ocorram erros de definição de requisitos, pois a falta de experiência dos clientes e/ou usuários dificulta a identificação de funcionalidades realmente necessárias. Em sua maioria, eles não possuem de forma concisa quais funcionalidades o software deverá ter e, por isso, deve-se aplicar métodos que contribuam para que os "Analistas" compreendam as funcionalidades e requisitos pretendidos pelo cliente.

Exemplos de métodos para análise de requisitos:

E**ntrevista fechada:** é quando já se possui um conjunto de perguntas pré-definidas que vão direcionar a entrevista, de forma que os analistas consigam obter respostas objetivas e claras.

E**ntrevista aberta:** é uma conversa sem perguntas pré-definidas, onde os analistas falam de forma aberta sobre as necessidades das partes interessadas.

Q**uestionários:** são utilizados geralmente nos casos em que há diversos grupos de usuários e/ou que estão em localizações geográficas diferentes. Nesse caso, são elaboradas perguntas simples e concisas, juntamente com a realização da seleção de usuários

que farão parte da pesquisa. As perguntas podem ser questões abertas ou fechadas, e devem conseguir extrair o máximo de informações sobre as necessidades/requisitos do cliente.

B**rainstorming: ("tempestade de ideias", em tradução literal)** é uma técnica de ideação que visa gerar um grande volume de novas ideias. A técnica se baseia em princípios como foco em quantidade, ausências de críticas e combinação de ideias. Para que seja produtivo deve-se prestar atenção nos seguintes pontos:

- Tenha objetivos claros e certifique-se que todos estejam cientes deles;
- Tenha um líder que irá direcionar a dinâmica;
- Faça em um ambiente favorável, local bem organizado e que incentive a criatividade;
- Evite conflitos, deixe claro aos participantes que essa é uma atividade colaborativa com foco em geração de ideias e não disputa de egos.

P**rototipagem:** uma versão de ideação do sistema a ser desenvolvido, de baixo custo e rápido para ser construído, pode ser usado para validar os requisitos identificados, funcionalidade e interface do usuário. Algumas das ferramentas mais utilizadas nos dias atuais são:

- Adobe XD;
- Fluid;
- Sketch;
- Figma;
- Ziplin.

Em geral, todas são excelentes opções, possuem diversas ferramentas que tornam o processo de construção do protótipo mais simples e rápido. É importante que seja definido o que vai ser validado com o protótipo, se atentar aos objetivos das partes interessadas, para garantir que o mesmo atinja o resultado esperado, no caso a validação ou invalidação do protótipo.

Joint Application Design (JAD): desenvolvida pela IBM, visa acelerar os processos de desenvolvimento de sistemas de informação. Utiliza-se de dinâmicas em grupo acompanhada de planejamento, estruturação e sistematização de reuniões. Com essa metodologia consegue-se incentivar os usuários a formular problemas e criar soluções, despertando assim, o sentimento de participação e envolvimento com o sucesso do produto.

Princípios básicos:

- Dinâmica de grupo;
- Uso de técnicas visuais;
- Manutenção do processo organizado e racional;
- Utilização de documentação padrão.

A técnica é composta por duas etapas:

- Planejamento, extração e especificação de requisitos;
- Projeto de Software.

Cada uma delas é composta por três fases: adaptação, sessão e finalização.

- **Adaptação**: consiste na preparação para a sessão, organização da equipe e definição dos participantes;
- **Sessão**: consiste em um ou mais encontros marcados, entre as partes interessadas e os analistas;
- **Finalização**: etapa onde é realizado a compilação das informações levantadas durante a sessão, e passado para um documento de especificação de requisitos.

Os participantes são compostos por seis posições, são elas:

- Líder da sessão;
- Engenheiro de requisitos;
- Executor;
- Representantes dos usuários;
- Representantes de produtos de software;
- Especialista.

2.3. Documentação e solução

Uma vez que os requisitos tenham sido identificados, pode-se utilizar dos insumos adquiridos durante os passos anteriores, para produzir um documento de requisitos completo e assertivo, assim os arquitetos de software podem seguir com a projeção da solução.

2.3.1. O que é um documento de requisitos?

O documento de requisitos delimita o escopo do conjunto de funcionalidades e descreve os atributos de qualidade que o sistema deve ter. O documento deve ser elaborado de forma concisa, de forma que os requisitos do sistema fiquem claros, e que todas as partes interessadas consigam compreendê-lo [Sommerville 2011].

Principais pontos:

- O documento de requisitos deve ser elaborado pelo engenheiro de software e deve ser analisado e validado pelos *stakeholders*;
- Serve como mecanismo de comunicação para os *stakeholders*, referência para testes, manutenção e evolução do sistema.

O documento de requisitos deve conter:

- Introdução e visão geral do documento;
- Descrição de requisitos funcionais e não funcionais;
- Escopo não contemplado de funcionalidades;
- Documentação de apoio.

É importante salientar que o documento de requisitos é determinante para o sucesso do projeto, ele identifica quais funcionalidades fazem parte do escopo e quais não fazem.

3. Aplicabilidade

A presente seção exemplifica a aplicação da análise de requisitos em um projeto real, por meio da demonstração de um estudo de caso [Ferreira and Souza 2017] onde os autores relatam os processos e os resultados obtidos ao aplicar a análise de requisitos no projeto de um sistema de checagem de equipamentos, desenvolvido para uma empresa de checagem

de equipamentos do ramo têxtil. A empresa em questão atua desde 1976, sendo o software desenvolvido para uma filial instalada no ano de 2002 em Três Lagoas, Mato Grosso do Sul.

O sistema construído tinha como objetivos principais reduzir a impressão de papéis, contribuir à sustentabilidade, economizar recursos, aumentar a eficiência e agilidade de processos da empresa, melhorar a organização de dados e facilitar o acesso e utilização de informações.

Ao decorrer de seu trabalho, [Ferreira and Souza 2017] relatam acerca do impacto da análise de requisitos, aliada às técnicas de modelagem, para a qualidade do produto final, bem como retratam parte do importante processo de compreender as necessidades da empresa e identificar soluções pertinentes aos objetivos propostos.

3.1. Levantamento e análise de requisitos

Levou-se em consideração para a elaboração e análise de requisitos do sistema algumas particularidades da estrutura organizacional da instituição, havendo, portanto, a análise de diferentes setores da empresa.

Durante o processo, foram definidos os eventos do sistema, os atores e os casos de uso do software. Posteriormente, artefatos foram gerados durante a etapa de modelagem, visando especificar como o sistema deveria ser estruturado e como a suas funcionalidades seriam executadas pelos usuários. Para tanto, optou-se pelo uso da linguagem UML (*Unified Modeling Language*) para a diagramação do escopo do projeto. No entanto, o artigo destaca apenas dois diagramas: de classes e casos de uso.

3.1.1. Diagrama de classes

O diagrama de classes mostra um conjunto de classes, interfaces, colaborações e relacionamentos [Sommerville 2011]. São usados para fazer a modelagem estática do software, e comuns de serem encontrados em sistemas modelados que são orientados a objetos.

De acordo com os criadores da UML [Booch et al. 2005], entre os princípios relacionados, estão:

- Associação;
- Agregação;
- Composição;
- Generalização (herança);
- Dependência.

A Figura 1, retirada do artigo, apresenta as classes identificadas para o contexto analisado. Contém a representação das entidades “Usuário”, “Tipo Usuário”, “Empresa”, “Setor”, “Equipamento”, “Tipo Equipamento”, “Marca”, *“Checklist”*, *“Checklist Coletor”* e *“Checklist Impressora”*. Pode-se observar que as classes *“Checklist Coletor”* e *“Checklist Impressora”* possuem relacionamentos de herança.

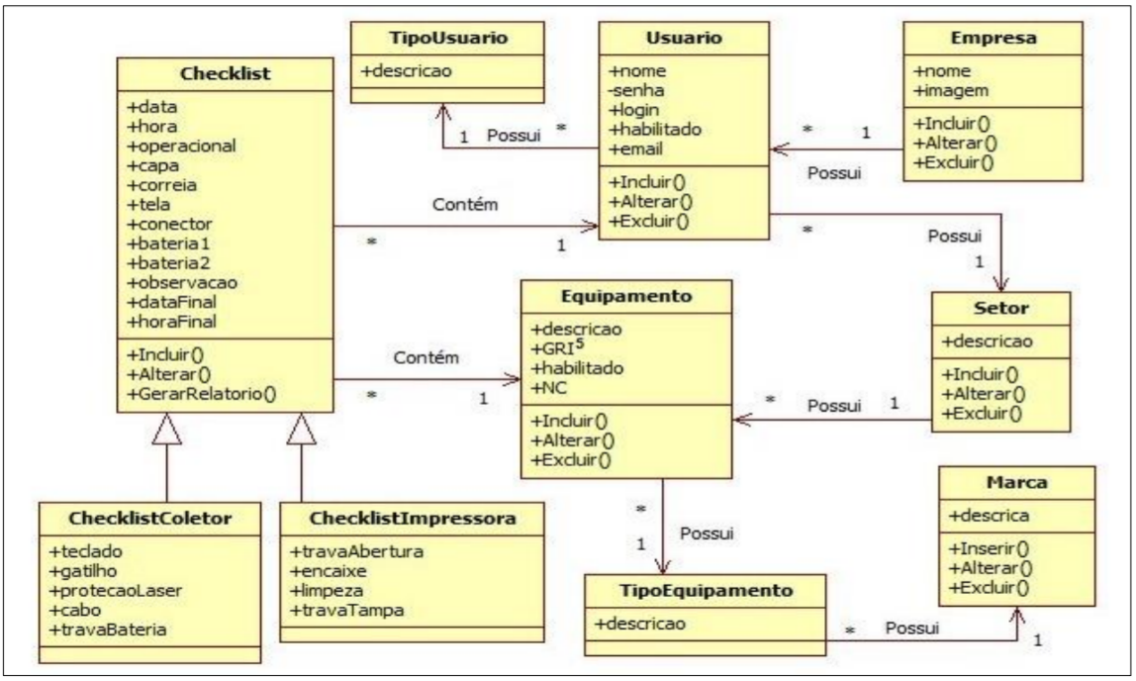

Figure 1. Diagrama de classes

3.1.2. Lista de eventos

A lista de eventos retrata os eventos – ou ocorrências – do sistema, e refere-se às etapas pelas quais os usuários irão passar durante a utilização do software. De forma simplificada, essas etapas também podem ser interpretadas como os "desejos do usuário".

Para o sistema em questão, foram identificados os seguintes eventos:

- Acessar Sistema;
- Cadastrar Empresa;
- Cadastrar Setor;
- Cadastrar Usuário;
- Cadastrar Equipamento;
- Cadastrar Marca;
- Checar Equipamento;
- Gerar Relatório de Checagem.

3.1.3. Atores do sistema

Os atores do sistema são os usuários. Dado o contexto em que os autores estavam, foram identificados três atores diferentes. São eles:

- A**dministrador:** possui acesso total ao sistema e é responsável pelos cadastros;
- A**nalista de qualidade:** possui acesso à funcionalidade de relatórios;
- A**uxiliar de produção:** responsável pela checagem dos equipamentos.

3.1.4. Diagramas de casos de uso

Os casos de uso são uma técnica de descoberta de requisitos e são documentados por um diagrama de casos de uso [Sommerville 2011]. O diagrama em questão é composto por gráficos de atores, por conjuntos de casos existentes no limite de domínio, e pela participação e associação dos atores [Furlan 1998]. Eles são usados para identificar funcionalidades e comportamentos de um sistema.

No artigo [Ferreira and Souza 2017], os autores destacam o diagrama do evento "Checar Equipamento", que fora considerado por eles um dos mais importantes do contexto. Na Figura 2, pode-se observar os roteiros elaborados por eles referentes à ação de realizar ou finalizar a checagem dos equipamentos, onde o ator/usuário sempre terá de informar os dados por completo (para que os mesmos sejam validados). Em casos onde os dados de sucessivas checagens entrem em divergência, o sistema envia um *e-mail* para os responsáveis pelo setor, pois essa é considerada uma ação importante para o adequado controle de recursos da empresa.

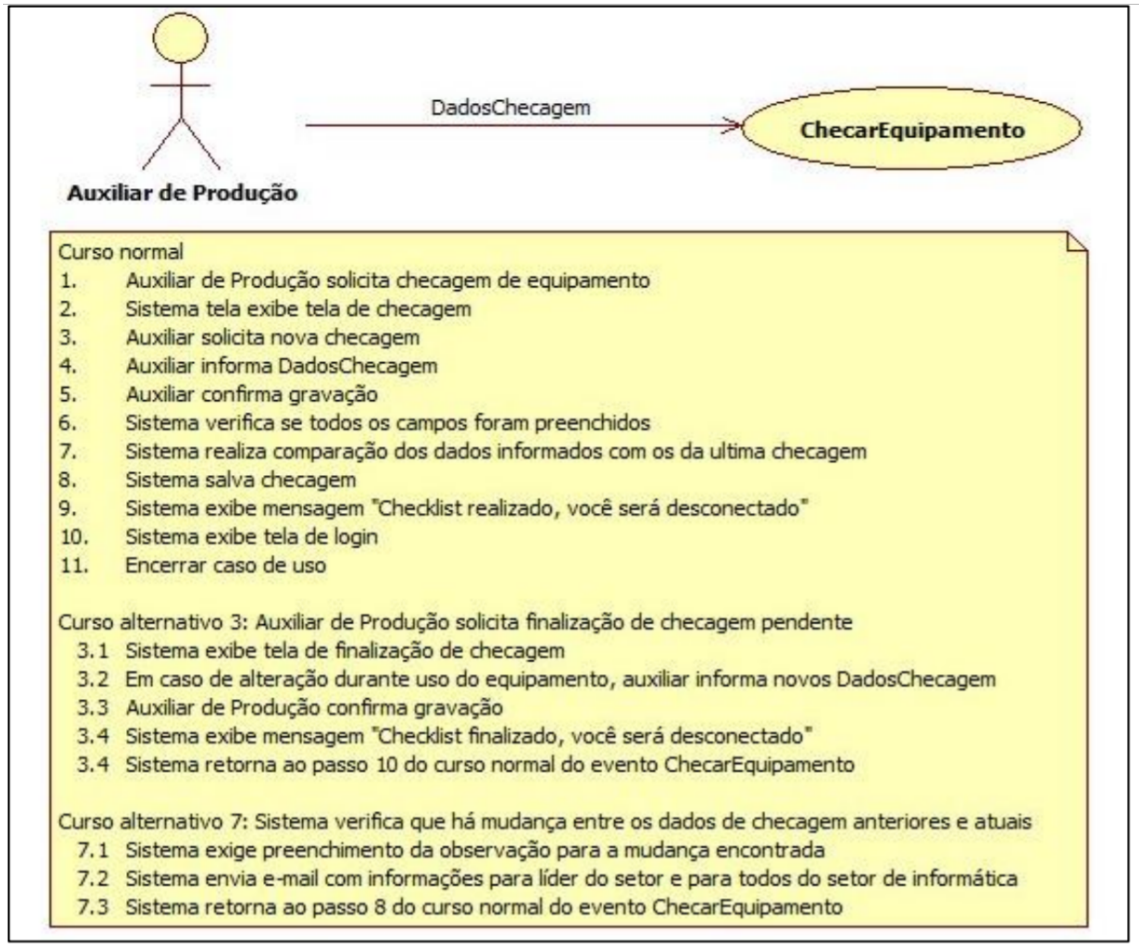

Figure 2. Casos de uso

Para os demais eventos do sistema executou-se o mesmo processo, cada qual possuindo seus diagramas de casos de uso.

3.1.5. Resultado

A fim de alcançar os resultados esperados, foram adotadas abordagens sistemáticas de utilização dos métodos e técnicas descritos para identificar, compreender e solucionar as necessidades da instituição.

Após a implantação do sistema na empresa, pode-se observar progressos em relação à organização dos equipamentos (sobretudo coletores de dados e impressoras portáteis), impacto positivo na redução dos danos causados por mau uso destes equipamentos, e redução significativa na quantidade de impressões em papéis.

Com base nos dados coletados pelos autores até a data de publicação do artigo [Ferreira and Souza 2017] da qual trata a presente seção, constatou-se uma economia média mensal de 21,10 reais em impressões, e estimou-se uma diminuição dos valores gastos com manutenções dos equipamentos e seus acessórios. Outras constatações ficaram pendentes de análise em futuros acompanhamentos.

Em suas considerações finais, os autores ressaltam a importância que a análise de requisitos teve para a identificação dos problemas e compreensão acerca das regras de negócio, fornecendo subsídios importantes para as etapas seguintes do projeto.

4. Ferramentas

Atualmente existe uma grande diversidade de ferramentas que simplificam os processos da análise de requisitos. A presente seção busca demonstrar algumas das principais delas, categorizando-as de acordo com suas finalidades.

4.1. Ferramentas de *Brainstorming*

- **IdeaBoardz**[1]: uma ferramenta para a colaboração em equipe, permitindo que diversos usuários "troquem ideias" e compartilhem informações em tempo real. O IdeaBoardz (Figura 3) também é totalmente gratuito, possui grande interatividade e facilidade de se utilizar, não exigindo nenhum tipo de login ou cadastro. Outra funcionalidade dessa ferramenta é que ela permite colecionar ideias que ficam armazenadas, e que podem ser discutidas e organizadas posteriormente pela equipe.

[1] IdeaBoardz: `http:\\ideaboardz.com`

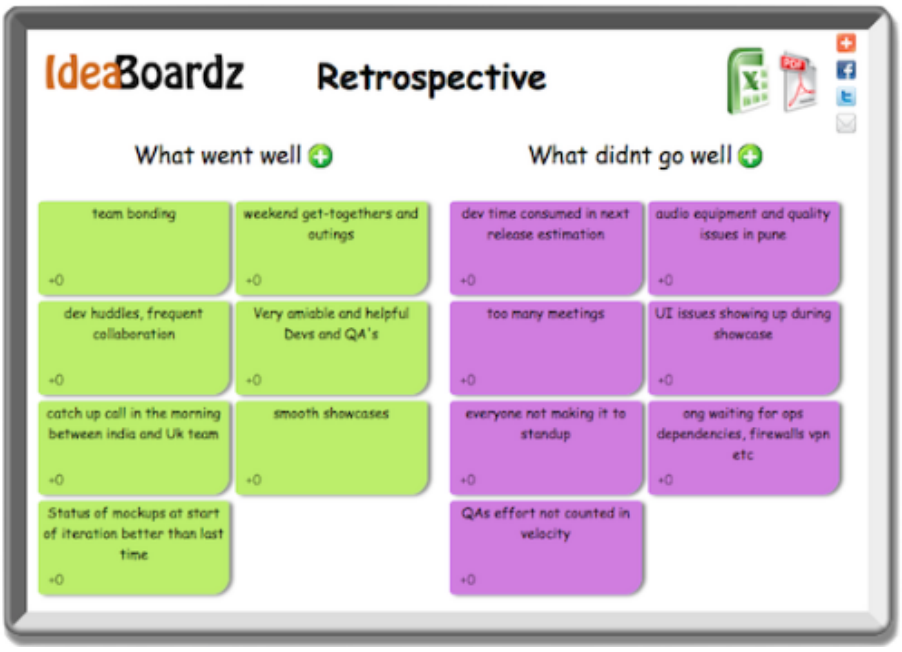

Figure 3. Exemplo de projeto na IdeaBoardz

- **Google Documents** [2]: um conjunto de ferramentas (Figura 4) que permite criar e compartilhar documentos entre os membros de uma equipe. Assim como o IdeaBoardz, esta ferramenta é totalmente gratuita, oferecendo recursos que permitem editar um documento online, ver quem está escrevendo e deixar comentários para que outros membros da equipe possam interagir em tempo real.

Figure 4. Logotipo e ícones das ferramentas do Google Documents

4.2. Ferramentas de Diagramas

- **Bizagi Modeler** [3]: é uma ferramenta (Figura 5) simples de se utilizar, sendo uma opção interessante para usuários que nunca lidaram com modelagem de processos. O Bizagi Modeler é útil para a construção de diagramas e mapas mentais. Para se ter acesso a todas as funcionalidades que a ferramenta pode oferecer (como o compartilhamento de documentos) deve-se realizar um cadastro no site oficial do Bizagi Modeler. A ferramenta também oferece o compartilhamento de projetos entre usuários. Porém, na versão gratuita, algumas funcionalidades são limitadas, como a quantidade de projetos a serem armazenados.

[2]Google Documents: `https://docs.google.com/`
[3]Bizagi Modeler: `https://www.bizagi.com/pt/plataforma/modeler`

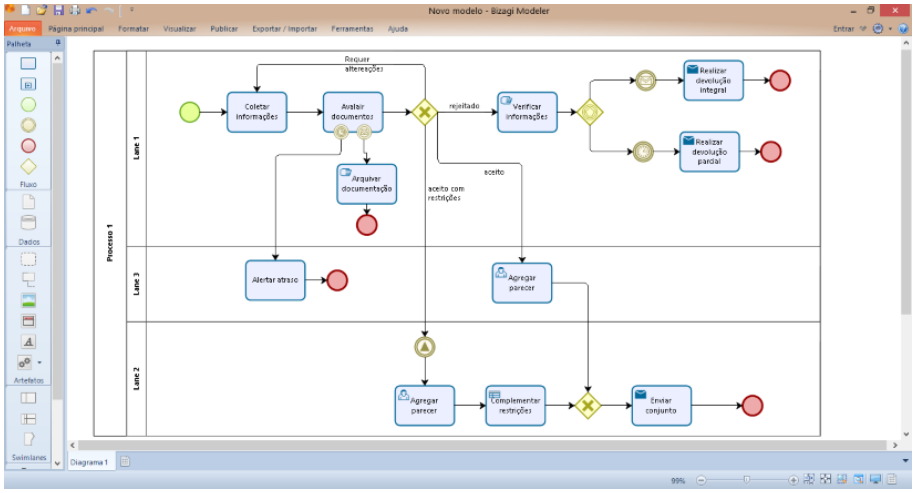

Figure 5. Exemplo de diagrama no Bizagi Modeler

- **BPMN.io** [4]: também conhecida como *Business Process Management Initiative*, esta ferramenta visa a construção de diagramas, com bom nível de facilidade de uso. Como uma grande vantagem dessa ferramenta, a BPMN.io (Figura 6) pode funcionar somente pelo navegador do computador, sem a necessidade da instalação do aplicativo. Outra vantagem é que a ferramenta não exige muitos recursos computacionais para ser executada.

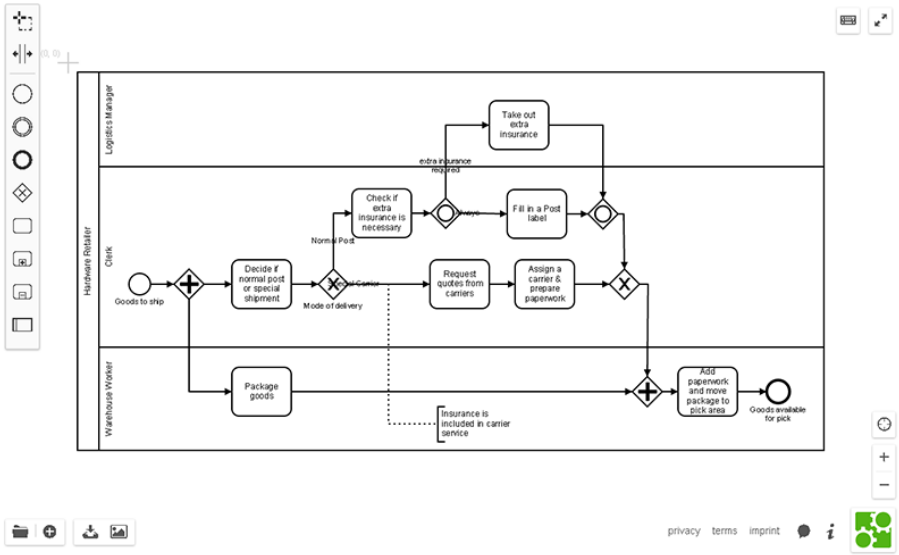

Figure 6. Exemplo de diagrama no BPMO.io

4.3. Ferramentas de Prototipação

- **Adobe XD** [5]: é uma aplicação completa para a criação e validação de interfaces, auxilia os profissionais de UX/UI na criação de protótipos. O Adobe XD (Figura 7) possui tanto um plano gratuito quanto pago. Os acessos diferenciam-se em "uso

[4]BPMN.io: `https://bpmn.io/`
[5]Adobe XD: `https://www.adobe.com/br/products/xd.html`

pessoal” (que possui um limite de compartilhamento de projetos entre os usuários) “uso profissional”, que pode ser usado por equipes ou empresas.

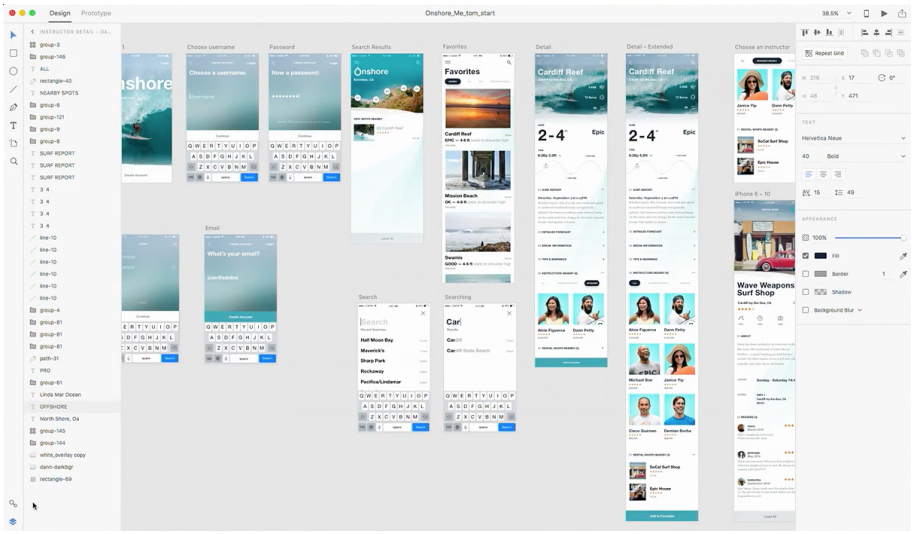

Figure 7. Demonstração da interface do Adobe XD

- **Figma** [6]: assim como no Google Documents, no Figma (Figura 8) é possível que vários usuários interajam em um mesmo arquivo, tornando-a uma ferramenta de design colaborativa. A ferramenta permite escolher quem pode visualizar, editar ou administrar um documento. Uma das vantagens é que a aplicação não requer instalação, pois pode ser utilizada através do navegador.

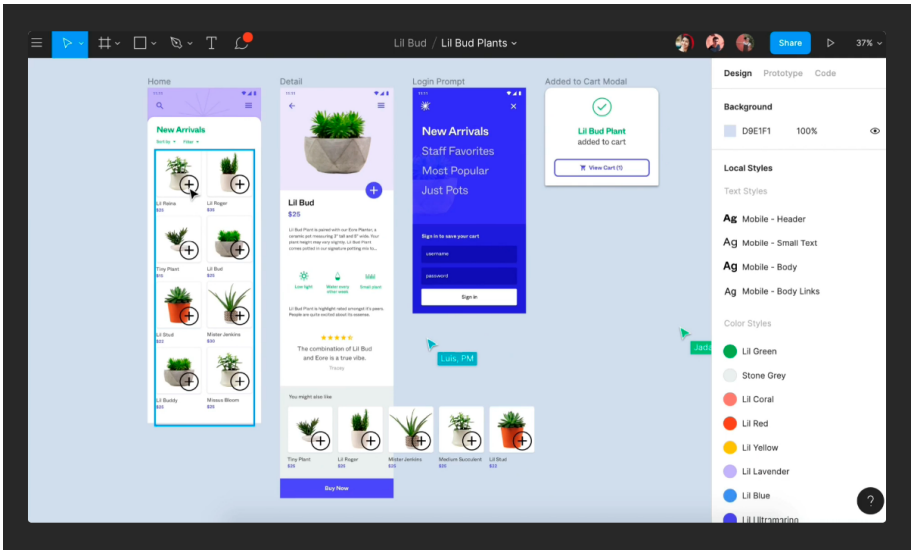

Figure 8. Exemplo da interface do Figma

- **Sketch** [7]: é um software completo para a criação de interfaces. Conta com recursos que permitem também a prototipação e a colaboração entre usuários. Este software (Figura 9) está disponível apenas para sistemas macOS.

[6]Figma: `https://www.figma.com/`
[7]Sketch: `https://www.sketch.com/`

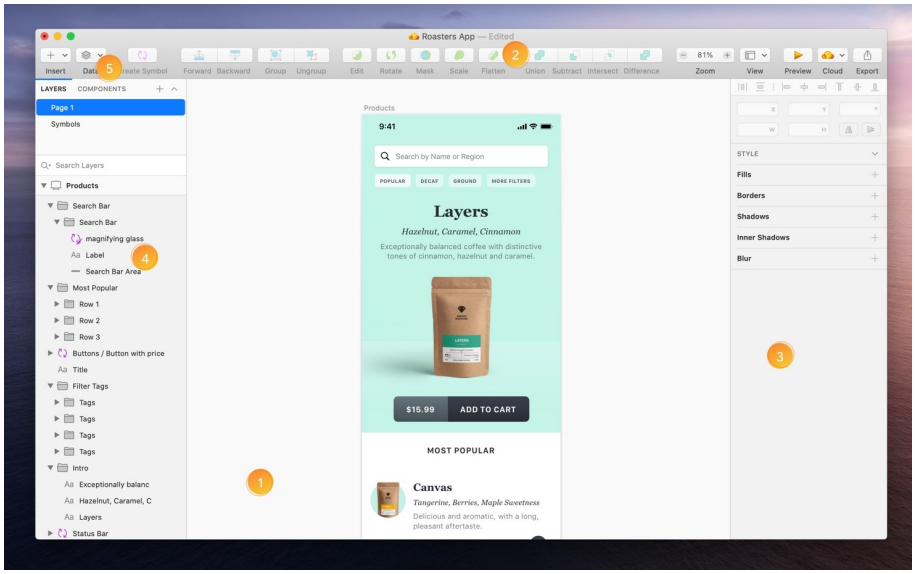

Figure 9. Interface do Sketch

4.4. Ferramentas de Gerenciamento de Tarefas/Projetos

- **Trello** [8]: é uma ferramenta de gestão de tarefas fácil de se utilizar, que tem seu formato de funcionamento baseado em quadros, listas, cartões e checklists. É possível, por exemplo, dividir as tarefas por qualquer critério, anexar imagens e arquivos, fazer comentários, mencionar usuários e adicionar datas de entrega a cada item. Este software (Figura 10) pode ser utilizado para trabalhos em equipes ou de forma individual, onde pode-se determinar quem poderá administrar, editar e visualizar um projeto como forma eficaz de monitoramento e organização. O Trello não exige instalação, pois pode ser acessado através de navegadores. Há, porém, a necessidade de cadastro no site oficial do Trello para se ter acesso às funcionalidades e ao compartilhamento de projetos.

[8]Trello: `https://trello.com/`

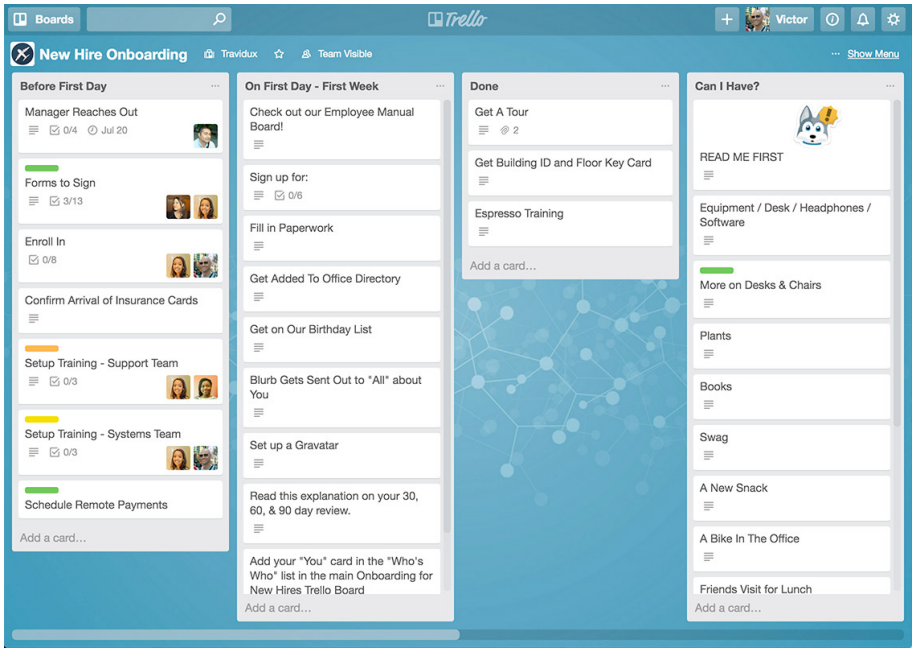

Figure 10. Painel do Trello

- **Artia** [9]: é uma das ferramentas de gestão de projetos mais completas, tendo como funcionalidade o controle financeiro (comparando o custo estimado com o real), um sistema de apontamento de horas de colaboradores, relatórios de desempenho, entre outras. O maior diferencial desta ferramenta é implementar a técnica pomodoro. Como método para facilitar o entendimento do usuário em relação ao trabalho de desenvolvimento, o Artia (Figura 11) possui uma interface bem dinâmica. Este software possui uma versão gratuita e uma paga. Na versão gratuita o espaço para armazenamento de arquivos é limitado, assim como a quantidade de participantes.

[9] Artia: `https://artia.com/`

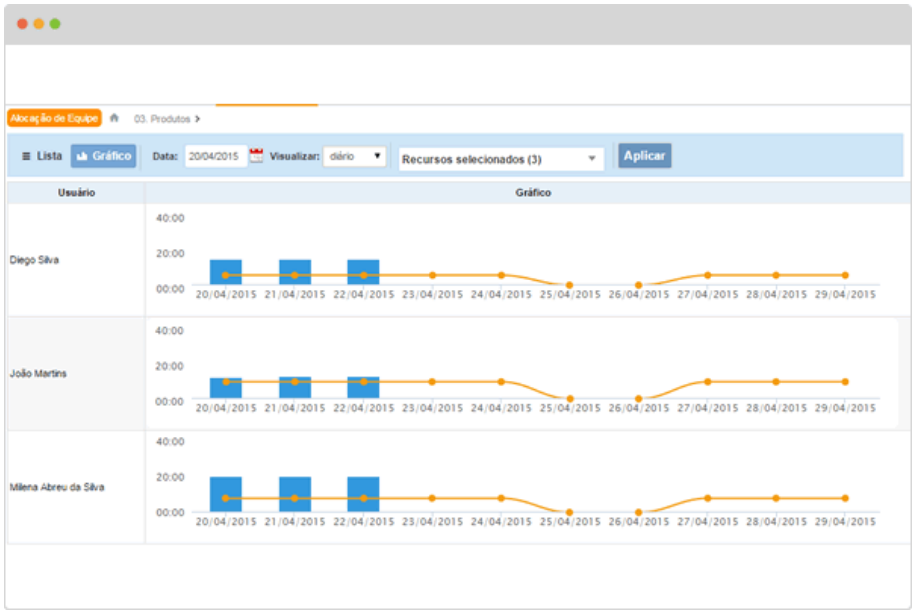

Figure 11. Exemplo de gráfico no Artia

- **Jira** [10]: é uma ferramenta de gerenciamento de projetos que implementa a metodologia ágil Scrum. Possui integrações com ferramentas de controle de versão de código (repositórios online), implementa quadros de Kanban, etc. Este software (Figura 12) é amplamente utilizado para a organização de trabalhos, onde tem-se o monitoramento da equipe e a preocupação com os objetivos do projeto a partir de um quadro de tarefas. O Jira tem planos gratuitos e pagos, onde a diferença apresenta-se principalmente na capacidade de armazenamento dos arquivos e também no compartilhamento de projetos com outros usuários.

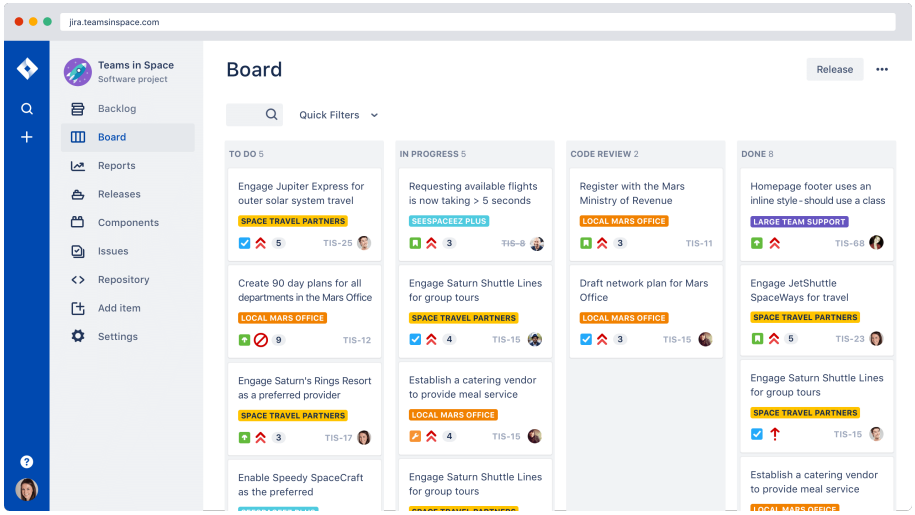

Figure 12. Quadro do Jira

[10]Jira: https://www.atlassian.com/br/software/jira

5. Vantagens e Desvantagens

5.1. Vantagens

"The most important single aspect of software development is to be clear about what you are trying to build"

— Bjarne Stroustrup

"O aspecto mais importante do desenvolvimento de software é ser claro sobre o que você está tentando construir"

— Bjarne Stroustrup

Ao discutir sobre as vantagens da análise de requisitos de modo geral, é indubitável dizer que ela possibilita estabelecer o alinhamento de interesses entre os analistas e os *stakeholders*, estabelece os objetivos em detalhes e identifica os serviços/funcionalidades que o sistema deve fornecer. Permite também a identificação de requisitos não funcionais, tais como: restrições técnicas e legais, segurança dos dados, disponibilidade do sistema, compatibilidade, entre outras.

Como forma mais eficiente de se coletar informações detalhadas dos *stakeholders*, destaca-se o método de entrevistas. Ele possibilita o contato direto com o cliente, tendo assim uma visão mais clara em relação aos requisitos, a realização da validação imediata e até a detecção de problemas antecipadamente, dessa forma minimizando o seu impacto em relação ao tempo. Esse também pode ser considerado um momento no qual a equipe de analistas pode aproximar-se do cliente, proporcionando assim, um maior entendimento sobre o negócio para qual o sistema será projetado.

Por fim, os maiores benefícios em se aplicar a análise de requisitos, são os conhecimentos e artefatos finais gerados pelo processo, com ênfase para o documento de requisitos, que contém os insumos gerados e validados junto aos *stakeholders*, e que será usado para guiar o processo de desenvolvimento da solução.

5.2. Desvantagens

Como "nem tudo são flores", a análise de requisitos também possui desvantagens. Pode-se destacar o tempo e esforços iniciais investidos pela equipe de analistas durante o processo, que engloba diversas etapas a serem realizadas com a finalidade de garantir a compreensão dos requisitos. Conforme [Kendall 2010], o processo de entrevistar pode tornar-se longo e caro, pois os analistas precisam conduzir as etapas com cautela, buscando entender as necessidades reais do cliente.

Durante o processo, pode-se ainda haver a necessidade de retomar passos anteriores, devido a mudanças inesperadas de requisitos, surgimento de novas necessidades e até a identificação de funcionalidades insatisfatórias. Isso ocorre pois é comum acontecerem erros [Sommerville 2011] na identificação de requisitos, atores do sistema e no momento de estabelecer os objetivos junto ao cliente.

É válido pontuar também que a equipe de analistas pode tornar-se uma desvantagem caso a mesma seja formada por profissionais não experientes, impactando a capacidade de lidar com os diferentes perfis de *stakeholders*, e o entendimento de suas necessidades.

6. Trabalhos Relacionados

Esta seção apresenta uma análise comparativa entre este e outros três artigos que abordam o tema de análise de requisitos.

6.1. Visão geral dos artigos complementares

O primeiro artigo a ser citado, escrito por [Falbo 2012], aborda os processos da engenharia de requisitos como um todo e, dentre eles, a análise de requisitos. De modo geral, o artigo apresenta definições, quais são os objetivos e princípios relacionados ao tema. Também são mostradas as técnicas de levantamento de requisitos, dentre elas as entrevistas, questionários, etapa de observação, investigação de documentos e prototipagem. Explica-se brevemente sobre a linguagem de modelagem unificada *(Unified Modeling Language – UML)*, apresentando em seguida técnicas de levantamento de requisitos e um de vários métodos que podem ser usados no levantamento de requisitos funcionais. Ainda discute sobre a especificação de requisitos não funcionais, verificação e validação, gerenciamento de requisitos, e por fim a documentação dos artefatos gerados durante o processo, juntamente com seus benefícios.

O segundo artigo foi escrito por [Chaves 2005] como trabalho final de mestrado, o foco central é direcionado à especificação e documentação de requisitos, onde é apresentado um modelo aplicável à análise da informação utilizando "Casos de Uso". O artigo começa introduzindo o leitor aos desafios e causas que levaram ao surgimento da engenharia de requisitos juntamente com a definição, conceitos e fundamentos. Também é abordado o que são requisitos e seus tipos, processos, técnicas e documentação. Embora o artigo tenha como foco a apresentação do modelo desenvolvido pela autora, o mesmo apresenta uma profundidade significativa em relação ao tema, trazendo outros tópicos importantes como: problemas, qualidade, validação e gerenciamento de requisitos.

O terceiro artigo a ser citado foi escrito por [Figueira 2012] como trabalho de conclusão de curso, onde é apresentado como tema central a análise das técnicas de levantamento de requisitos para o desenvolvimento de software. O autor parte da premissa que a aplicação de uma determinada técnica apropriada facilita a elicitação de requisitos, e o relacionamento com o cliente. Logo, durante o artigo busca-se validar essa hipótese, utilizando-se de pesquisas por questionários para coleta de dados em 18 empresas de sua cidade. Como o foco do artigo é relacionado à análise de técnicas de levantamento de requisitos, o mesmo apresenta várias delas, como: pontos de vista, entrevistas, questionários (que são usados pelo próprio autor a fim de validar suas premissas), casos de uso, entre outras. Entretanto, o artigo não deixa de apresentar uma seção dedicada à introdução da engenharia de requisitos, onde apresenta a elicitação de requisitos, o que são requisitos e seus tipos, análise e negociação, validação, gerenciamento de requisitos e a documentação. Por fim, o autor apresenta os resultados obtidos em sua pesquisa, mostrando que as técnicas onde há a participação de diversos *stakeholders* não são usadas com frequência ou até são desconhecidas por grande parte dos analistas, onde é perceptível a necessidade de um melhor estudo por parte dos analistas em relação às técnicas mais adequadas.

6.2. Análise comparativa entre os artigos

O artigo [Falbo 2012] aborda um tema muito mais amplo em relação à engenharia de requisitos, embora com relação ao assunto "análise de requisitos" o mesmo não se aprofunda

em subtópicos. Apresenta uma boa definição e coerência entre os parágrafos. Aborda assuntos como o gerenciamento de requisitos, verificação e validação, no entanto, observa-se que poucos métodos de levantamento de requisitos são citados. Por outro lado, neste artigo vários desses métodos são abordados com certa profundidade. A partir da publicação é possível explorar outras etapas presentes no vasto assunto da engenharia de requisitos.

O trabalho de [Chaves 2005] tem foco em apresentar o modelo de análise de informações desenvolvido pela autora, porém ainda que o foco não seja abordar o tema da análise de requisitos em si, a autora se aprofunda o suficiente, trazendo para o artigo uma boa introdução dos desafios e causas, problemas relacionados, tipos de requisitos, validação, qualidade e documentação. Pode-se observar que o tema de requisitos e seus tipos, apresenta uma abordagem semelhante com a vista neste artigo, onde se utiliza de exemplos e linguagem simplificada. Porém, há outros temas complementares não abordados neste artigo que podem ser vistos no artigo [Chaves 2005], como: qualidade, validação, gerenciamento de requisitos e o próprio modelo desenvolvido pela autora durante seu mestrado. Vale citar que o artigo é bem embasado e possui ótimas referências bibliográficas.

A publicação [Figueira 2012] aborda a análise de técnicas de levantamento de requisitos, tendo como seu objetivo apresentar os resultados obtidos pelo autor durante seu processo de pesquisa. Vale destacar que durante o processo de pesquisa foi utilizado um dos métodos que são apresentados durante a seção de "Técnicas de levantamento de requisitos", onde pode-se aprender sobre várias outras técnicas, sendo que cada uma delas se aplica bem a um determinado tipo de situação. Muitas técnicas que podem ser vistas neste artigo, também estão presente no artigo do autor, com a mesma simplicidade na explicação e um bom embasamento, porém pode-se observar que este artigo não apresenta todas as técnicas existentes, que por outro lado podem ser conferidas no artigo em questão de [Figueira 2012]. Baseado na proposta do trabalho, pode-se dizer que o autor cumpre muito bem com o prometido, apresentando várias técnicas e as explicando de forma simples. A publicação ainda introduz o leitor ao tema de engenharia de requisitos antes mesmo de abordar o assunto principal. Ao final, são apresentados os resultados obtidos pela pesquisa do autor, expondo conclusões relacionadas ao uso dos métodos pelos analistas.

7. Considerações finais

O presente artigo teve como intuito abordar a análise de requisitos sob uma perspectiva macro, buscando elucidar acerca do contexto em que surgiu e se desenvolveu, do que ela é constituída, de quais são as suas etapas, de como e por qual motivo aplicá-la, e do impacto em não aplicá-la.

Evidenciou-se que, em virtude de fatores como o crescimento de complexidade dos softwares, a exponencial popularização dos computadores, a contribuição das ciências voltadas ao ser humano à área da computação (exemplifica-se, por exemplo, a disciplina de interação humano-computador), houve, ao longo das últimas décadas, uma mudança gradual de atenção na área do desenvolvimento de software que caminhou em direção aos usuários, com ênfase às especificações de seus problemas e necessidades [Alexander 1997]. Foi neste contexto que a análise de requisitos mostrou-se fundamental para a assertividade e mantenabilidade de um projeto de software.

Requisitos podem ser definidos como condições básicas e necessárias para atingir certo propósito. São exigências, demandas ou proposições para o que se almeja e espera enquanto produto. Um software, em paralelo, tende a ser um produto de constituição e manutenção complexas. Portanto, não mostra-se prudente, ou equitativamente eficaz, construir um software sem compreender de forma satisfatória suas finalidades.

Abdicar-se da análise de requisitos em um projeto de software traz consigo riscos imensuráveis, uma vez que, sem a aplicação dos métodos e técnicas presentes nesta etapa, não há garantias mínimas acerca da assertividade dos requisitos elicitados, ou suficiente direcionamento para a etapa de desenvolvimento. Isso pois, a posteriori, os artefatos gerados durante a análise de requisitos atuam como peças-chave para as demais etapas do ciclo de concepção de um software.

"Quando o software é feito de maneira certa, ele exige só uma fração dos recursos humanos para ser criado e mantido."

— Robert C. Martin

Constatou-se que os benefícios obtidos ao aplicar a análise de requisitos superam os malefícios, pois reduzem as chances de haver retrabalho – e consequentemente o gasto exacerbado de recursos – e aumentam a precisão das soluções a serem entregues. Analisar requisitos não parece tratar-se de um ação demasiadamente cautelosa, mas de um processo que visa eficiência, sobretudo em maior prazo.

References

Alexander, I. F. (1997). A historical perspective on requirements. In *Requirenautics Quarterly, October 1997*. Requirements Engineering Specialist Group of the British Computer Society.

Booch, G., Rumbaugh, J., and Jacobson, I. (2005). The unified modeling language user guide. In *The Unified Modeling Language User Guide (2nd Edition)*. Addison-Wesley Professional.

Chaves, F. C. (2005). Especificação e documentação de requisitos. In *Um Modelo Aplicável à Análise da Informação Utilizando "Casos de Uso"*. Universidade Estadual de Campinas.

Falbo, R. A. (2012). Engenharia de requisitos. In *Engenharia de Requisitos*. Universidade Federal do Espírito Santo.

Ferreira, G. S. and Souza, A. P. (2017). Aplicação da engenharia de requisitos e especificação de requisitos na identificação de escopo de sistema. In *Conexão Eletrônica*. Faculdades Integradas de Três Lagoas.

Figueira, A. M. S. (2012). Análise das técnicas de levantamento de requisitos. In *para Desenvolvimento de Software*. Universidade Estadual do Sudoeste da Bahia.

Furlan, J. D. (1998). Modelagem de objetos. In *Modelagem de Objetos através da UML*. MAKRON Books.

Kendall, K. E. (2010). Análise e design de sistemas. In Yagan, S., editor, *Systems analysis and design 8th Edition*. Pearson Education Inc.

Nuseibeh, E. B. (2000). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*. Association for Computing Machinery.

Sommerville, I. (2011). Engenharia de software. In Horton, M., editor, *Software Engineering 9th edition*. Pearson Education Inc.