CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
UNIDADE DE PÓS-GRADUAÇÃO, EXTENSÃO E PESQUISA
MESTRADO PROFISSIONAL EM GESTÃO E TECNOLOGIA
EM SISTEMAS PRODUTIVOS

ELIANA SANTOS DE OLIVEIRA

A UTILIZAÇÃO DOS PRINCÍPIOS DO *LEAN SOFTWARE DEVELOPMENT* NA ENGENHARIA DE REQUISITOS DE SOFTWARE

São Paulo
Março/2015

ELIANA SANTOS DE OLIVEIRA

A UTILIZAÇÃO DOS PRINCÍPIOS DO *LEAN SOFTWARE DEVELOPMENT* NA ENGENHARIA DE REQUISITOS DE SOFTWARE

Dissertação apresentada como exigência parcial para a obtenção do título de Mestre em Gestão e Tecnologia em Sistemas Produtivos do Centro Estadual de Educação Tecnológica Paula Souza, no Programa de Mestrado Profissional em Gestão e Tecnologia em Sistemas Produtivos, sob a orientação da Profa. Dra. Marilia Macorin de Azevedo.

São Paulo
Março/2015

FICHA ELABORADA PELA BIBLIOTECA NELSON ALVES VIANA
FATEC-SP / CEETEPS

Oliveira, Eliana Santos de

O48u A utilização dos princípios do *Lean Software Development* na engenharia de requisitos de software. / Eliana Santos de Oliveira. – São Paulo: CEETPS, 2015.

118 f. : il.

Orientadora: Profa. Dra. Marilia Macorin de Azevedo

Dissertação (Mestrado Profissional em Gestão e Tecnologia em Sistemas Produtivos) – Centro Estadual de Educação Tecnológica Paula Souza, 2015.

1. Lean Software Development. 2. Engenharia de Requisitos. 3. Engenharia de Software. 4. Lean Thinking. I. Azevedo, Marilia Macorin de. II Centro Estadual de Educação Tecnológica Paula Souza. III. Título.

ELIANA SANTOS DE OLIVEIRA

A UTILIZAÇÃO DOS PRINCÍPIOS DO *LEAN SOFTWARE DEVELOPMENT* NA ENGENHARIA DE REQUISITOS DE SOFTWARE

Profa. Dra. Marilia Macorin de Azevedo
Orientadora

Prof. Dr. Mauro de Mesquita Spinola
Membro Interno

Prof. Dr. Antonio César Galhardi
Membro Externo

São Paulo, Março de 2015

À Deus.

À minha família: meu pai Orlando, minha mãe Ernestina
e minhas irmãs tão iguais e tão diferentes Vanessa, Laís e Mary.

.

AGRADECIMENTOS

À minha orientadora, Profa. Dra. Marilia Macorin de Azevedo, pelas inúmeras lições aprendidas nesse período, pela sua atenção e dedicação, por saber ser tão lógica e precisa e, ao mesmo tempo, tão humana. Obrigada!

Aos componentes da banca examinadora, Prof. Dr. Mauro Spindola e Prof. Dr. Antônio Cesar Galhardi pela disponibilidade na leitura do trabalho e pelas valiosas contribuições prestadas.

Aos demais professores do curso de Mestrado, especialmente àqueles com os quais tive o prazer e a honra de participar de suas aulas.

Aos meus amigos de sala, pelas inúmeras lições aprendidas, pelas risadas, pela ajuda mútua.

Aos meus amigos e companheiros da Paróquia São João Batista, que sempre me deram uma palavra de apoio e incentivo, alegrando os momentos mais críticos desta caminhada.

Às empresas que trabalhei nesse período, bem como aos líderes e colegas de trabalho que colaboraram, cada um a sua maneira, para que eu pudesse concluir este trabalho.

Ao meu pai Orlando e à minha mãe Ernestina, por terem me dado a oportunidade de estudar, crescer e evoluir.

Ao meu caro amigo Thiago Carvalho, por suas valiosas contribuições, por todo o apoio e também por se preocupar com cada detalhe e progresso do trabalho.

A meu querido amigo Felipe Teodoro, que me incentivou a ingressar no programa de Mestrado.

A todos que colaboraram direta e indiretamente para a realização deste trabalho.

É muito melhor lançar-se em busca de conquistas grandiosas,
mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito,
que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta,
onde não conhecem nem vitória, nem derrota.

Theodore Roosevelt.

RESUMO

OLIVEIRA, E. S. **A utilização dos princípios do *Lean Software Development* na engenharia de requisitos de software**. 118 f. Dissertação (Mestrado Profissional em Gestão e Tecnologia em Sistemas Produtivos). Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2015.

O presente trabalho tem por objetivo analisar as principais ferramentas da engenharia de requisitos, em conjunto com os princípios do *Lean Software Development* e, a partir daí, estabelecer uma relação entre essas ferramentas e esses princípios do *Lean Software Development*. A fundamentação teórica deste trabalho trata dos processos de desenvolvimento de software passando pela engenharia de software e engenharia de requisitos; é realizado ainda um estudo sobre o *LeanThinking, Lean Production* e, por fim, *Lean Software Development.* Para a pesquisa foi disponibilizado um questionário com 14 perguntas, por meio das quais são apresentadas situações em que as técnicas e os princípios são exibidos de forma conjunta para que especialistas e profissionais da área pudessem expor a sua percepção a respeito da combinação desses conceitos. Os resultados alcançados nesta pesquisa satisfazem os objetivos propostos, apontam que os profissionais possuem uma percepção positiva sobre a utilização dos princípios do *Lean Software Development* na engenharia de requisitos.

Palavras-chave: *Lean Software Development.* Engenharia de Requisitos. Engenharia de Software. *Lean Thinking*.

ABSTRACT

OLIVEIRA, E. S**. The use of the principles of Lean Software Development in engineering software requirements**. 118 f. Dissertação (Mestrado Profissional em Gestão e Tecnologia em Sistemas Produtivos). Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2015.

This study aims to analyze the main tools of the engineering requirements, together with the principles of Lean Software Development e and, from there, establish a relationship between these tools and these principles of Lean Software Development. The theoretical basis of this work deals with software development processes through the software engineering and engineering requirements; is also carried out a study on the Lean Thinking, Lean Production and finally Lean Software Development. For the research it is provided a questionnaire with 14 questions, through which situations are presented in which the techniques and principles are displayed together to experts and professionals could expose their perception of the combination of these concepts. The results achieved in this study meet the proposed objectives, show that professionals have a positive perception of the use of the principles of Lean Software Development in requirements engineering.

Keywords: Lean Software Development. Requirements Engineering. Software Engineering. Lean Thinking.

LISTA DE QUADROS

LISTA DE TABELA

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

EPI	Equipamento de Proteção Individual
IEEE	*Institute of Eletrical and Electronics Engineers*
IIBA	*International Institute of Business Analysis*
JAD	*Join Application Design*
LSD	*Lean Software Development*
MTE	Ministério do Trabalho e Emprego
PDCA	*Plan, Do, Check, Act*
PO	*Product Owner*
PU	Processo Unificado
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
TDD	*Test Driven Development*
UML	*Unified Modelling Language*
WYSIWYG	*What You See Is What You Get*
XP	*Extreme Programming*

SUMÁRIO

1. INTRODUÇÃO

O mercado atualmente apresenta um cenário que exige agilidade, rapidez e assertividade no processo de tomada de decisão, na operacionalização dos processos, bem como no processamento das informações.

Para auxiliar no gerenciamento desse cenário as organizações contam com sistemas informatizados, desenvolvidos para atender às necessidades e expectativas dos usuários e das organizações.

Porém, a partir do momento em que se detecta a necessidade de criação e desenvolvimento de um software até a etapa em que o software é de fato “entregue” para uso, existem diversas fases que compõem o seu processo de desenvolvimento. Dentre estas fases tem-se a engenharia de requisitos.

A engenharia de requisitos é a área responsável por compreender as necessidades e expectativas do cliente e repassar essas necessidades para a equipe de desenvolvimento.

Os profissionais da área de engenharia de requisitos são analistas ou engenheiros de software, especializados no processo de desenvolvimento e concepção do software e que atuam tanto junto ao usuário como junto aos desenvolvedores.

O *Lean Thinking* é uma filosofia cuja principal intenção é produzir com o máximo de lucro e o mínimo de prejuízo. Para tanto faz uso de estrutura enxuta que visa agregar valor aos produtos e serviços do cliente no momento em que ele precisa com o mínimo de desperdício.

A partir dessa filosofia outras modalidades de *Lean* foram criadas como, por exemplo, o *Lean Production* voltado para linhas de produção, o *Lean Service* voltado para a área de serviços e, por fim, o *Lean Software Development* voltado para área de desenvolvimento de software.

O *Lean Software Development* foi desenvolvido por Mary e Tom Poppendieck em 2003, com o intuito de melhorar a qualidade de entrega dos projetos de software desenvolvido pelas equipes.

Esta pesquisa pretende estudar a utilização e a aplicabilidade da engenharia de requisitos com os princípios do *Lean Software Development,* segundo a

percepção dos profissionais da área de engenharia de software, de forma que as práticas de engenharia de requisitos sejam mais efetivas.

1.1. Questão de Pesquisa

Nesta dissertação busca-se resposta à seguinte questão:

Qual a percepção dos profissionais da área de software quanto à aplicação dos princípios do *Lean Software Development* na engenharia de requisitos?

1.2. Objetivos

Para responder a questão foram delineados os objetivos.

1.2.1. Objetivo Geral

Apurar a percepção dos profissionais de TI quanto à utilização e à aplicabilidade dos princípios do *Lean Software Development* na engenharia de requisitos.

1.2.2. Objetivos Específicos

Como objetivos específicos este estudo pretende:

- Analisar a relação entre as ferramentas utilizadas pela engenharia de requisitos identificadas nesse estudo, com os princípios do *Lean Software Development*, a partir da percepção dos profissionais da engenharia de software.
- Verificar se o tempo de experiência dos profissionais de desenvolvimento de software e o porte da empresa que utiliza a engenharia de requisitos em seus processos de desenvolvimento de software exercem influência na utilização desses princípios.

1.3. Justificativa

A justificativa para esta pesquisa se dá pela busca por um modelo de trabalho mais eficaz e assertivo para a engenharia de requisitos. Igualmente, se busca explorar a questão do *Lean Software Development* junto à engenharia de requisitos, uma vez que na literatura não há relatos suficientes a respeito dessa relação que, do ponto de vista deste estudo, pode contribuir positivamente com a engenharia de requisitos, de maneira a apresentar uma forma diferente de pensar os requisitos de software, tendo sempre em mente que mais importante que os recursos tecnológicos a serem utilizados é alcançar a satisfação do cliente e agregar valor ao seu processo. Dessa forma, é possível contribuir para o crescimento do cliente e, consequentemente, para o crescimento do mercado de software.

1.4. Organização do Estudo

O trabalho está estruturado em capítulos.

O capitulo 1, da Introdução, apresenta a questão de pesquisa, os objetivos, a justificativa e a maneira como todo o texto foi estruturado.

O capítulo 2, intitulado Fundamentação Teórica apresenta todo o embasamento teórico do trabalho: - o processo de desenvolvimento de software, seus modelos e variações; - a engenharia de requisitos, seus objetivos, etapas e principais problemas; - a filosofia do *Lean Thinking* bem como suas variações (*Lean Production* e *Lean Software Development),* os conceitos, princípios e ferramentas utilizadas na aplicação da filosofia *Lean* tanto na área de produção quanto na área de software que é o foco deste estudo.

O capítulo 3, da Metodologia, apresenta a classificação da pesquisa, o instrumento de pesquisa (*survey*), o público da pesquisa e como foi escolhida a amostra para responder à *survey*.

O capitulo 4 apresenta os resultados, a análise e a discussão dos resultados alcançados com a *survey*. Tais resultados foram analisados questão por questão e em seguida realizada a análise sob o ponto de vista do porte da empresa e pelo tempo de experiência dos profissionais da área de engenharia.

Ainda no capitulo 4 é realizada uma retrospectiva sobre os assuntos tratados na Fundamentação Teórica de forma a demonstrar para o leitor como o questionário foi construído.

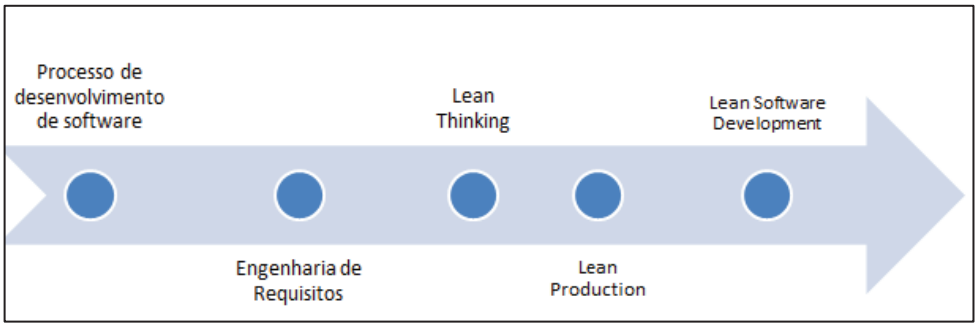
O capitulo 5 apresenta as Considerações Finais a respeito do trabalho e sobre o alcance dos objetivos da pesquisa.

Na sequência seguem as Referências e o Apêndice contendo o questionário utilizado na pesquisa.

2. FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica foi construída por meio de pesquisa bibliográfica sobre os seguintes assuntos: Processo de desenvolvimento de software; Engenharia de requisitos; *Lean Thinking; Lean Production; Lean Software Development,* conforme demonstra o esquema na Figura 1.

Figura 1 − Esquema da Fundamentação Teórica



Fonte: Elaborada pela Autora

2.1. Processo de Desenvolvimento de Software

O processo de desenvolvimento de software possui diversas etapas, cujo resultado final é a construção de um mecanismo capaz de solucionar problemas e otimizar processos. Contudo, o software vai além de um conjunto de instruções técnicas capaz de encontrar alternativas para os problemas cotidianos, ele é um organismo vivo que está em constante processo de evolução levando a uma continua transformação no ambiente onde está inserido.

Para Sommerville (2008, p.42), “o processo de desenvolvimento é um conjunto de atividades que leva à produção de um produto de software”.

Ainda segundo o autor, existe uma diversidade de modelos de processos de desenvolvimento de software, porém, algumas atividades permanecem inalteradas e são comuns a todos eles.

a) Especificação de software: define suas funcionalidades e restrições.

b) Projeto e implementação: o software deve atender à especificação do que deve ser produzido.

c) Validação de software: o software deve ser validado pelo cliente de tal forma que atenda as suas exigências.

d) Evolução do software**:** o software deve evoluir de maneira a acompanhar as novas demandas estabelecidas pelo cliente.

É importante ressaltar, na concepção de Norrmalm (2011), que o processo de desenvolvimento de software é por si só, uma tarefa complexa, independente da complexidade do software a ser desenvolvido, por envolver uma série de variáveis: usuários, cenário organizacional, cultura da empresa. Enfim, todos esses fatores contribuem para que a tarefa de desenvolver software seja complexa.

Nesta seção são descritos os modelos de desenvolvimento de software utilizados com mais frequência. São eles: modelo cascata que é um modelo tradicional baseado em etapas bem definidas; modelo espiral que é um modelo que funciona em espiral com uma etapa diretamente ligada a outra; etapas que podem se repetir ao longo do processo, e ainda o processo unificado que combina a sequência bem determinada do modelo cascata com a liberdade e a flexibilidade do modelo ágil. Por fim é apresentado o conceito de desenvolvimento ágil que se refere a uma abordagem mais flexível e que procura descomplicar o processo de desenvolvimento de software como um todo.

2.1.1. Modelo Cascata

As metodologias tradicionais são baseadas na engenharia tradicional com fases bem definidas, em que é preciso aguardar a finalização de uma fase para o início da próxima; mudanças e alterações de requisitos podem significar problemas ao desenvolvimento do projeto.

O modelo em cascata é uma metodologia tradicional na qual todas as atividades são consideradas fundamentais, ou seja, desde a especificação até a implementação. Este modelo surgiu a partir dos processos mais gerais da engenharia, quando ocorreu um encadeamento das fases; daí surgiu o nome modelo em cascata. (BASSI FILHO, 2008).

O modelo em cascata, de acordo com Sommerville (2008), contempla os seguintes estágios:

a) Análise e definição de requisitos: é a definição das funcionalidades, serviços e restrições do sistema.

b) Projeto de sistema e software: é a fase de definição de uma arquitetura geral do sistema, o momento em que o sistema é dividido em requisitos de hardware e software.

c) Implementação e teste de unidade: nesta etapa o software é implementado e são realizados testes para verificar se os itens estão implementados de acordo com a especificação descrita no início do projeto.

d) Integração e teste de sistema: nesta etapa os itens do sistema são integrados e testados, sendo verificado se o sistema atende às especificações do cliente. Caso os testes sejam bem sucedidos, o sistema está pronto para ser liberado para o cliente.

e) Operação e manutenção: o sistema é disponibilizado em produção para a utilização por parte do cliente. Na maioria dos casos, esta é a fase mais longa; problemas que não foram localizados na fase de testes são detectados e corrigidos.

Para Pressman (2011), o modelo cascata é um modelo que sugere uma abordagem sistemática e sequencial que contempla as seguintes fases:

a) Comunicação: Compreende o início do projeto e o levantamento das necessidades.

b) Planejamento: Fase que inclui a elaboração de estimativas, cronogramas e acompanhamento do projeto.

c) Modelagem: Trata-se da análise do projeto.

d) Construção: Refere-se à codificação e testes.

e) Emprego: Compreende entrega, suporte e feedback.

Os autores Sommerville (2008) e Pressman (2011) possuem visões diferentes a respeito do modelo cascata, diferença que fica evidente na forma como ambos descrevem as etapas do modelo. Para Pressman (2011), o modelo cascata é basicamente um modelo de comunicação e para Sommerville (2008) no modelo as fases estão intimamente ligadas, ou seja, para o início de uma fase a próxima fase deve ser finalizada e devidamente aprovada.

A vantagem desse modelo, segundo Sommerville (2008), é a riqueza de detalhes das documentações produzidas ao longo das interações. A principal desvantagem é a divisão inflexível das etapas dos projetos, que torna a alteração dos requisitos um processo trabalhoso e de alto custo.

Pressman (2011) aponta três desvantagens na utilização do modelo cascata:

a) Projetos reais dificilmente seguem o fluxo sequencial, sendo assim constantes mudanças podem causar confusão à medida que o projeto evolui.

b) Para o cliente é difícil estabelecer logo no início do projeto quais serão todas as suas necessidades de forma detalhada como requer o modelo.

c) O cliente deve ser paciente, pois uma versão inicial do sistema não estará disponível até o final do projeto, dessa forma, se ocorrer algum erro grave na codificação ou nos requisitos, esse erros serão detectados e corrigidos no final do projeto.

Ainda na concepção de Pressman (2011), atualmente o trabalho de desenvolvimento de software possui um ritmo bastante acelerado e uma cadeia de mudanças intermináveis e esse formato não é bem suportado pelo modelo cascata, o que caracteriza que o profissional deve avaliar criteriosamente o contexto de todo o projeto antes de definir qual o modelo será utilizado, seja ele um modelo tradicional ou um modelo ágil.

2.1.2. Modelo Espiral

O modelo espiral foi proposto por Boehm (1988) que, ao invés de representar o desenvolvimento de software como uma sequência linear de atividades, representou como uma espiral. Cada *loop* da espiral é definido como

uma fase do processo; sendo assim, o *loop* mais interno está relacionado com a viabilidade do sistema, o segundo *loop* refere-se à definição de requisitos em seguida ao projeto do sistema e assim por diante. (SOMMERVILLE, 2008).

Já para Pressman (2011), o modelo espiral é considerado um modelo de processo de software evolucionário por conta de sua tendência iterativa. O modelo possui duas características marcantes: a primeira é a abordagem cíclica como objetivo de, a cada iteração, grau de definição e implementação do sistema, aumente, ao mesmo tempo que os riscos; a segunda característica são os pontos de controle do projeto que asseguram o comprometimento dos envolvidos ao longo do projeto.

O modelo permite realizar uma série de entregas do software, iniciando com a entrega da prototipação e a incrementação dessa prototipação até atingir uma versão mais completa. Cada *loop* da espiral está dividido em quatro etapas:

a) Definição dos objetivos: Nesse momento é definido o objetivo do projeto, quais são as suas restrições, riscos e dependências. A partir da definição desses pontos é possível definir quais serão as alternativas adotadas.

b) Avaliação e mitigação dos riscos: Os riscos são analisados criteriosamente e tomadas às providências para que sejam mitigados.

c) Desenvolvimento e Validação: Nessa etapa é escolhido o modelo de desenvolvimento a ser utilizado.

d) Planejamento: O projeto é totalmente revisado e as falhas são corrigidas, a fim de assegurar que todos os requisitos necessários para a próxima etapa do projeto sejam preenchidos. A etapa de planejamento pode ocorrer diversas vezes durante o *loop,* além de auxiliar na tomada de decisão sobre a continuidade ou não do projeto.

Essas etapas não possuem um tempo de duração exato e podem variar de projeto para projeto, entretanto, Pressman (2011) pontua dois problemas:

a) Número incerto de ciclos: Não há como prever com exatidão a quantidade de ciclos necessários para a conclusão do projeto, o que atrapalha o planejamento.

b) Velocidade da evolução: Em modelos evolucionários não há como estabelecer uma velocidade máxima da evolução; se a evolução for rápida demais não há tempo para acomodar todas as alterações e o projeto poderá evoluir para o caos e se a velocidade da evolução for lenta demais, a produtividade poderá ser afetada.

É importante compreender que o objetivo principal dos modelos evolucionários incluindo o modelo espiral é o de produção de software de alta qualidade, o que pode exigir dos gestores uma dose extra de equilíbrio para encontrar na utilização desses modelos a flexibilidade e a alta qualidade que o mercado exige. (PRESSMAN, 2011).

2.1.3. Processo Unificado

O Processo Unificado (PU) é um modelo da *Unified Modeling Language* (UML) e do processo unificado de desenvolvimento de software proposto inicialmente por Ivar Jacobson, Grady Booch e James Rumbaugh (1999). Esses autores discutiram a necessidade de um processo de software voltado para os casos de uso. (PRESSMAN, 2011).

O processo unificado é uma tentativa de aproveitar as melhores características dos modelos tradicionais de desenvolvimento, com parte dos melhores princípios do desenvolvimento ágil. O processo unificado reconhece a importância da comunicação constante com o cliente e dá ênfase ao papel do arquiteto de software como peça importante para o desenrolar do projeto. (PRESSMAN, 2011).

O modelo conta com cinco fases no processo de desenvolvimento.

a) Concepção: Nessa etapa justifica-se o desenvolvimento do software do ponto de vista do cliente. (PRESSMAN, 2011).

b) Elaboração: Fase em que o projeto é detalhado em um nível onde seja possível realizar a construção do software. (PRESSMAN, 2011).

c) Construção: Etapa na qual é construída uma versão totalmente funcional do produto para a apresentação ao cliente. (PRESSMAN, 2011).

d) Transição: Fase onde o produto é disponibilizado em ambiente de homologação e teste. (SOMMERVILLE, 2008).

e) Produção: Etapa em que o software é disponibilizado no ambiente de produção do cliente e, a partir daí, é monitorado pela equipe de desenvolvimento. É nessa etapa que são confeccionados os relatórios de acompanhamento e realizadas as solicitações de mudança. (PRESSMAN, 2011).

O modelo unificado é o inicio da aproximação do modelo tradicional com o conceito ágil, ou seja, uma mistura de ambos os modelos. É possível perceber que o modelo unificado já carrega mais elementos do modelo ágil, o que valoriza a comunicação, a colaboração e a simplificação dos processos.

2.1.4. Conceito de Desenvolvimento Ágil

O processo de desenvolvimento de software é complexo. Além de todas as implicações políticas e comportamentais que acompanham cada projeto, este lida também com variáveis técnicas e de ambiente. Dessa forma, o desenvolvimento de software é considerado uma tarefa difícil, que exige a utilização de metodologias de trabalho capazes de suportar com eficácia todas as etapas do projeto, permitindo que os resultados alcançados sejam satisfatórios. (PRESSMAN, 2011).

A engenharia de software, desde o seu surgimento, conta com um método de trabalho realizado em etapas predefinidas; é realizado todo o levantamento de requisitos, documentação de alto nível, aprovações, desenvolvimento, testes e implantação. Esse método não possui flexibilidade para mudanças e funciona para projetos que não têm grandes alterações em seus requisitos. (FAGUNDES, 2005).

O modelo tradicional não conseguiu suprir as novas demandas e formatos de projetos. Com o aumento da utilização do software nos diversos mercados, aumentou também a exigência por qualidade, o cumprimento de prazo e, principalmente, a habilidade em lidar com mudanças. Isto tem forçado os profissionais da área a buscarem alternativas e formas de adaptação ao novo cenário.

No ano de 2001, Kent Beck e outros 16 profissionais renomados da área de software assinaram o “Manifesto para o Desenvolvimento Ágil de Software”, também

conhecimento simplesmente como “Manifesto Ágil”. Entre esses especialistas havia profissionais que já praticavam algumas metodologias hoje consideradas ágeis como, por exemplo, o Scrum *Extreme Programming* (XP) e outros. Esses profissionais procuravam uma forma de tornar o processo de desenvolvimento de software menos pesado e mais ágil. (BECK, 2001). O Manifesto Ágil é uma declaração de princípios e orientações que norteiam o processo de desenvolvimento de software

Muito mais do que propor novas formas de desenvolver software os idealizadores tinham o desejo de difundir um conceito mais amplo de modelo de trabalho nas organizações, onde as pessoas, a colaboração e a difusão livre do conhecimento fossem itens de fato valorizados nas organizações. Com essa visão foi formulado o manifesto de desenvolvimento ágil. (BECK, 2001).

A primeira parte do Manifesto ilustra, de forma clara, o que os idealizadores queriam enfatizar:

- Indivíduos e interações **valem mais** do que processos e ferramentas.
- Um software funcionando **vale mais** do que uma documentação extensa.
- A colaboração do cliente **vale mais** do que a negociação do contrato.
- Responder às mudanças **vale mais** do que seguir um plano.

O manifesto ágil é composto por doze princípios:

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho não realizado é essencial.
11. As melhores arquiteturas, requisitos e design emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e, então, refina e ajusta seu comportamento de acordo. (BECK, 2001).

É importante ressaltar que as metodologias ágeis não vão contra as metodologias tradicionais, apenas propõem uma desburocratização do processo com o intuito de torná-lo mais ágil.

Em resumo, as metodologias ágeis se diferem das metodologias tradicionais em dois aspectos:

- São adaptativas ao invés de prescritivas: ao seguir rigorosamente a documentação, elas se adaptam às mudanças ao longo do projeto.

- São orientadas a pessoas e não a processos: é valorizada muito mais a interação entre as pessoas do que a execução rigorosa de processos.

Mesmo não indo contra as demais metodologias, o conceito ágil traz mudanças na forma de condução dos projetos de desenvolvimento de software. Entre eles estão:

a) **Tratamento das Mudanças de Requisito:** No modelo cascata, por exemplo, uma mudança de requisito é muito difícil de ser tratada, pois, após a etapa de levantamento de requisitos, praticamente não há mais contato com o cliente. Dessa forma, qualquer alteração implica em retrabalho e aumento do custo total do projeto e isso é visto como um problema. Entretanto, no modelo ágil a mudança é muito bem vista e é atendida prontamente, uma vez que o contato com o cliente é mantido durante todo o projeto.

b) **Contato com o cliente:** Essa é provavelmente a maior mudança do tradicional para o ágil; no modelo tradicional o cliente é acionado apenas no início e no final do processo, já no ágil o cliente acompanha todo o processo, ou seja, o cliente torna-se um parceiro, um membro, uma vez que participa das principais decisões e mudanças de rumo do projeto. Ao final desse processo, o número de surpresas no momento da entrega diminui consideravelmente. Contudo, é importante que a equipe tenha muito cuidado ao conduzir o cliente ao longo desse

processo, pois, o fato do cliente estar mais perto da equipe não significa que ele compreenda em detalhes tudo o que acontece no projeto.

c) **Documentação Enxuta:** O modelo tradicional é conhecido por gerar uma quantidade considerável de documentação bastante detalhada, mas, também, difícil de atualizar e manter, o que acaba, muitas vezes, caindo no esquecimento. O modelo ágil não despreza a documentação do software, apenas sugere que a documentação seja gerada de forma mais resumida e de fácil atualização. Dessa forma é mais fácil à continuidade do trabalho, mesmo que a equipe apresente algum desfalque.

2.2. Engenharia de Requisitos de Software

A engenharia de requisitos está inserida no contexto da engenharia de software e é responsável pelo levantamento, entendimento e documentação dos requisitos junto ao usuário.

A engenharia de requisitos é a responsável por compreender, de maneira precisa, quais são as regras e as necessidades do negócio e propor soluções para esse cenário. Qualquer falha ou má interpretação por parte da área de requisitos pode implicar em problemas graves no futuro, que podem gerar retrabalho para a equipe de desenvolvimento, além de gerar um alto custo financeiro para o projeto como um todo. (ALI; SOH; TORABI, 2005).

Segundo Kotonya e Sommerville (1997), o termo engenharia de requisitos refere-se a todas as atividades realizadas para a descoberta, documentação e manutenção de um conjunto de requisitos para o desenvolvimento de um determinado sistema. Pode-se afirmar, ainda, que é a disciplina que atua na determinação dos objetivos e funções dos requisitos de software.

Pressman (2011, p.127) define a engenharia de requisitos como: “a área de engenharia de software responsável pelo entendimento dos requisitos. Inicia-se na comunicação dos requisitos e vai até a etapa de modelagem”.

Ainda de acordo com o *The Institute of Eletrical and Electronics Engineers* (IEEE, 1984, p.12), o termo engenharia de requisitos pode ser definido como: “A especificação e entendimento dos requisitos de software do cliente para

documentação e posterior desenvolvimento de um sistema de software conforme os requisitos listados".

De acordo com Zave (1997), a engenharia de requisitos é uma área do conhecimento multidisciplinar, que utiliza aspectos sociais e humanos e desempenha um papel importante para a engenharia de software.

A engenharia de requisitos pode ser definida, ainda, a partir de duas visões: a primeira visão trata a engenharia de requisitos como um processo de investigação de problemas para a obtenção de possíveis soluções e a segunda visão trata a engenharia de requisitos como a disciplina que fará a descrição do contexto, as atribuições do sistema, bem como as suas funções. (ENGELSMAN; FRANKEN; IACOB, 2009).

As definições propostas por Sommerville (2008) e Pressman (2011) fundamentam a proposta deste trabalho.

A Engenharia de Requisitos é responsável direta pelo sucesso no desenvolvimento e implementação do sistema. E esse sucesso está diretamente ligado à forma como os requisitos atendem as necessidades dos usuários e do ambiente onde está inserida; a engenharia de requisitos é responsável por compreender e acomodar esses requisitos. (ATLEE, CHENG, 2007).

2.2.1. Classificação de Requisitos

Durante a fase de elicitação e especificação de requisitos é necessário que estes requisitos sejam classificados para facilitar o trabalho de entendimento e documentação das solicitações do cliente. De forma geral, é possível classificar os requisitos em categorias: funcionais e não funcionais; fixos e voláteis.

a) Requisitos Funcionais

Os requisitos funcionais de um sistema descrevem o que o sistema deve fazer; os requisitos não funcionais são aqueles que não estão diretamente relacionados às funções do sistema. (SOMMERVILLE, 2008).

Os requisitos funcionais devem também descrever como o sistema irá reagir perante as entradas e qual será o seu comportamento em determinadas situações (GAVA, 2009). Eles permitem, igualmente, descrever as transformações que serão

realizadas tanto na entrada como na saída de informações do sistema. (SANTANDER, 2002).

De maneira geral, os requisitos funcionais para os usuários são os mais importantes, pois são eles que irão determinar a funcionalidade do sistema. (SILVA, 2006).

b) Requisitos Não Funcionais

Os requisitos não funcionais estão relacionados ao desempenho, confiabilidade, portabilidade, segurança, manutenibilidade, entre outras características que o sistema deve possuir. (SANTANDER, 2002).

Eles podem ser definidos, também, como restrições de tempo ou de processos oferecidos pelo sistema. (SANTANDER, 2002). Ou, ainda, como requisitos que não estão relacionados com o que o sistema irá executar e que apresentam restrições relacionadas ao desenvolvimento, às questões de segurança, usabilidade e desempenho. (MARQUIONI, 2008).

A diferença entre requisitos funcionais e não funcionais está no fato de que os primeiros definem o que o sistema deve fazer e os não funcionais definem as restrições de como o sistema deve fazer. (SANTANDER, 2002).

Os requisitos funcionais e não funcionais são obtidos por intermédio dos usuários solicitantes do sistema. Estes usuários são chamados de *stakeholders* ou partes interessadas e estão diretamente ligados à concepção e ao desenvolvimento do projeto. Podem ser classificados como *stakeholders*: colaboradores, clientes, investidores, comunidade, fornecedores e acionistas, entre outros. Para o projeto de desenvolvimento de software é necessário o correto entendimento dos requisitos de todos os *stakeholders*, de forma a assegurar que o software a ser entregue atenda as necessidades e expectativas das partes interessadas. (SOMMERVILLE, 2008).

A coleta dos requisitos é realizada a partir do contato dos *stakeholders*. Para facilitar a identificação é aconselhável criar uma lista de possíveis *stakeholders* e, a cada comunicação, efetuar o seguinte questionamento, segundo Pressman (2011, p. 136): “Com quem mais você acha que eu deva falar?”. Por meio desta pergunta é possível criar uma lista consistente de *stakeholders*, o que terá impacto direto na captação e análise dos requisitos.

c) Requisitos Fixos e Requisitos Voláteis

O requisito fixo, como o próprio nome diz, trata-se de um requisito estável e geralmente relacionado com os principais negócios da organização, impactando diretamente nos sistemas.

Os requisitos voláteis são mutáveis e sofrem modificações durante todas as etapas do projeto. (SOMMERVILLE, 2008).

O requisito fixo possui um alto grau de estabilidade, sendo improvável que ocorra alguma alteração ao longo do projeto. Os requisitos voláteis têm grandes chances de alteração ao longo do projeto e são considerados bastante instáveis. Por conta dessa instabilidade, os requisitos voláteis ainda são categorizados em:

a) Requisitos instáveis (mutáveis): são alterados quando o ambiente do sistema muda.

b) Requisitos emergentes: são requisitos que não estão claros no momento da especificação, mas que tomam forma durante o processo de especificação e implementação.

c) Requisitos consequentes: são baseados em premissas que, após a verificação, podem ser consideradas incorretas e sofrem alterações a partir da solicitação dos usuários do sistema.

d) Requisitos de compatibilidade: são requisitos baseados em negócios ou outros sistemas. Caso ocorram alterações, os requisitos também são alterados.

A principal vantagem da identificação dos requisitos voláteis é o fato de que isso pode auxiliar a gestão de mudança e a priorização dos requisitos. Para que seja realizada a correta identificação dos requisitos fixos e requisitos voláteis é necessário o auxilio dos *stakeholders*. Isto porque eles conhecem a fundo as regras do negócio e podem afirmar com propriedade quais requisitos irão sofrer variações ao longo do tempo. Assim, a elaboração do documento de requisitos será mais efetiva, além de contribuir de maneira positiva para o planejamento do sistema. (SOMMERVILLE, 2008).

A engenharia de requisitos de software se divide em: elicitação dos requisitos; análise de requisitos, documentação e validação, além da classificação dos requisitos de acordo com as suas características, que são detalhadas nos tópicos que seguem.

2.2.2. O Processo de Engenharia de Requisitos

O processo de engenharia de requisitos compreende todas as etapas necessárias ao entendimento dos requisitos e, a partir daí, gerar a documentação de requisitos do cliente. (SOMMERVILLE, 2008).

A engenharia de requisitos compreende todas as atividades que realizam a descoberta, documentação e manutenção de requisitos para um sistema computacional.

O processo consiste na obtenção, definição, compreensão e análise dos requisitos. Esse processo ocorre continuamente, ou seja, as fases se repetem por diversas vezes até a consolidação e entendimento dos requisitos por parte de todos os *stakeholders.* (PRESSMAN, 2011).

A engenharia de requisitos tem por objetivo aplicar as técnicas de análise dos requisitos com o intuito de fazer a junção entre o desejo de informatização do processo e o projeto de software, buscando fazer com que o software atenda, de maneira satisfatória, as necessidades do processo, (CAMPOS; AZEVEDO JUNIOR, 2008).

As principais etapas do processo de engenharia de requisitos são:

a) **Análise e negociação**: consiste na análise detalhada dos requisitos, de acordo com cada uma das origens de requisitos, de forma a decidir quais serão aceitos. Esta etapa é necessária, pois, invariavelmente, haverá conflitos entre requisitos e suas respectivas origens − pode haver requisitos incompletos ou incompatíveis, tanto do ponto de vista financeiro (orçamento disponível para o desenvolvimento do projeto) como do ponto de vista técnico (alguma limitação técnica que impeça o desenvolvimento do requisito). (SOMMERVILLE, 2008).

A etapa de análise e negociação é responsável ainda por checar se todos os requisitos que foram elicitados são passíveis de implementação, assim como verificar se existem erros relacionados à ambiguidade, inconsistência e incoerência. A análise de impacto e de risco de cada requisito deve ser realizada também nessa etapa. (ATLEE; CHENG, 2007).

A análise de requisitos quando bem efetuada, auxilia também na visualização e priorização dos requisitos. (ATLEE; CHENG, 2007).

b) **Documentação:** após a análise e negociação é preciso modelar e documentar os requisitos em um nível no qual todos os utilizadores do sistema entendam. Geralmente, são utilizadas as linguagens escrita e gráfica para melhor entendimento de todas as partes envolvidas.

c) **Validação:** etapa onde os requisitos são cuidadosamente analisados pelos envolvidos no projeto, para verificar se há coerência entre os requisitos e se há alguma falha ou algum item que não foi descrito. (SOMMERVILLE, 2008).

Além da análise dos envolvidos no projeto é preciso envolver os *stakeholders*; por meio de uma descrição formal as partes interessadas devem participar da validação e verificação dos requisitos com o objetivo de corrigir possíveis falhas, acrescentar novas funcionalidades caso necessário e, principalmente, oficializar todas as informações coletadas até o momento. (ATLEE; CHENG, 2007).

d) **Gerenciamento:** A etapa de gerenciamento de requisitos reúne atividades de todas as outras; inclui-se aqui, também, a evolução do software ao longo do tempo; além disso, o gerenciamento de requisitos é responsável por avaliar a maturidade e a estabilidade dos requisitos. (ATLEE; CHENG, 2007).

2.2.3. Elicitação e Análise de Requisitos

A elicitação e a análise de requisitos compõem o momento em que se inicia o trabalho de obtenção, análise e estudo de viabilidade de requisitos que atendam às necessidades dos *stakeholders.* (SOMMERVILLE, 2008).

Essa fase de elicitação é propícia à identificação do problema, à proposição e estudo de uma possível solução, bem como à determinação de possíveis requisitos para o sistema. (PRESSMAN, 2011).

A elicitação de requisitos faz parte do levantamento de requisitos e busca encontrar quais são os objetivos do sistema e levar em consideração os principais problemas apontados pelos *stakeholders.* (BRONZE et al., 2009).

Essa etapa é o início do trabalho da engenharia de requisitos. Caso a elicitação e análise dos requisitos não ocorra de maneira correta, todo o projeto de software ficará comprometido. Justamente por ser uma etapa de grande importância,

vale destacar alguns problemas que os engenheiros de requisitos enfrentam nessa fase:

a) Frequentemente os *stakeholders* possuem dificuldades em expressar e, até mesmo, em definir quais são suas reais necessidades em relação ao sistema.

b) A diversidade de *stakeholders*: cada *stakeholder* possui uma necessidade específica e uma expectativa em relação ao sistema; gerenciar estas necessidades e expectativas é papel do engenheiro de requisitos.

c) Alterações no ambiente organizacional afetam os requisitos, modificando a sua importância e influenciando a inserção ou retirada de requisitos descritos anteriormente. (SOMMERVILLE, 2008).

Entre as atividades de elicitação e análise de requisitos podem ser citadas:

a) Obtenção de requisitos: etapa em que há a interação entre engenheiros e *stakeholders*, para a coleta de requisitos.

É nessa etapa que se dá o levantamento dos requisitos; essa fase é responsável por identificar e modelar as necessidades do projeto. Nessa fase deve ocorrer a junção entre os interesses da área de negócio e o alinhamento dos *stakeholders* quanto aos objetivos do projeto. (CAMPOS; AZEVEDO JUNIOR, 2008).

b) Classificação e organização dos requisitos: nesta etapa ocorre a organização e categorização dos requisitos.

c) Priorização e negociação dos requisitos: em projetos onde há uma quantidade razoável de *stakeholders* invariavelmente há conflitos entre os requisitos. Sendo assim, é necessária a priorização e a negociação para o estabelecimento da ordem de desenvolvimento desses requisitos.

d) Documentação de requisitos: os requisitos são documentados. A documentação dos requisitos pode acontecer de maneira formal ou informal. (SOMMERVILLE, 2008).

Conforme já citado anteriormente, o processo de elicitação e análise dos requisitos podem ser definidos como uma espiral, onde a cada volta os requisitos são incrementados e melhorados. Cada uma das etapas supramencionadas pode

ser refeita até que os requisitos estejam consistentes com as expectativas dos *stakeholders.* (SOMMERVILLE, 2008).

2.2.4. Técnicas Utilizadas na Elicitação de Requisitos

A elicitação é uma etapa importante no processo de engenharia de requisitos, pois, é nesse momento que os requisitos são identificados e definidos. Isto torna essa fase decisiva, uma vez que a definição incorreta dos requisitos compromete todo o projeto. (SOMMERVILLE, 2008).

Essa etapa compreende atividades relacionadas ao entendimento das metas e objetivos da construção do sistema proposto. A elicitação também envolve a identificação dos requisitos e também dos *stakeholders.* (ATLEE; CHENG, 2007).

Ainda durante a elicitação de requisitos é necessário ter em mente que além da utilização de técnicas para o levantamento dos requisitos outros pontos devem ser levados em consideração. São eles:

a) Assegurar que todos os envolvidos e afetados pelo software foram consultados e estão cientes do andamento do projeto.

b) Auxiliar os usuários a entender suas necessidades e comunicá-las da melhor forma possível.

c) Analisar todos os cenários possíveis de modo a garantir que o sistema funcionará em todos os cenários.

d) Realização de feedback com todos os envolvidos, por meio da revisão de todo o levantamento realizado, a fim de assegurar que todos os pontos foram contemplados. (ATLEE; CHENG, 2007).

Para facilitar o processo de elicitação de requisitos, são utilizadas algumas técnicas de levantamento:

a) *Workshop* de requisitos: consiste em reunir os principais *stakeholders* do projeto (decisores) para uma reunião, que deve ser conduzida de forma focada e em um curto período de tempo. O objetivo dessa reunião é definir os pontos-chave do projeto e melhorar a comunicação interna. A vantagem dessa técnica é que as

decisões são tomadas num período curto de tempo, já que os decisores estão reunidos. (COSTA; ZOUCAS; ALVES, 2012).

O *Workshop* contribui ainda para a redução das divergências entre os *stakeholders* e a melhoria da comunicação. Outro ponto importante dessa técnica é a inclusão do usuário dentro do processo de desenvolvimento. Este envolvimento motiva o usuário e faz com que ele colabore de maneira mais intensa com o projeto. (COSTA; ZOUCAS; ALVES, 2012).

b) Cenário: consiste na elaboração de cenários, incluindo fatos previsíveis inseridos no dia a dia dos usuários. O principal objetivo da utilização de cenários é estimular a reflexão dos usuários sobre situações e problemas que possam ocorrer ao longo do tempo. Essa técnica oferece aos engenheiros de requisitos e aos usuários uma visão mais ampla dos requisitos, permitindo um melhor entendimento e facilitando a identificação dos processos-chave do projeto (GALETTI; SPINOLA, 2005).

Esse procedimento favorece ainda a participação mais ativa do usuário, que pode mostrar ao engenheiro de requisitos como são realizadas suas atividades e quais informações o sistema deve prover para que a atividade possa ser executada. (FRANCETO, 2005).

A técnica permite antecipar situações em que é possível observar a interação humana. Igualmente, é possível, através da técnica, ter uma visão mais ampla da urgência dos problemas a serem resolvidos pelo sistema. A visualização de cenários permite aos participantes perceberem necessidades que seriam notadas apenas após a implantação do sistema em produção. (GOMES; WANDERLEI, 2003).

c) Etnografia: consiste na observação dos usuários em seu local de trabalho, durante a realização de suas tarefas, sem nenhuma interferência do engenheiro de requisitos, que apenas observa e efetua anotações detalhadas sobre a rotina de trabalho dos usuários. A vantagem da etnografia é a observação do cenário em tempo real. (GALETTI; SPINOLA, 2005).

A técnica promove, ainda, a familiarização do engenheiro de requisitos com o ambiente onde o sistema será implantado, bem como facilita a compreensão e o entendimento dos requisitos, uma vez que é possível observar e entender o contexto de cada necessidade dos usuários. (COSTA; ZOUCAS; ALVES, 2012).

A etnografia é uma metodologia que tem origem na Antropologia Social e tem por objetivo enxergar o contexto com o mesmo olhar dos integrantes desse cenário. A partir da etnografia é possível entender os aspectos sociais e culturais do grupo e executar as atividades com as mesmas motivações. (SILVA JUNIOR; CARVALHO; BORGES, 2008).

De acordo com os referidos autores a etnografia oferece diversas vantagens. Entre elas estão:

1. Maior familiarização com a organização.

2. Descoberta de aspectos informais que não estão documentados, que não foram percebidos ou não foram reportados durante as entrevistas.

3. Maior percepção sobre a forma como as pessoas interagem com os sistemas, tecnologias e procedimentos.

4. Mais visibilidade aos problemas relacionados à usabilidade e a sub-utilização de tecnologias já implementadas além da percepção a respeito de atividades que não devem ser automatizadas.

5. Identificação das pessoas frente às dificuldades encontradas e como elas se adaptam frente a essas dificuldades. (SILVA JUNIOR; CARVALHO; BORGES, 2008).

Com o objetivo de melhor atender ao ciclo de desenvolvimento de sistemas, a etnografia foi dividida da seguinte forma:

1. Etnografia Concorrente: São realizados estudos curtos e interativos antes do inicio do desenvolvimento, para que sejam colhidos os requisitos iniciais e a geração dos primeiros protótipos. Em seguida os estudos são repetidos, acompanhados da observação dos usuários com os protótipos até que todos os requisitos sejam confirmados.

2. Etnografia Rápida: Acontece quando não há possibilidade de execução de estudos intensivos e mais demorados, então efetua-se um estudo rápido, de forma a se ter uma visão geral do contexto e realizar o desenho o escopo do projeto de sistema.

3. Etnografia Avaliativa: Tem o objetivo de validar requisitos ou modelos conceituados e verificar se estão de acordo com atividades executadas. (SILVA JUNIOR; CARVALHO; BORGES, 2008).

Além das vantagens listadas anteriormente vale ressaltar que a técnica da etnografia também apresenta algumas desvantagens. Entre elas:

1. Por se tratar de uma técnica invasiva, a presença do observador pode alterar a forma como os observados se comportam, o que pode influenciar diretamente na performance dos usuários e do processo.

2. A utilização dessa técnica pode elevar consideravelmente os custos do projeto, uma vez que é necessário mais de um analista para o mesmo projeto, além de um tempo maior de análise.

3. As informações capturadas através da etnografia são compiladas em formato descritivo e textual; a conversão dessas informações em requisitos pode ser um processo difícil e pouco prático. (SILVA JUNIOR; CARVALHO; BORGES, 2008).

A utilização dessa técnica, assim como qualquer outra técnica de análise de requisitos, está condicionada a características do projeto de software. E, por essa razão, devem ser analisados os prós e os contras antes de iniciar a sua prática.

A etnografia pode também ser combinada com a prototipação para facilitar o entendimento do cenário. Essa técnica fornece uma grande quantidade de detalhes que passariam despercebidos em outras técnicas. Em contrapartida, também permite que oportunidades de melhorias e correções no sistema sejam deixadas de lado, pois, o engenheiro de requisitos e o restante da equipe ficam focados somente no entendimento do processo atual. (GAVA, 2009).

d) Prototipagem (Descoberta por experimentação): consiste na elaboração de uma versão simples do sistema logo no início do projeto, quando o usuário pode manipular e realizar testes, por meio do contato com uma versão que simula as situações reais em que o software deverá atuar. (GAVA, 2009).

Por propiciar ao usuário a possibilidade de realizar testes, é bastante utilizada com o intuito de minimizar os erros nas etapas de elicitação e análise de requisitos e, dessa forma, diminuir custos. (RANGEL, 2003).

A prototipação pode ser classificada ainda em: prototipação de baixa fidelidade e prototipação de alta fidelidade. A prototipação de alta fidelidade é a que oferece ao usuário a experiência de executar tarefas e obter respostas de forma bastante similar ao software original. A prototipação de baixa fidelidade é representada pelos modelos mais rudimentares e de baixo custo, confeccionados em papel, que servem para o usuário tenha uma ideia inicial do sistema. (VAZ, VANDERLEI, BARROS, 2011).

A prototipação para desenvolvimento de software é dividida de acordo com as seguintes abordagens de desenvolvimento:

1. Prototipação Evolucionária: São protótipos desenvolvidos logo no inicio do projeto e são refinados ao longo do período de desenvolvimento, dessa forma o processo de desenvolvimento do software e de criação do protótipo são essencialmente os mesmos. (SOARES, 2003).

2. Prototipação Exploratória: Permite que a equipe de desenvolvimento desenvolva um protótipo claro o suficiente, de modo que o usuário possa interagir e confirmar os requisitos funcionais que já foram elicitados para o projeto. (VAZ; VANDERLEI; BARROS, 2011). É útil ainda para auxiliar na consolidação das ideias dos usuários e ajudar os desenvolvedores a obterem novas ideias e alternativas para o desenvolvimento de software (RANGEL, 2003).

3. Prototipação Experimental: Fornece aos usuários diversas alternativas para o software, de forma a dar para os usuários uma visão de futuro do sistema, no que se refere a questões como performance e demais requisitos técnicos. (RANGEL, 2003).

e) Entrevista: a realização de entrevistas formais e informais com os *stakeholders* é uma prática comum nos projetos de sistema. Estas entrevistas podem ser abertas ou fechadas. As fechadas contêm um roteiro de perguntas predefinidas; as abertas não seguem um roteiro predefinido e o entrevistador explora diversos aspectos do sistema junto ao entrevistado, captando, deste modo, uma quantidade maior de informações. Geralmente, os engenheiros de requisitos mesclam os dois tipos de entrevista a fim de obter melhores resultados — às vezes, entrevistas muito abertas podem resultar em falta de objetividade e entrevistas muito

fechadas podem impedir a visão do todo. Sendo assim, a mescla das duas técnicas apresenta resultados satisfatórios. (SOMMERVILLE, 2008).

A entrevista é uma das técnicas de elicitação de requisitos mais utilizadas, pois, além de fornecer uma grande quantidade de informações em pouco tempo, ainda auxilia na validação dos requisitos. Uma entrevista pode auxiliar a determinar a viabilidade de um projeto, sua real necessidade e o que é determinante para a continuidade do projeto. (FERNÁNDEZ; MORENO; TUBIO, 2013).

Para que a entrevista seja bem sucedida devem ser levadas em consideração quatro dimensões: - o tempo necessário que cada participante possui disponível; - o grau de compartilhamento de objetivos entre entrevistado e entrevistador (ambos devem estar afinados e conscientes dos objetivos da entrevista); - a confiança entre entrevistado e entrevistador (o entrevistador deve confiar nas informações prestadas pelo entrevistado, o entrevistado por sua vez deve ter a segurança de que suas informações serão utilizadas da maneira correta); - durante o processo é importante que sejam realizados esforços para que haja o máximo de cooperação e o mínimo de conflito entre os *stakeholders*. (BORTOLI, 1999).

A entrevista é uma técnica essencialmente de dialogo é muito útil para entender com detalhes a rotina e a forma como o entrevistado compreende e conecta as informações ao seu redor. (BORTOLI, 1999).

f) *Joint Application Design (*JAD) é uma técnica desenvolvida pela IBM do Canadá, a partir de 1977, para o levantamento de requisitos. O principal objetivo do JAD é estabelecer uma reunião estruturada com os principais *stakeholders* do projeto. (BELGAMO; MARTINS, 2009).

A técnica permite o desenvolvimento de sistemas em tempo reduzido, além de ser aderente a diversas metodologias de desenvolvimento. Um dos motivos do sucesso do JAD é o fato de que a técnica coloca na mesma posição os usuários e os profissionais da área de sistemas; todos são igualmente responsáveis pelo sucesso ou fracasso do sistema. (PALUDO et al., 2003).

Para a realização do JAD é necessário estabelecer um líder neutro que será o responsável por conduzir as sessões, elaborar a agenda e os materiais que serão utilizados para a realização de cada sessão, além de assegurar que todas

acontecerão conforme o previsto e que irão atingir os objetivos esperados. (PALUDO et al., 2003).

O JAD possui alguns princípios que devem ser observados durante a sua realização:

1. Dinâmica de grupo: Após reunir o grupo é preciso estimulá-los a interagir uns com os outros, de forma a definir qual será o objetivo e o escopo do projeto. (SANTOS; CORREIA NETO, 2009).

2. Recursos audiovisuais: Permitem aos usuários perceberem o trabalho a ser realizado de modo menos abstrato e mais concreto, além de possibilitar uma visão mais clara do que se quer atingir. É interessante que sejam disponibilizados aos usuários materiais que os permitam transpor suas ideias para o papel e, assim, interagir e participar mais ativamente do processo. (SANTOS; CORREIA NETO, 2009).

3. Processo organizado e racional: O JAD utiliza a análise *top down* que permite traçar, de maneira precisa, quais serão as etapas a serem executadas do início ao fim do projeto; atividades bem definidas auxiliam a manter o foco, além de transmitir segurança para todos os envolvidos. (SANTOS; CORREIA NETO, 2009).

4. Documentação com a abordagem WYSIWYG (*What You See Is What You Get*): Trata-se de um rol de documentos com a seguinte ideia: o que você vê é o que você obtém; uma documentação clara e completa, obtida ao final do processo, que deve ser de comum acordo entre todos os integrantes do projeto. (SANTOS; CORREIA NETO, 2009).

5. Escolher um lugar adequado: O local para a realização das sessões de JAD deve ser tranquilo, agradável para todos os participantes, fora do ambiente de trabalho e com acesso restrito para evitar interrupções. (PALUDO et al., 2003).

Para que as sessões ocorram com sucesso é preciso também selecionar com cuidado os participantes, a fim de obter um JAD de boa qualidade. Contudo, antes de mais nada, é preciso identificar quem serão os participantes:

- Condutor (líder de reunião; mediador): é a pessoa que conduz a reunião, direciona e orienta os participantes a fim de garantir os resultados.

- Analista de Sistemas: é o responsável por dar continuidade aos trabalhos de desenvolvimento do sistema.

- Patrocinador: é a autoridade máxima sobre a área funcional do sistema, é o patrocinador que deve participar em tempo integral das reuniões e prover todos os recursos para o desenvolvimento do sistema.

- Usuários do Sistema: são as pessoas que terão contato direto com o sistema.

- Documentador: responsável por realizar todas as anotações durante as reuniões e auxiliar o líder na elaboração do documento final.

- Demais Participantes: outras pessoas que participam direta ou indiretamente do projeto. (PALUDO et al., 2003).

O JAD é organizado de modo a contemplar quatro sessões:

1. Primeira Reunião: Conduzida pelo patrocinador, define o escopo e os objetivos do projeto; são definidas as datas da sessão de design.

2. Levantamento de Dados e Análises: Etapa executada pelo analista de sistemas que deverá levantar os dados, realizar as análises necessárias e produzir a documentação e fluxos.

3. Planejamento da sessão de design: é realizado entre o analista de sistema e o condutor da reunião; o analista apresenta o material produzido na reunião anterior para que o condutor possa tomar conhecimento, realizar suas críticas e sugestões e, em seguida, preparar a reunião de design.

4. Reunião de Design: a reunião de design é a essência do JAD; nessa reunião o analista de sistemas apresenta em conjunto com o condutor a proposta para execução do projeto. Nessa reunião são discutidos os seguintes pontos: apresentação e definição dos objetivos do sistema, problemas e limitações; análise e definições dos novos processos a partir do novo sistema, bem como as novas interfaces; elaboração do cronograma de execução, aprovação de todos os envolvidos e avaliação da reunião.

Ao final, todas as informações colhidas pelo documentador são recolhidas para a elaboração do documento final que será utilizado ao longo de todo o projeto. (PALUDO et al., 2003).

O documento final é a consolidação de todo o trabalho realizado ao longo das quatro sessões de JAD e representa o compromisso de todos os envolvidos no projeto, uma vez que o documento foi elaborado com o conhecimento e o consentimento de todos. (PALUDO et al., 2003).

g) *Brainstorming:* É uma técnica de geração de ideias bastante utilizada para promover a interação entre um pequeno grupo, a fim de se obter a resolução para um determinado problema. (PINTO, 2007).

Para a aplicação da técnica do *brainstorming* é necessário que o problema seja simples e bem entendido por todos os membros do grupo. Caso o problema tenha uma complexidade maior, é necessário que o problema seja quebrado em porções menores e tratado em diversas sessões de *brainstorming*. (PINTO, 2007).

A técnica possui três fases que são: exposição de abertura, exposição de ideias e fase de escrutínio.

1. Fase de exposição de abertura: o condutor da reunião expõe o problema a ser resolvido bem como os recursos disponíveis, deixando bem claro os objetivos da reunião para todo o grupo.

2. Fase de exposição de ideias: é considerada a etapa de produção efetivamente, onde as ideias são apresentadas e discutidas pelos membros.

3. Fase de escrutínio: é a fase onde se realiza a seleção e documentação das ideias. (PINTO, 2007).

2.2.5. Documentação de Requisitos

A documentação é a etapa em que os requisitos são formalizados e apresentados ao cliente para validação. (PRESSMAN, 2011).

Nessa fase, o engenheiro de requisitos prepara a documentação de maneira que todos os *stakeholders* possam entender. (AUDY; ESPINDOLA; MAJDENBAUM, 2004).

Para a elaboração da especificação de requisitos pode ser utilizada uma ou mais maneiras de representação, entre elas a linguagem formal, semiformal e informal, representações gráficas e simbólicas, para atingir todos os *stakeholders.* (SOMMERVILLE, 2008).

O documento de requisitos deve obedecer algumas premissas. São elas: - especificar o comportamento externo do sistema; - especificar as restrições; - servir como ferramenta de referência para os desenvolvedores; - fornecer respostas aceitáveis para eventos indesejáveis; - e efetuar o registro do ciclo de vida do software. (SOMMERVILLE, 2008).

Ainda sobre o documento de requisitos é importante frisar que o documento deve ser organizado de maneira que evidencie ao desenvolvedor as informações relevantes relacionadas ao planejamento da arquitetura e design do software. O documento deve definir claramente o estado da solução, os atributos, a inter-relação entre os requisitos e a rastreabilidade. Nessa etapa podem ser utilizadas formas de representação gráfica, já consolidadas no mundo de software, que são os casos de uso (representação visual de uma funcionalidade do sistema), que podem ser um aliado importante tanto para a documentação quanto para o entendimento. (BASTANI, 2007).

A documentação dos requisitos tem por objetivo, ainda, a tarefa de unificar todos os requisitos em um único documento, de maneira a facilitar a sua manutenção e rastreabilidade. Embora alguns estudiosos da área defendam o armazenamento desses requisitos em banco de dados − visando facilitar a sua manutenção − comumente eles são guardados em formato de documento. (AUDY; ESPINDOLA; MAJDENBAUM, 2004).

2.2.6. Validação de Requisitos

A etapa de validação de requisitos é importante, pois é a que define se os requisitos descritos mostram, de fato, o sistema que o usuário deseja. A validação auxilia na descoberta de erros e ambiguidades; se a falha nos requisitos for identificada após o desenvolvimento, o custo de correção pode ser muito mais alto. (SOMMERVILLE, 2008).

A validação de requisitos também tem por objetivo localizar defeitos na especificação de requisitos, além de alinhar o entendimento de todos os *stakeholders* a respeito do que será desenvolvido. (FERNANDES; WERNECK; BARBOSA, 2010).

De acordo com Pressman (2011), no momento da validação de requisitos, além de verificar se os requisitos estão alinhados com as expectativas dos *stakeholders*, é importante averiguar alguns itens: - se todos os requisitos estão de acordo com as características globais do projeto; - se todos foram descritos num nível de abstração adequado; - se há algum conflito entre eles; - se é possível implementá-los e testá-los. Estas e outras questões devem ser levantadas no momento da validação.

Para Sommerville (2008), os requisitos podem ser verificados utilizando os seguintes critérios:

1. Verificação de validade: verifica-se se o requisito descrito é válido.

2. Verificação de consistência: verifica-se se há conflito nos requisitos.

3. Verificação de completeza: verifica-se se os requisitos estão completos e possuem todas as funções e restrições descritas pelos *stakeholders*.

4. Verificação de realismo: verifica a possibilidade técnica e financeira de implementação do requisito.

5. Facilidade de verificação: averigua se o requisito é passível de verificação após a implementação.

Algumas técnicas são utilizadas para a validação dos requisitos, na concepção de Sommerville (2008):

1. Geração de casos de uso para teste: trata-se da geração de casos de testes para os requisitos, com o intuito de verificar se é possível implementá-los.

2. Revisão de requisitos: no geral é realizada por uma equipe de revisores que não participou da etapa de construção de requisitos.

3. Prototipação: é uma técnica que permite a melhor visualização dos requisitos por parte dos usuários. Dessa forma, não há a necessidade do usuário aguardar o término do desenvolvimento do projeto para observar o requisito já implementado.

Essas técnicas auxiliam a visualização e possibilitam melhor abstração dos requisitos no momento da validação.

2.2.7. Gerenciamento de Requisitos

A etapa de gerenciamento de requisitos tem como objetivo o controle e o gerenciamento de mudança nos requisitos. Essa etapa é realizada de forma simultânea com outras etapas. As três principais razões para a realização do gerenciamento de requisitos são:

1. As entregas são realizadas de forma incremental, após cada entrega, os requisitos são incorporados e entregues na próxima fase, isso demanda um gerenciamento mais cuidadoso dos requisitos.

2. Requisitos mutáveis costumam ser as principais fontes de manutenção e reengenharia.

3. Diversas organizações possuem sistemas legados que não podem ser substituídos de imediato, e que devem ser gerenciados de forma que possam evoluir. (GRANDE; MARTINS, 2006).

De forma geral o gerenciamento de requisitos pode ser definido a partir das seguintes premissas:

1. Realizar o gerenciamento dos requisitos acordados.

2. Gerenciar o relacionamento entre os requisitos.

3. Gerenciar a dependência entre o documento de requisitos e os demais documentos produzidos ao longo do processo. (BATISTA, 2003).

É nesta fase de gerenciamento que os requisitos são identificados e rastreados, para que seja possível a avaliação dos impactos que podem ser causados a partir da sua implantação. Para sistemas de grande porte e com uma grande quantidade de requisitos é importante utilizar uma ferramenta para a realização do gerenciamento e rastreabilidade dos requisitos, com o objetivo de assegurar a eficiência de todo o processo. (BATISTA, 2003).

2.2.8. Principais Problemas

De acordo com Arruda (2009), os principais problemas da área de engenharia de requisitos são:

1. Falta de definição clara dos requisitos: na maioria dos projetos, os *stakeholders* não sabem exatamente o que o sistema deve fazer e isso gera impactos no momento da construção do sistema.

2. Estrutura da organização: em organizações com uma estrutura rígida há certa dificuldade em acessar os usuários que detêm as informações necessárias para o desenvolvimento do sistema.

3. Resistência por parte dos usuários: geralmente os usuários apresentam um pouco de resistência ao processo de levantamento de requisitos, especialmente quando as alterações ou a implantação de um sistema implicam em mudança na sua rotina.

4. Falta de tempo hábil para o processo de requisitos: em alguns projetos o tempo destinado ao desenvolvimento dos requisitos é reduzido e, em alguns casos, até mesmo subtraído, para que possa ser atendido o prazo de desenvolvimento e entrega.

Parte dos problemas que impactam a engenharia de requisitos é de origem externa e social, e exige muito mais do que a correta utilização das inúmeras ferramentas existentes no mercado. Exige, sim, habilidades sociais e de relacionamento por parte do analista.

Quando há resistência por parte dos usuários em colaborar com o fornecimento de informações podem ser utilizadas algumas técnicas como, por exemplo, propiciar o envolvimento dos usuários desde o início do projeto, promover encontros de conscientização sobre a forma como ocorrerá o processo de entrada do novo sistema, discorrer sobre quais serão suas vantagens, etapas etc. Em muitas situações o usuário não colabora por não ter conhecimento do projeto, por não saber, efetivamente, quais serão as vantagens e desvantagens e como a mudança irá afetá-lo.

A inclusão de técnicas e ferramentas que facilitem a documentação e manutenção dos documentos de requisitos e que facilitem a coleta dos requisitos dos *stakeholders* do projeto pode ser de grande valor para o desenvolvimento do projeto.

2.3. *Lean Thinking*

A intenção de se realizar um determinado trabalho com o mínimo de desperdício e o máximo de eficiência é uma ideia enxuta. (WOMACK; JONES, 2004).

A principal característica do pensamento enxuto é localizar e especificar o que é, de fato, valor percebido pelo cliente. Para isso, são propostos cinco princípios que podem auxiliar os gestores na aplicação do pensamento enxuto nas organizações. (COSTA; JARDIM, 2010).

a) Identifique o que é valor para o cliente: este princípio estabelece que é preciso enxergar, dentro do processo executado, o que realmente é considerado valor para o cliente. Tudo aquilo que não é tido como valor para o cliente pode ser visto como desperdício. (COSTA; JARDIM, 2010).

b) Mapeie o fluxo de valor e identifique o desperdício: este princípio estabelece que o processo atual deve ser descrito de forma visual, para que seja possível enxergar qual o fluxo a ser seguido pelo processo, de modo a entregar o valor esperado pelo cliente e, sobretudo, identificar tudo aquilo que não agrega valor ao cliente, ou seja, tudo aquilo que é considerado desperdício. (WOMACK; JONES, 2004).

c) Implante o fluxo contínuo: a implantação do fluxo de valor contínuo consiste em fazer com o que o valor do processo flua pela cadeia sem interrupções desnecessárias, ocorrendo de maneira sincronizada e eficiente. (SELLITTO; BORCHARDT; PEREIRA, 2010).

d) Adoção de um sistema de produção puxada: um determinado produto ou serviço só pode ser produzido a partir da solicitação do cliente, ou seja, só há trabalho a partir da solicitação de uma demanda. Caso não haja demanda, não há

trabalho. Desta forma, evita-se o desperdício de tempo e recurso. (SELLITTO; BORCHARDT; PEREIRA, 2010).

e) Busca pela perfeição: este princípio estabelece que, mesmo após a modelagem e implantação do processo *Lean*, é necessário buscar continuamente o aperfeiçoamento e melhoria dos processos. (WOMACK; JONES, 2004).

2.3.1. *Lean Production* − Histórico

Desde o princípio da evolução manufatureira, a produção artesanal era, em sua maioria, composta de trabalhadores altamente especializados com ferramentas simples, mas, com um processo de produção bastante flexível, sendo possível produzir exatamente o que o cliente gostaria, ou seja, um produto personalizado. (DALLA; MORAIS, 2006).

Após a Segunda Guerra Mundial, Alfred Sloan (1875-1966), da General Motors, e Henry Ford (1863-1947), da Ford, idealizaram o modelo de produção em massa, utilizado inicialmente nas indústrias automobilísticas americanas. No pós-guerra, o modelo pareceu interessante também para os europeus, que passaram a copiá-lo, sem levar em consideração suas vantagens e desvantagens.

A produção em massa, em contraponto com a produção artesanal, utilizava-se de profissionais qualificados para projetar os produtos e a manufatura era realizada pelos trabalhadores sem qualificação, que não tinham conhecimento e nem entendimento do processo que estava sendo executado. Dessa forma, o consumidor à época tinha um preço baixo em relação à produção artesanal, porém, perdia a variedade e o poder de escolha, assim como aconteceu com o primeiro automóvel produzido pela Ford − o modelo T − que era fabricado em apenas uma cor. (DALLA; MORAIS, 2006).

Porém, para o Japão, esse modelo de produção não era viável, pois, no período pós-guerra o Japão enfrentava escassez de recursos, de pessoas e de matéria-prima; não era possível fazer uso do mesmo modelo utilizado nos Estados Unidos, pelos seguintes motivos: a necessidade de se produzir uma variedade muito grande de produtos, visando atender a demanda interna; a economia do Japão que estava devastada pela Segunda Guerra Mundial. (WOMACK; JONES, 2004).

Os japoneses tinham interesse em aprender os fundamentos da produção em massa e, por essa razão, tiveram uma visão mais crítica do processo. Como o modelo americano não poderia ser aplicado no Japão e precisaria de adaptações, Kiichiro Toyoda (1894-1952) idealizou, para a fábrica de sua família, a Toyota, um novo modelo produtivo adaptado às necessidades da fábrica.

Toyoda percebeu que, para alcançar a produção em larga escala no Japão, precisava levar em consideração as limitações de sua realidade, entre elas:

a) Mercado interno: o mercado interno japonês apresentava uma demanda de veículos muito diversificada, que ia de carros luxuosos para as autoridades até veículos compactos para a população.

b) Limitação de compra de tecnologia: o Japão havia sido devastado pela guerra e, por isso, não tinha recursos econômicos para realizar grandes investimentos em maquinário.

c) Mão de obra: altamente especializada e pouco disposta a entrar nos padrões da produção em massa (com longas jornadas de trabalho e baixos salários).

d) Competição externa: havia não somente o interesse em vender carros para outros países, como também atender as demandas do mercado interno. (WOMACK; JONES, 2004).

Então, para fazer com que o modelo de fato funcionasse, a Toyota conseguiu solucionar alguns problemas antes de colocar o referido modelo em funcionamento:

a) Produção em fluxo: Produzir uma peça de cada vez, sendo que uma peça passa de um processo para o outro até ser finalizada.

b) Tecnologia flexível: Produzir uma grande variedade de produtos com o mínimo de alteração no processo.

c) Correção de erros: Ao identificar problemas na produção, os erros devem ser corrigidos imediatamente e de forma que não voltem a ocorrer. (WOMACK; JONES, 2004).

O *Lean Production* é mais do que um conjunto de ferramentas e técnicas para a redução de perdas e otimização da produção — pode ser considerado uma filosofia. Além das mudanças nos processos da organização, o *Lean Production* promove uma mudança de cultura na organização. É preciso mudar o pensamento de toda a organização. (FORTES, 2010).

O *Lean Production* altera a forma como a organização aprende e também como ela aplica o conhecimento adquirido, o que torna o processo mais limpo. Com o objetivo de tornar o conceito do *Lean Production* mais claro, Steven Spear (1999) visitou 33 unidades de produção da Toyota (primeira empresa a utilizar o *Lean Production)* e conseguiu destacar 4 regras que regem um sistema de produção *Lean*, como bem demonstram Staats e Upton (2009):

- Regra 1: Todo o trabalho deve ser altamente especificado no que se refere ao conteúdo, à sequência, ao tempo e ao resultado.

- Regra 2: Cada relação de cliente e fornecedor deve ser simples objetiva e direta, especialmente no que tange ao formato de execução das atividades.

- Regra 3: O caminho para a confecção de um determinado produto deve ser simples e direto.

- Regra 4: Qualquer melhoria a ser realizada no processo deve ser realizada sob a supervisão e o apoio de um especialista no assunto. (STAATS; UPTON, 2009).

A aplicação dessas quatro regras pode tornar o processo de implementação do *Lean Production* mais definido e de melhor aceitação entre os envolvidos.

2.3.2. Principais Conceitos do *Lean Production*

O *Lean Production* baseia-se num conjunto de conceitos que visa simplificar o modo como uma organização produz e agrega valor aos seus clientes e, paralelo a isso, eliminar todos os desperdícios. (PINTO, 2008).

Dentro desse contexto, podem ser relacionados cinco conceitos para o *Lean Production*:

a) Valor: o valor de um produto ou serviço só é expresso corretamente quando é definido a partir da perspectiva do cliente, de forma a atender uma necessidade, num determinado momento e por um preço especifico. (WOMACK; JONES, 2004). Para Pinto (2008) valor é definido como a identificação do conceito de valor para os clientes, ou seja, aquilo que os clientes querem, uma vez que é por meio deste conceito de valor que os clientes agem; eles precisam extrair o máximo de valor de seus produtos e serviços.

b) Cadeia de valor: refere-se a todas as etapas necessárias para a confecção do produto ou serviço que será entregue ao cliente; a cadeia de valor é todo o processo de produção e elaboração de um determinado produto ou serviço. (WOMACK; JONES, 2004).

A cadeia de valor permite a entrega de valor aos clientes. É a sequência dos processos e etapas que produz e entrega o valor desejado. O *Lean Production* racionaliza e simplifica cada etapa do processo. A análise da cadeia de valor baseia-se na identificação de três ações: atividades que criam valor; atividades que são inevitáveis, mas não agregam valor ao cliente; e atividades que não agregam valor e são dispensáveis. (PINTO, 2008).

c) Fluxo: organiza a cadeia de valor de modo a eliminar tudo o que não acrescente valor ou que gere desperdício. O Fluxo é a forma como as coisas acontecem até o produto ou serviço desejado ser entregue ao cliente, é o conjunto de processos que faz o valor ser perceptível ao cliente frequentemente esse conjunto possui numerosas etapas e grandes períodos de espera que, sob a ótica do *Lean*, devem ser eliminados ou ao menos reduzidos drasticamente. (WOMACK; JONES, 2004).

d) Puxar: consiste em produzir somente o necessário, de acordo com a solicitação do cliente e no momento em que o cliente necessita. (WOMACK; JONES, 2004).

e) Perfeição: consiste na melhoria contínua do processo, sempre em busca da redução do custo do produto e a eliminação do desperdício em todas as etapas do processo de produção, até alcançar um estado próximo à perfeição, tanto em termos de produtividade quanto em satisfação do cliente. (WOMACK; JONES, 2004).

Figura 2 − Princípios do *Lean*

Fonte: Elaborada pela Autora

O sistema *Lean Production* foi concebido para a solução de problemas, por meio de um processo autocrático de reengenharia. Trata-se de um processo gradual que deve partir da diretoria, ou seja, a ideia de adoção do *Lean Production* deve vir do alto escalão da empresa.

Os princípios da produção enxuta podem ser resumidos no Quadro 1.

Quadro 1 − Os Cinco Princípios da Produção Enxuta, segundo Womack e Jones

Princípios da Produção Enxuta
1. Produção em fluxo, estável, sem interrupções.
2. Produzir somente quando demandado pelo cliente ou processo posterior.
3. Entender o que é valo0r para o cliente e oferecer maior valor agregado, sem desperdícios.
4. Identificar e eliminar desperdícios ao longo de toda a cadeia de valor, da matéria-prima ao cliente final.
5. Melhoria contínua através da rápida detecção e solução de problemas na base.

Fonte: Womack e Jones (2004)

Kotter (1999) afirma que, para que o *Lean Production* possa ser implementado com sucesso, é necessário seguir oito passos:

1. Estabelecer um senso de urgência: A mudança no processo precisa ser realizada o quanto antes.

2. Formar um grupo forte para implementação da mudança: Com representantes de todas as etapas do processo produtivo, especialmente aqueles que lidam diretamente com os clientes.

3. Criar uma visão: Uma visão que seja clara e que todos os envolvidos percebam que é possível alcançar tal visão.

4. Comunicar a visão: Deve estar sempre visível para todos.

5. Distribuir poder para o alcance da visão: Todos devem possuir as ferramentas e oportunidades necessárias para alcançar tal visão.

6. Gerar pequenos ganhos: De imediato é importante que sejam alcançadas as melhorias para que todo o grupo se sinta estimulado a continuar.

7. Consolidar melhorias: As melhorias alcançadas devem se tornar hábitos enraizados no dia a dia da organização.

8. Institucionalizar o novo comportamento: Formalização do novo comportamento e abertura de canais para sugestões de melhorias e manutenção dos benefícios alcançados.

O *Lean Production* pode ser entendido como uma filosofia que tem o objetivo de aplicar os conceitos: fazer mais com menos; fazer certo da primeira vez; reduzir ao máximo o desperdício de material e o tempo de produção, para que, dessa forma, seja possível agregar valor ao cliente e trazer vantagem competitiva para a organização.

2.3.3. Ferramentas do *Lean Production*

Para a aplicação dos conceitos de *Lean Production* são encontradas algumas ferramentas que auxiliam a implementação do pensamento enxuto nas organizações.

a) Mapeamento do fluxo de valor: mapeando o fluxo de valor, utilizando uma notação específica, é possível efetuar o mapeamento da situação atual do processo, identificar desperdícios e desconexões e, por meio dessa visão, escrever o processo final, de maneira a eliminar as falhas encontradas no mapeamento inicial. (NEVES; SILVA; SILVA, 2006).

O mapeamento de fluxo de valor por meio da notação especifica permite dissecar o processo de produção e identificar os desperdícios que ocorrem ao longo da cadeia. Nesse sentido, pode-se identificar e classificar os processos em três tipos: processos que agregam valor ao cliente; processos que são necessários e que não agregam valor ao cliente, mas devem ser mantidos por se tratarem de serviços de manutenção e qualidade; e os processos que não agregam nenhum tipo de valor ao cliente e devem ser descartados. (ELIAS et al., 2013).

b) *Layout* e ergonomia: a segunda ferramenta é *layout* e ergonomia. Deve ser uma preocupação da organização em estabelecer um fluxo lógico e coerente para os processos de fabricação e prestação de serviços. Sendo assim, é necessário adaptar o local de trabalho para que a ergonomia e a movimentação ocorram de modo mais suave possível. (NEVES; SILVA; SILVA, 2006).

c) 5S: a terceira ferramenta é o programa 5S, que visa incentivar os colaboradores a manterem a limpeza e a organização do ambiente de trabalho. O foco do programa está em cinco diretrizes:

- Senso de utilização: é a distinção de itens necessários e desnecessários e o descarte desses itens de acordo com a frequência de uso e necessidade.
- Senso de arrumação: consiste em organizar e dispor os materiais de trabalho de maneira que facilite o manuseio das ferramentas durante o processo de fabricação.
- Senso de limpeza: refere-se à limpeza e à manutenção constante das ferramentas de trabalho.
- Senso de saúde e higiene: significa manter e prover condições favoráveis à saúde física, mental e psicológica dos funcionários, além da correta sinalização de locais perigosos e correta utilização dos EPIs (Equipamentos de Proteção Individual).

- Senso de autodisciplina: prega a educação e a obediência às regras de trabalho. (COSTA; MONTEIRO JUNIOR; SILVA, 2011).

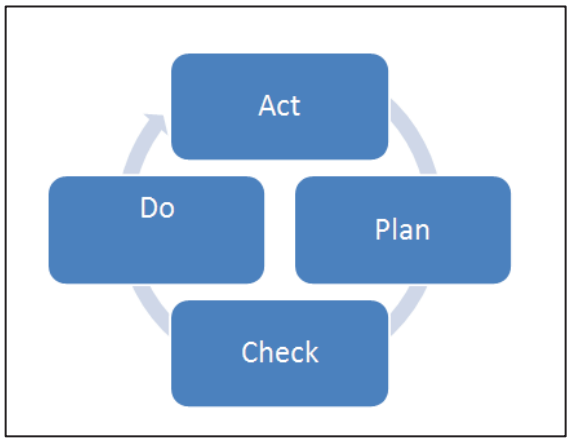
Kaizen: traduzindo para o português significa melhoria contínua. Trata-se de uma metodologia que busca o melhoramento contínuo dos processos e que deve envolver e motivar os colaboradores a participar. O objetivo principal é eliminar o desperdício de todas as atividades. (DALLA; MORAIS, 2006).

d) O *Kaizen* utiliza como ferramenta o PDCA, que foi proposto pela primeira vez em 1931 por Walter Shewart (1891-1967). O ciclo do PDCA consiste em:

- *Plan:* planejar as ações que serão executadas.
- *Do:* implementar o plano de melhoria.
- *Check:* checar e analisar as ações implementadas.
- *Act:* executar ações de melhoria ao longo de todo o processo.

A Figura 3 mostra o ciclo do PDCA.

Figura 3 – Ciclo do PDCA



Fonte: Elaborada pela Autora

O *Kaizen* possui dez princípios fundamentais:

- Rejeitar o estado atual das coisas, abandonar velhos padrões.

- Pensar em como fazer ao fazer, ao invés de procurar explicações sobre porque não pode ser feito.
- Propostas de melhorias devem ser colocadas em prática imediatamente.
- Não procurar a perfeição de imediato; buscar ganhar 60% de imediato.
- Corrigir o erro no momento e no local onde ele ocorre.
- Buscar ideias no momento da dificuldade.
- Procurar sempre a causa real de um problema.
- Levar em consideração as ideias de todo o grupo e não esperar pela ideia de uma única pessoa.
- Experimentar e depois validar.
- A busca pela perfeição é contínua. (CRUZ, 2013).

e) *Standard Work:* nesta ferramenta os operadores conhecem todo o processo, sabem exatamente o que vem antes e o que vem depois de cada atividade executada e podem sugerir melhorias e readequações. O *Standard Work* é uma das ferramentas de suporte para o *Kaizen*, uma vez que para se realizar a melhoria ou remodelagem de um sistema é necessário que as atividades estejam alinhadas, ou seja, que elas sigam uma sequência lógica.

Para a implementação do *Standard Work* deve ser definido o melhor fluxo de trabalho entre todas as alternativas, realizar a documentação de todas as etapas e distribuí-las entre os operadores. (CRUZ, 2013).

f) *Kanban:* é um conceito de produção puxada, ou seja, a ideia central é a de que uma etapa do processo de produção puxe a outra. Dentro desse conceito fica evidenciada a relação cliente/fornecedor entre as etapas da produção. (PINTO, 2008).

Existem ainda diversas ferramentas utilizadas durante a implantação do *Lean Production* e que normalmente levam em consideração a organização e o seu tipo de negócio.

O pensamento enxuto pode auxiliar as empresas a criar um fluxo de valor em toda a sua cadeia e agregar esse valor ao cliente, de maneira que atenda às suas necessidades. (NEVES; SILVA; SILVA, 2011).

Entre os principais benefícios da utilização do *Lean Production* listados pelo *Lean Institute* dos EUA vale destacar: crescimento dos negócios; aumento da produtividade; redução de estoques; aumento no nível de serviço; aumento da qualidade de serviço prestado ao cliente; redução de defeitos; maior envolvimento e participação das pessoas.

2.3.4. *Lean Software Development*

O *Lean Software Development* (LSD) é uma metodologia de desenvolvimento de software baseada nos conceitos de produção enxuta da Toyota.

A utilização da filosofia *Lean* no processo de desenvolvimento de software foi proposta inicialmente por Mary e Tom Poppendieck, em 2003. O *Lean Software Development* combina os princípios mais fortes do *Lean* que são: noção de valor, minimização do desperdício e maximização dos resultados. A metodologia é indicada especialmente para empresas de médio e grande porte e que já possuam alguma experiência com o *Lean;* equipes que já tenham tido alguma experiência com desenvolvimento ágil também são indicadas. Entretanto, qualquer empresa que se disponha a aprender e a mudar seu comportamento pode se beneficiar do *Lean Software Development.* (NORRMALM, 2011).

A essência do *Lean Software Development* possui características muito próximas do modelo de desenvolvimento enxuto de produto, utilizado pela Toyota. De acordo com Kosaku Yamada engenheiro chefe do Lexus ES 300 (modelo de carro produzido pela Toyota), a real diferença entre a Toyota e os demais concorrentes não é o Sistema Toyota de Produção e sim o Sistema Toyota de Desenvolvimento de Produtos. O desenvolvimento de produto tem características muito diferentes em suas operações, ou seja, as técnicas aplicadas a operações nem sempre serão bem sucedidas no desenvolvimento de produtos. (POPPENDIECK; POPPENDIECK, 2003).

Um produto ao ser lançado tende a seguir dois caminhos: ter um excelente design e atender de maneira satisfatória as exigências do mercado ou ficar aquém

tanto na expectativa dos clientes quanto nas suas receitas; os produtos lançados pela Toyota tendem a seguir o primeiro caminho; esse sucesso se deve ao fato de que seus engenheiros são especialistas em seus produtos. Para que um engenheiro alcance um cargo de gerência é necessário primeiro que ele desenvolva uma vasta experiência no desenvolvimento do seu produto e tenha um profundo conhecimento técnico a respeito dele, somente a partir desse ponto é que ele pode assumir a gerência. (POPPENDIECK; POPPENDIECK, 2003).

Essa metodologia do *Lean Software Development* começou a ser aplicada em meados de 2002, como metodologia ágil, voltada para o desenvolvimento de software, mas, também pode ser utilizada para o gerenciamento de projetos. (PEREIRA, 2010).

O *Lean Software Development* foi idealizado por Mary e Tom Poppendieck, com a intenção de eliminar todos os desperdícios do desenvolvimento de software, cortando os excessos e produzindo somente aquilo que é necessário ao cliente.

Entende-se por desperdício na visão do *Lean Software Development*: códigos e funcionalidades desnecessários, tempo de espera no processo de desenvolvimento de software, requisitos ambíguos, burocracia e problemas de comunicação interna. Basicamente, o *Lean Software Development* consiste em identificar qual é o valor para o cliente e a partir daí combater todos esses e outros desperdícios que impeçam a entrega desse valor ao cliente. (NORRMALM, 2011).

Para Poppendieck e Poppendieck (2003), o desenvolvimento *Lean* de software trata-se, na verdade, de um subconjunto da área de desenvolvimento de produto, uma vez que boa parte dos softwares é desenvolvida com a visão de produto, com exceção dos softwares embarcados.

Tanto o sistema de produção Toyota, como o sistema de desenvolvimento Toyota, deriva dos mesmos princípios; para se realizar uma implementação de sucesso do *Lean Software Development* é necessário compreender esses princípios.

2.3.5. Princípios do *Lean Software Development*

O *Lean Software Development* conta com sete princípios que norteiam a sua prática. São eles:

Princípio 1 − Eliminar o desperdício: De acordo com Poppendieck e Poppendieck (2003), eliminar o desperdício consiste em diminuir a linha do tempo entre o momento que o cliente faz o pedido e o momento em que se recebe o dinheiro acordado pela entrega. A redução dessa linha do tempo é obtida através da eliminação de desperdício.

Para promover a eliminação do desperdício o primeiro passo é reconhecê-lo, partindo do princípio que desperdício é tudo aquilo que não agrega valor ao cliente; assim, a melhor maneira de se eliminar o desperdício é reconhecer o que o valor realmente é, pois, quando se trata de software, essa noção de valor é bastante volátil. Nem sempre o cliente sabe do que ele realmente precisa e essa ideia muda quando ele se depara com o software pronto e funcionando. Nesse sentido, é possível desenvolver esse senso de valor e satisfazer continuamente o cliente.

O segundo passo é desenvolver a capacidade de enxergar o real desperdício. Desperdício é qualquer atividade, processo ou funcionalidade que impeça de se desenvolver e entregar aquilo que o cliente realmente deseja, no tempo e no contexto que ele deseja; tudo o que não agrega valor ou causa atraso ao cliente é considerado desperdício.

Para Poppendieck e Poppendieck (2003) os maiores desperdícios em software são:

a) Estoque: Em termos de software, o estoque refere-se a um software parcialmente acabado; este se torna obsoleto, faz a organização perder dinheiro, consome tempo em manutenções e esconde uma série de problemas de qualidade.

b) Requisitos altamente voláteis: Ocorre em casos em que a especificação se dá muito antes da codificação, e os testes acontecem muito depois da codificação e o eixo testar-corrigir se torna inevitável. Todavia, essa situação é apenas uma parte de um problema maior que é o atraso na integração entre os componentes do software.

c) Antecipação de Funcionalidades: Para Poppendieck (2003), esse é o maior desperdício em termos de software, pois foram investidos tempo e dinheiro em funcionalidades que não foram solicitadas pelo cliente e tampouco agregam valor ao seu negócio. E, ainda, elevam o custo do projeto e aumentam consideravelmente a complexidade do software.

Há autores que trazem outras situações que são consideradas como desperdício em software. Para Pereira (2010) os seguintes pontos devem ser combatidos durante o processo de eliminação de desperdício:

a) Funcionalidades incompletas: significam tempo, esforço e custo que não serão utilizados pelo cliente.

b) Excesso de processos: demanda muito tempo tanto para elaboração como para a execução, aumenta a complexidade do projeto e não agrega valor ao cliente.

c) Criação de documentos: a criação de documentos em excesso consome tempo e recurso e dificilmente agrega valor ao cliente. É uma via de comunicação de mão única, ou seja, não há interação entre o leitor e o autor do texto, o que pode gerar problemas de entendimento, além de ser um processo burocrático. O ideal é que sejam produzidos somente os documentos que, de fato, serão utilizados.

d) Processos complexos: dificulta o entendimento da equipe e aumenta a quantidade de documentação. Por consequência, aumenta a burocracia.

e) Troca de tarefas: caracteriza-se como outra forma de desperdício, pois a constante mudança de contexto do desenvolvedor acarreta na redução da produtividade.

f) Esperas: a espera por requisitos, testes ou *feedback* atrasa o fluxo de desenvolvimento e retarda a localização de erros. (PEREIRA, 2010).

Princípio 2 − Integrar Qualidade: Integrar qualidade desde o inicio do desenvolvimento significa evitar ao máximo a ocorrência de erros. Existem dois tipos de inspeção: inspeção após a ocorrência de defeitos e a inspeção para prevenir defeitos. Para que efetivamente seja alcançada a qualidade ao longo do processo é necessário inspecionar o produto a cada pequeno passo, localizar o defeito, parar a produção, encontrar a causa do defeito e corrigi-lo imediatamente.

O que comumente ocorre é uma maior preocupação em localizar o erro para registrá-lo no sistema de rastreabilidade e manter, assim, a fila de erros atualizada. Já para o conceito de *Lean,* o mais importante é localizar e corrigir o erro para que não haja fila de erros; dependendo do tamanho e da complexidade do software nem o sistema de rastreabilidade de erros deve existir.

Importante ressaltar que dentro do paradigma do *Lean* esse é um conceito perfeitamente tangível de ser alcançado e sua vantagem está na possibilidade de utilização de outras ferramentas para auxiliar no alcance da meta.

O *Test Driven Development* (TDD) é uma ferramenta de desenvolvimento ágil de software que consiste em escrever os testes unitários e de aceitação antes de iniciar a escrita do código; a integração entre testes e código ocorre assim que possível e se repetem em curtos intervalos de tempo até o final do desenvolvimento. No final do dia é executado um teste mais resistente e completo para verificar se realmente não há nenhum erro e caso seja localizado um erro a produção do código é paralisada e só retorna após a correção total do erro em sua causa raiz.

Essa ferramenta pode ser muito útil na implementação do *Lean Software Development* e na integração da qualidade ao processo de desenvolvimento.

Um paradigma importante que esse princípio ajuda a quebrar é a ideia de que a função dos testes é encontrar defeitos quando, na verdade, a função dos testes é assegurar que o que foi desenvolvido está de acordo com o esperado e está correto. Nesse cenário, encontrar erros deve ser a exceção e não a regra; a utilização da ferramenta TDD como apoio nesse processo pode reverter em benefícios para todos: programadores, analistas e clientes.

Princípio 3 − Criar Conhecimento: significa entender que o software é muito mais do que uma documentação de especificação, que se trata de um organismo vivo, que cresce e se modifica de acordo com as tendências do mercado.

Alan MacCormack, professor da Escola de Administração de Harvard, estuda como as organizações aprendem e, em seus estudos, identificou quatro práticas que levam a um desenvolvimento bem sucedido. (POPPENDIECK; POPPENDIECK, 2003):

a) *Releases* Breves: com poucas funcionalidades e distribuídos com frequência, para que os clientes possam avaliar e dar o seu feedback.

b) *Build* diária e feedback rápidos dos testes de integração.

c) Equipe/líder com *feeling* (instinto) para tomar as decisões corretas.

d) Arquitetura modular que permita adicionar com facilidade novas funcionalidades.

É importante ressaltar que não basta apenas o ambiente organizacional contar com essas características, o compartilhamento de conhecimento deve ser incentivado o tempo todo, de forma a criar uma cultura de compartilhamento. (POPPENDIECK; POPPENDIECK, 2003).

Princípio 4 - Adiar comprometimentos: consiste em treinar a equipe para tomar decisões somente após ter o máximo de informações a respeito do cenário; dessa forma, a decisão é a mais acertada possível.

O mesmo vale para a tomada de decisões irreversíveis que, muitas vezes, devem ser tomadas ao longo do projeto de desenvolvimento de software. Nesse caso é importante planejar as decisões para serem tomadas somente no último momento, todavia, isso não significa que todas as decisões devem ser levadas até a última consequência.

A maioria das decisões deve ser reversível. As decisões podem ser executadas e caso não funcionem facilmente podem ser revertidas, pois, um sistema não precisa ser totalmente flexível a todas as alterações, mas, deve possuir pontos de mudança que sejam facilmente alterados.

Também é possível retardar ao máximo a tomada de algumas decisões, e tomá-las somente quando o cenário estiver mais definido, o que assegura uma tomada de decisão mais assertiva. Entretanto, para praticar esse princípio é preciso que a equipe esteja preparada para lidar com mudanças frequentes. (FADEL; SILVEIRA, 2010).

Princípio 5 − Entregar rápido: Realizar entregas rápidas e com qualidade é uma característica admirável atualmente no mundo de desenvolvimento de software. A possibilidade de realizar rapidamente a entrega de um produto não dá ao cliente a chance de solicitar alterações ou mudar processos que possam impactar o desenvolvimento.

As entregas rápidas possibilitam, ainda, que o cliente já comece a utilizar o produto e a extrair benefícios dele, além de facilitar a rastreabilidade e a realização dos testes. (POPPENDIECK; POPPENDIECK, 2003).

Princípio 6 − Respeitar as Pessoas: Esse princípio refere-se a respeitar o trabalho dos desenvolvedores e de toda a equipe responsável; significa atribuir

confiança e condições de trabalho favoráveis para que a equipe possa ter um bom rendimento e, principalmente, proteger a equipe de interferências externas que podem prejudicar o andamento dos trabalhos.

Os três princípios de gestão de pessoas, do sistema de desenvolvimento de produtos da Toyota, trazem uma visão ampla do que esses princípios representam:

a) Líder empresarial: as pessoas gostam de trabalhar com produtos de sucesso e os produtos de sucesso geralmente possuem bons líderes de equipe, uma organização que reconhece essa equação e a respeita, garante que a equipe seja motivada e engajada.

b) Mão de obra técnica: a organização deve assegurar que a equipe seja altamente especializada e prover essa especialização caso ela ainda não exista, pois, isso promove uma vantagem competitiva sustentável.

c) Responsabilidade baseada em planejamento e controle: as pessoas têm acesso a planos genéricos e objetivos razoáveis; ao invés de mostrar o que fazer e como fazer, as equipes se auto-organizam e trabalham para atender as expectativas. (POPPENDIECK, 2003).

Princípio 7 — Otimizar o todo: otimizar toda a cadeia de valor da organização e não apenas o processo de desenvolvimento de software, desde o recebimento da solicitação até a implementação definitiva do software.

Ao realizar a otimização somente da etapa de desenvolvimento, realiza-se o que se chama de sub-otimização do processo que não conduz a resultados satisfatórios e tampouco aos resultados desejados, pois ao otimizar apenas uma parte acaba-se perdendo tempo na próxima etapa que não foi otimizada e, assim, sucessivamente, (POPPENDIECK; POPPENDIECK, 2003).

Encerra-se neste capitulo a apresentação da fundamentação teórica e, no capítulo que segue, desenvolve-se a metodologia utilizada neste estudo.

3. METODOLOGIA E APRESENTAÇÃO DA PESQUISA

Este capítulo apresenta a metodologia utilizada neste trabalho, com base nos objetivos estabelecidos e nos procedimentos técnicos usados para o desenvolvimento da pesquisa.

3.1. Classificação da Pesquisa

A pesquisa, na concepção de Gil (2002), pode ser classificada em: exploratória, descritiva ou explicativa.

A pesquisa exploratória tem como objetivo propiciar maiores informações sobre um tema, para explicitar melhor determinado assunto. Pode-se dizer que o objetivo da pesquisa exploratória é o aprimoramento de ideias ou a descoberta de novas intuições e percepções. (GIL, 2002).

Segundo Gil (2002) a pesquisa exploratória é feita por meio dos seguintes passos: levantamento bibliográfico; entrevistas com pessoas que tiveram experiências com o assunto tratado no estudo e, por fim, a análise de exemplos que auxiliem na compreensão.

A pesquisa descritiva tem como objetivo descrever as características de uma determinada população ou fenômeno, para o reconhecimento e estabelecimento de relação entre as variáveis. Uma de suas características mais marcantes é a utilização de técnicas padronizadas para a coleta de dados. Algumas pesquisas descritivas vão além da identificação das relações entre as variáveis e busca, ainda, entender a natureza dessas relações ─ o que faz com que a pesquisa descritiva, em alguns casos, se aproxime do perfil de uma pesquisa exploratória. (GIL, 2002).

A pesquisa explicativa estuda quais são os fatores determinantes para a ocorrência de uma sequência de eventos. Neste sentido, realiza um estudo profundo de conhecimento de uma determinada realidade e tenta explicar o porquê das coisas. Isto a torna bastante complexa e delicada, pois o risco de erros é consideravelmente maior. (GIL, 2002).

Com base na classificação supramencionada, o presente estudo trata-se de uma pesquisa descritiva; por meio da revisão bibliográfica e do questionário buscou-

se estabelecer uma relação entre a engenharia de requisitos e o *Lean Software Development.*

3.2. Delineamento da Pesquisa

Após a definição do tipo de pesquisa a ser desenvolvido, foi traçado o seu delineamento, baseado na identificação do procedimento para a coleta de dados. De maneira geral, existem dois grandes grupos de coleta de dados: o primeiro é a coleta em fontes de papel e o segundo é a coleta a partir das pessoas. (GIL, 2002).

Para este estudo foi realizada a pesquisa bibliográfica, enquadrada no primeiro grupo de procedimentos de coleta de dados, ou seja, a coleta em fontes de papel.

A pesquisa bibliográfica é realizada com base em material já elaborado, especialmente livros e artigos científicos ─ em alguns casos, estudos exploratórios também podem ser considerados pesquisa bibliográfica. Uma vantagem da pesquisa bibliográfica é a possibilidade de uma ampla cobertura de determinado assunto. Esta característica é vantajosa principalmente para estudos que requerem uma grande quantidade de dados dispersos (GIL, 2002).

O presente estudo conta com uma pesquisa bibliográfica sobre engenharia de requisitos de software e os conceitos relacionados à filosofia *Lean*, em especial *Lean Software Development.* A pesquisa bibliográfica fornece o embasamento teórico necessário para a análise dos dados, a contraposição de opiniões e, por fim, a consolidação dos resultados.

3.3. Abordagem de Pesquisa

Do ponto de vista da abordagem é possível classificar as pesquisas de duas formas: quantitativa e qualitativa.

A abordagem quantitativa tem por objetivo determinar as características de uma determinada população frente a um produto ou serviço, ou seja, a abordagem quantitativa deve ser utilizada em situações onde é possível quantificar os dados. (MORESI, 2003).

Diferente da abordagem quantitativa, a qualitativa não segue um procedimento rígido para a avaliação dos dados. Entre suas principais características estão: a análise de experiências de indivíduos ou grupos, procurando entender quais são suas impressões a respeito de um determinado tema ou de um determinado contexto; o estudo das formas de interações e comunicações entre os indivíduos envolvidos e através da investigação de documentações e relatos sobre um determinado assunto. (FLICK, 2009).

De acordo com Appolinário (2012), a pesquisa qualitativa trata da coleta de dados por meio das interações entre o pesquisador e o seu objeto de pesquisa, para posterior tratamento e análise, e busca, ainda, descrever comportamentos e situações variáveis e a pesquisa quantitativa envolve controle estatístico e controle da amostra.

A natureza da pesquisa realizada neste estudo é qualitativa-quantitativa, pois, envolve aspectos tanto da abordagem qualitativa quanto da abordagem quantitativa.

Para Creswell e Clark (2006) as vantagens da utilização da abordagem combinada de pesquisa (qualitativa e quantitativa) são:

- Ajuda a responder questões que não podem ser respondidas separadamente.
- Compensa a lacuna de ambas as abordagens.
- Traz evidências mais abrangentes para o estudo de um problema de pesquisa do que cada abordagem separadamente.

Por conta desses pontos a abordagem combinada foi o tipo de abordagem escolhida para a condução deste estudo.

3.4. Procedimentos para o Levantamento dos Dados

Para o levantamento de dados e a avaliação das práticas propostas foi utilizado o método *de Delphi*, que consiste no questionamento de um grupo de respondentes especialistas no assunto com o intuito de obter informações acerca do entendimento que os sujeitos de pesquisa possuem a respeito do tema pesquisado,

e buscar a validação do cenário proposto no questionamento, pode ser aplicado tanto em pesquisas qualitativas ou quantitativas. (CASTRO; REZENDE, 2009).

Utiliza-se o método *Survey* quando se deseja obter dados ou informações sobre características de determinado grupo de pessoas a respeito de um assunto. Geralmente, utiliza-se um questionário como meio de pesquisa. (APPOLINÁRIO, 2012). O survey foi enviado ao grupo de especialistas na área de engenharia de requisitos para que respondessem de acordo com suas impressões, conforme determina a técnica de Delphi (que consiste na aplicação do survey em um grupo de especialistas na área).

O questionário consiste de uma série ordenada de perguntas a serem respondidas, por escrito, pelos sujeitos, geralmente sem a presença do pesquisador. As perguntas podem ser respondidas por email, em papel e enviadas pelo correio, ou por meio eletrônico e formulários disponibilizados na web. (APPOLINÁRIO, 2012).

As questões inicialmente foram confeccionadas com o intuito de apurar do respondente qual o entendimento que ele possui a respeito da aplicação dos princípios do *Lean Software Development* na engenharia de requisitos de software*. S*endo assim, nas questões são apresentados cenários nos quais o respondente é levado a avaliar a aplicabilidade no cenário proposto.

O público da pesquisa é composto por especialistas na área de engenharia de requisitos de software (analista de sistemas, analista de requisitos e analista de negócios), segmentados por tempo de experiência, tipo de sistema em que atua e área de atuação da empresa.

O público da pesquisa foi extraído de grupos de pesquisa na internet que discutem assuntos relacionados à engenharia de requisitos, sendo que boa parte dos respondentes veio do grupo do IIBA (*International Institute of Business Analysis*), que é uma comunidade internacional de análise de negócios, além da divulgação em redes como Linkedin e Facebook, bem como, distribuição através de listas de contatos de empresas do segmento e universidades. A seleção da amostra foi realizada por conveniência por parte da autora deste estudo.

Ao todo a pesquisa contou com 50 respondentes, com o seguinte perfil:

- 60% atuam em grandes empresas (acima de 99 funcionários).

- 40% possuem mais de 10 anos de experiência e 27% possuem entre 5 e 10 anos de experiência; 20% possuem entre 3 e 5 anos de experiência e 13% possuem de 1 a 3 anos de experiência.

- 86% atuam em empresas do setor de serviços.

Desse total, cerca de 10% (5 participantes) foram descartados por terem menos de 1 ano de experiência.

A partir da coleta desses dados foi realizada a tabulação para elaboração das discussões e comparações com a fundamentação teórica. A pesquisa foi realizada por intermédio de meios eletrônicos e, posteriormente, da análise e tabulação dos dados.

No Quadro 2 é possível visualizar um resumo do que se refere ao delineamento da pesquisa.

Quadro 2 − Delineamento da Pesquisa

Delineamento da Pesquisa	
Tipo de Estudo	Exploratório
Delineamento	Pesquisa Bibliográfica
Natureza da Pesquisa	Qualitativa
Coleta de Dados	*Survey*

Fonte: Elaborado pela Autora

Considerando o contexto do Quadro 2, a pesquisa foi realizada de acordo com as seguintes etapas:

a) Realização da pesquisa bibliográfica a respeito do tema engenharia de requisitos de software e os conceitos e princípios do *Lean Production.*

b) Elaboração do questionário de pesquisa.

c) Realização de pré-teste do questionário de pesquisa para verificação e validação das questões.

d) Seleção dos sujeitos de pesquisa e envio do questionário.

e) Recebimento das respostas do questionário e análise dos dados.

f) Tabulação e análise criteriosa dos dados, por meio do embasamento teórico fornecido pela pesquisa bibliográfica.

g) Redação do relatório e considerações finais com os resultados consolidados do questionário.

3.5. Aplicação do Questionário

Após a realização da revisão bibliográfica e a definição dos procedimentos metodológicos, a próxima etapa da pesquisa foi a aplicação de um questionário (survey) junto a um grupo de especialistas na área de engenharia de requisitos de software.

3.5.1. Apresentação da Pesquisa Exploratória

Para a realização da pesquisa exploratória foi elaborado um questionário com o intuito de apresentar e apurar o nível de utilização dos princípios do *Lean Software Development* pelos profissionais da área de engenharia de requisitos de software.

A elaboração de um *survey* conta basicamente com os seguintes passos:

1. Construção e definição do modelo e referencial teórico que foi utilizado para a elaboração das questões.

2. Elaboração das questões e definição dos objetivos e de quais informações se deseja coletar.

3. Realização de teste piloto para validação das questões apresentadas.

4. Aplicação da pesquisa no público escolhido

5. Análise dos dados e elaboração do relatório final.

Para esta pesquisa foi elaborado um questionário com 14 questões que traçam um rápido perfil dos respondentes e, em seguida, apresenta cenários onde as ferramentas de elicitação de requisitos são utilizadas e como os princípios do *Lean Software Development* se encaixam nesses cenários.

A justificativa de realização deste questionário está em considerar a percepção dos profissionais quanto a utilização dos princípios do *Lean Software Development* na engenharia de requisitos de software e também se a combinação das técnicas é possível de ser aplicada e, se sim, qual o grau de utilização desses recursos.

3.5.2. Contribuição das Teorias Para a Construção do Questionário

Este tópico tem por objetivo fazer um resumo da revisão bibliográfica apresentada no capítulo 2, com o intuito de ajudar o leitor a compreender de forma mais ampla como o questionário foi construído.

O processo de construção do referencial teórico foi constituído a partir das definições a respeito do processo de desenvolvimento de software e engenharia de requisitos de software dada por Pressman (2011) e Sommerville (2008) apresentado no capítulo 2.

De acordo com Sommerville (2008) o processo de desenvolvimento de software consiste em um conjunto de atividades que levam à construção do software, e conta basicamente com quatro etapas: especificação, projeto e implementação, validação e evolução de software.

Para Pressman (2011), o processo de desenvolvimento de software depende basicamente de um bom projeto e do controle das variáveis que são apresentadas.

O processo de desenvolvimento de software conta com alguns modelos que auxiliam a extrair o máximo de beneficios. O Quadro 3 ilustra os modelos, sua definição e as principais vantagens e desvantagens.

Quadro 3 − Processos de Desenvolvimento de Software

Modelo	Definição	Vantagens	Desvantagens
Cascata	Um modelo tradicional onde o inicio de uma fase depende da finalização da fase seguinte; o software é construído etapa por etapa com a comunicação entre uma fase e outra que define quando uma temina e a outra inicia. (PRESSMAN, 2011; SOMMERVILLE, 2008).	1. Gera um documentação detalhada e de qualidade. (SOMMERVILLE, 2008).	1. Dificuldades em fazer com o que o projeto siga uma sequência linear. 2. O cliente pode ter problemas em definir logo no inicio do projeto, e de forma detalhada, todas as suas necessidades com relação ao projeto. 3. Demora na elaboração de uma versão inicial do projeto. (PRESSMAN, 2011).
Espiral	As atividades são executadas em espiral; o software começa com uma versão mais simples e vai sendo incrementado a cada etapa. (PRESSMAN, 2011).	1. O cliente recebe uma versão inicial e acompanha o incremento de cada etapa.	1. Não há como prever quantos ciclos serão necessários até o desenvolvimento completo. 2. É preciso dosar a velocidade de evolução que não pode ser rápida demais pelo risco de não haver tempo de acomodar todas as alterações e nem lenta demais pelo risco de afetar a produtividade. (PRESSMAN, 2011).
Processo Unificado	Trata-se de um arcabouço de técnicas que compõe parte dos processos de métodos agéis e da metodologia tradicional. Que traz a comunicação constante com o cliente e dá enfâse ao papel do arquiteto de software. (PRESSMAN, 2011).	1. Proximidade com o cliente.	
Métodos Ágeis	Consiste em um modelo onde o compartilhamento do conhecimento, a interação entre as pessoas e a satisfação do cliente é mais importante do que processos e contratos (BECK, 2001).	1. Proximidade com o cliente 2. Entregas Rápidas e incrementais. (BECK, 2001).	1. Exige uma análise delicada sobre quais tipos de projeto podem ser aplicados e cuidado redobrado quando se trata de documentação.

Fonte: Elaborado pela Autora

Sobre a engenharia de requisitos foram analisados diversos autores, que serviram como base para a construção dos conceitos utilizados na pesquisa, conforme apresentado no Quadro 4.

Quadro 4 − Definições sobre a Engenharia de Requisitos

Autor	Definição
IEEE (1984)	A especificação e entendimento dos requisitos de software do cliente para documentação e posterior desenvolvimento de um sistema de software conforme os requisitos listados.
Kotonya e Sommerville (1997)	Um conjunto de atividades com o objetivo de realizar a descoberta, análise e documentação dos requisitos.
Zave (1997)	Uma área de conhecimento multidisciplinar que utiliza aspectos sociais e humanos e desempenha um papel importante para a engenharia de software.
Atlee e Cheng (2007)	É a área responsável pelo sucesso no desenvolvimento e implementação do sistema. E esse sucesso está diretamente ligado à forma como os requisitos atendem às necessidades dos usuários e do ambiente onde estão inseridos; a engenharia de requisitos é responsável por compreender e acomodar esses requisitos.
Sommerville (2008)	São todas as etapas necessárias ao entendimento e, a partir daí, promover a geração da documentação de requisitos do cliente.
Engelsman, Franken e Iacob (2009)	É definida a partir de duas visões: a primeira visão trata a engenharia de requisitos como um processo de investigação de problemas para a obtenção de possíveis soluções e a segunda visão trata a engenharia de requisitos como a disciplina que fará a descrição do contexto, as atribuições do sistema bem como suas funções.
Pressman (2011)	Área da engenharia de software responsável pelo entendimento dos requisitos desde a comunicação até a modelagem.

Fonte: Elaborado pela Autora

Após a apresentação das várias definições sobre engenharia de requisitos, foram estudadas as etapas que constituem a engenharia de requisitos e, a partir dessas etapas, mais especificamente as etapas que se referem a elicitação de requisitos, foi construído o modelo proposto que foi apresentado aos profissionais no questionário de pesquisa.

De maneira genérica o processo de engenharia de requisitos consiste em quatro etapas distintas:

1. Análise e elicitação: trata-se da análise detalhada dos requisitos, para verificar a viabilidade técnica e operacional, além da coleta e descoberta dos requisitos junto aos *stakeholders.* (SOMMERVILLE, 2008).

2. Documentação de requisitos: etapa em que finalizada a elicitação e a análise os requisitos são documentados de maneira formal e encaminhados para a validação do cliente. (PRESSMAN, 2011).

3. Validação dos requisitos: tem o objetivo de localizar erros na especificação de requisitos e alinhar o objetivos dos *stakeholders* com relação ao software. (FERNANDES; WERNECK; BARBOSA, 2010).

4. Gerenciamento de requisitos: tem o objetivo controlar e acompanhar as mudanças nos requisitos. (SOMMERVILLE, 2008).

Após a pesquisa relacionada à engenharia de requisitos, foi realizada uma pesquisa sobre a Filosofia *Lean*, iniciando pelo *Lean Thinking* até chegar ao *Lean Software Development*.

De acordo com Womack e Jones (2004), a intenção da Filosofia *Lean* é se realizar uma tarefa com o minimo de desperdício e o máximo de eficiência, além de especificar e entregar o que de fato é valor para o cliente. A partir desse conceito, a Filosofia *Lean* evoluiu para diversos segmentos entre eles: Manufatura (*Lean Production*) e para a área de desenvolvimento de software (*Lean Software Development*), que é o foco deste trabalho.

O *Lean Production* deriva do sistema Toyota de produção. Logo após a Segunda Guerra Mundial o Japão buscava uma forma de retomar a sua produção e com isso atingir novos mercados; para isso, os japoneses foram até os Estados Unidos conhecer o modelo de produção em massa utilizado por Sloan e Henry Ford. Entretanto, eles constataram que esse modelo não era o mais adequado para o Japão e entre os principais motivos vale citar: a necessidade de se produzir uma variedade muito grande de produtos de forma a atender a demanda interna e a economia do Japão muito devastada pela Segunda Guerra Mundial. (WOMACK; JONES, 2004).

Dessa forma, os japoneses começaram a estudar um modo de estruturar um esquema de produção adequado à sua realidade no pós-guerra (pouca e especializada mão de obra; poucos recursos materiais e naturais; um mercado cada vez mais diversificado e com alta demanda). Desses estudos foi criado pela Toyota o Modelo Toyota de Produção que mais tarde seria chamado de *Lean Production.*

O *Lean Production* está alicerçado sobre cinco princípios:

- Valor: o valor de um produto ou serviço só é expresso corretamente quando é definido a partir da perspectiva do cliente de forma a atender uma necessidade em um momento e por um preço especifico. (WOMACK; JONES, 2004).

- Cadeia de Valor: refere-se a todas as etapas necessárias para a confecção do produto ou serviço que será entregue ao cliente; a cadeia de valor é todo o processo de produção e elaboração de um determinado produto ou serviço. (WOMACK; JONES, 2004).

- Fluxo: organiza a cadeia de valor de forma a eliminar tudo o que não acrescente valor ou gere desperdício. (WOMACK; JONES, 2004).

- Puxar: consiste em produzir somente o necessário. (PINTO, 2008).

- Perfeição: consiste na melhoria contínua do processo, sempre em busca da redução do custo do produto e a eliminação do desperdício em todas as etapas do processo de produção, até alcançar um estado próximo a perfeição, tanto em termos de produtividade quanto em satisfação do cliente. (WOMACK; JONES, 2004).

Importante ressaltar que para a aplicação do *Lean Production* nas organizações é necessário um estudo criterioso sobre como aplicar e quais os impactos disso na produção e também nas pessoas. Uma das características mais marcantes do *Lean* é o foco e a valorização das pessoas. Dada a eficiência do *Lean Production* nas organizações manufatureiras, o *Lean* passou a ser reproduzido em outros segmentos, entre eles no desenvolvimento de software. Em 2011, Mary e Tom Poppendieck, desenvolveram o *Lean Software Development.*

O principal objetivo do *Lean Software Development* é no desenvolvimento de software eliminar os desperdícios, cortando os excessos e produzindo aquilo que é necessário para o cliente. Para apoiar esses objetivos, Mary e Tom Poppendieck criaram sete princípios que norteiam o desenvolvimento de software a partir da perspectiva do *Lean Software Development.* São eles:

1. Eliminar Desperdício: todo e qualquer desperdício localizado ao longo da cadeia de desenvolvimento de software deve ser prontamente combatido. Na

concepção dos autores pode-se entender como desperdício: criação de novas funcionalidades, código inacabado, entre outros.

2. Integrar Qualidade: significa desde o inicio do projeto trabalhar para diminuir ao máximo a ocorrência de erros, através da verificação continua dos itens que já foram desenvolvidos e também dos itens que estão em desenvolvimento.

3. Criar Conhecimento: trata-se de promover na equipe de desenvolvimento o compartilhamento de ideias e situações e estar sempre conectado com as mudanças do mercado.

4. Adiar Comprometimentos: significa reunir todas as informações necessárias para efetuar uma tomada de decisão e tomá-la somente no último instante, de posse de todas as informações que foram colhidas.

5. Entregar Rápido: realizar pequenas, contínuas e sucessivas entregas ao cliente, de forma a atender prontamente suas necessidades.

6. Respeitar as pessoas: trata-se de respeitar o trabalho que essas pessoas estão desenvolvendo e também protegê-las de interferências externas.

7. Otimizar o todo: significa otimizar todo o processo de desenvolvimento de software e não apenas uma parte dele. (POPPENDIECK; POPPENDIECK, 2011).

A construção do questionário utilizado nessa pesquisa consiste na associação entre as principais ferramentas da engenharia de requisitos e os 7 princípios do Lean Software Development.

As ferramentas identificadas na pesquisa e que foram utilizadas são: brainstorming, prototipação, JAD, entrevista e etnografia.

Essas técnicas foram selecionadas após pesquisas na literatura atual referente ao assunto e são as técnicas consideradas de maior relevância; a partir disso as técnicas foram correlacionadas com os príncipios do Lean Software Development para a construção do questionário.

O quadro 5 exibe essa correlação:

Quadro 5 − Construção do Questionário

Princípios do *Lean Software Development*	**Ferramentas da Engenharia de Requisitos**
Eliminar Desperdícios	*Brainstorming*
Construir Certo da Primeira Vez	Prototipação
Criar Conhecimento	JAD
Adiar Comprometimento	Entrevista (Validação)
Promover Entregas Rápidas	Prototipação
Respeitar Pessoas	Etonografia
Otimizar o Todo	*Brainstorming*

Fonte: Elaborado pela Autora

Os princípios: eliminar desperdícios (princípio1) e otimizar o todo (princípio 7) foram relacionados com a ferramenta *brainstorming*, por intermédio dessa ferramenta é possível visualizar todo o processo, por meio da visão dos *stakeholders* envolvidos no projeto e, assim, localizar os desperdicios e otimizar os processos como um todo.

Os princípios: construir certo da primeira vez (princípio 2) e entregas rápidas (princípio 5) foram relacionados com a ferramenta de prototipação, que auxilia na validação e demonstração dos recursos do sistema aos usuários e contribui para entregas rápidas, uma vez que a prototipação pode ser reaproveitada no desenvolvimento do software.

O princípio: criar conhecimento (princípio 3) está relacionado com a ferramenta JAD que é utilizada para a coleta e validação de requisitos em grupo; é uma técnica estruturada que permite coletar e analisar informações relevantes acerca do projeto e que depois devem ser replicadas para o restante da equipe.

O princípio: adiar comprometimento (princípio 4) está relacionado com a entrevista, especialmente na etapa de validação de requisitos; nesse momento, pode se utilizar a validação do documento de requisitos pelo usuário para a validação de decisões importantes que precisam ser tomadas antes do início do projeto.

E, por fim, o princípio: respeitar pessoas (princípio 6) está relacionado com a ferramenta de etnografia; nessa técnica o analista observa o trabalho dos usuários e a partir dessa observação realiza a sua análise e sugestões de mudança no processo.

Dessa forma, a partir desse quadro, as questões da pesquisa foram elaboradas e disponibilizadas ao público da pesquisa.

4. APRESENTAÇÃO, ANÁLISE E DISCUSSÃO DOS RESULTADOS

O questionário, mencionado no capítulo anterior, com 14 perguntas foi disponibilizado eletronicamente, por meio de uma ferramenta especifica para questionários e o link enviado por e-mail para os grupos respondentes (Apêndice 1).

Antes do lançamento da pesquisa foi realizado um pré-teste com 3 especialistas na área de engenharia de software que realizaram a avaliação das questões e realizaram seus apontamentos, após essa validação o questionário foi disponibilizado ao público.

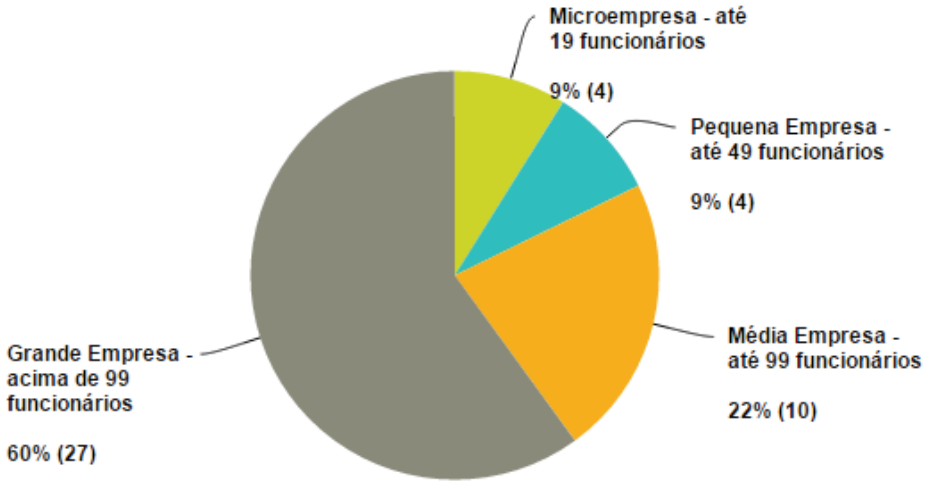
A pesquisa ficou disponível para os participantes no período entre o dia 01 de outubro de 2014 até o dia 05 de janeiro de 2015.

4.1. Perfil da Empresa e dos Respondentes

A primeira questão caracteriza o porte de empresa e contou com as seguintes opções: Microempresa (até 19 funcionários); Pequena Empresa (até 49 funcionários); Média Empresa (até 99 funcionários); Grande Empresa (acima de 99 funcionários), segundo dados do Serviço Brasileiro de Apoio às Micro e Pequenas Empresas − SEBRAE (2006).

Conforme demonstra o Gráfico 1, a maioria dos respondentes atua em grandes empresas, cerca de 60% do total de respondentes.

Gráfico 1 − Porte da Empresa onde atua

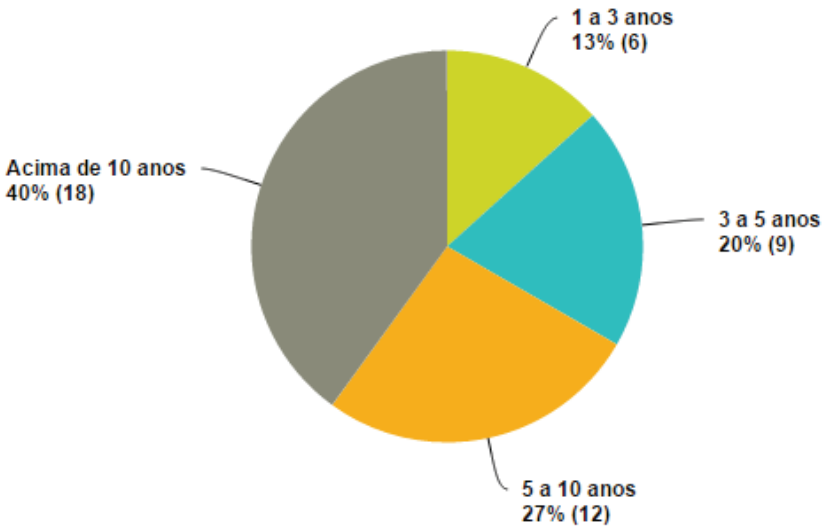


Fonte: Elaborado pela Autora

O público da pesquisa consiste em profissionais da área de engenharia de requisitos com as seguintes faixas de experiência: 0 a 1 ano; 1 a 3 anos; 3 a 5 anos e acima de 10 anos de experiência, conforme Gráfico 2.

As respostas do grupo com 0 a 1 ano de experiência foram desprezadas por conta da pouca experiência e baixo contato com as metodologias; os profissionais com essa faixa de experiência somam 10% dos respondentes (5 participantes) e em todos os dados apresentados esse percentual foi excluído.

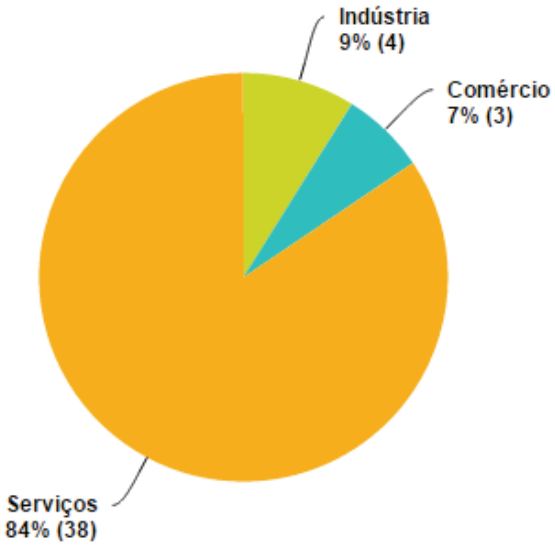
Gráfico 2 − Tempo de Experiência



Fonte: Elaborado pela Autora

Por fim, foi perguntado aos respondentes qual é o segmento da empresa onde atuam, conforme Gráfico 3, em que a maioria (84%) atua no setor de serviços.

Gráfico 3 − Segmento de Atuação da Empresa



Fonte: Elaborado pela Autora

O perfil da maioria dos respondentes pode ser resumido conforme os seguintes tópicos:

- 60% atuam em grandes empresas (acima de 99 funcionários).
- 40% possuem mais de 10 anos de experiência e 26,67% possuem entre 5 e 10 anos de experiência.
- 84% atuam em empresas do setor de serviços.

No total, a pesquisa contou com 50 respondentes, dos quais 5 (10%) foram desprezados por conta do tempo de experiência. Dessa forma, a pesquisa passou a contar com 45 respondentes válidos. Os respondentes têm origem de grupos organizados de profissionais da área de requisitos, análise de negócios e projetos de software.

4.2. Apresentação dos Resultados

O questionário foi elaborado com o objetivo de apurar a percepção dos profissionais da área de engenharia de requisitos sobre a utilização dos princípios do *Lean Software Development* na área de engenharia de requisitos.

Para isso, as questões foram elaboradas mesclando técnicas de elicitação de requisitos e os princípios do *Lean Software Development*, cuja abordagem foi utilizada por dois motivos:

1. Mesmo com o número elevado de técnicas de levantamento de requisitos, todos os anos são registradas taxas altas de falhas em projetos por conta da má especificação e erros no levantamento de requisitos.

2. A criação de cenários de trabalho com as técnicas de elicitação é uma forma mais didática de apresentar aos profissionais os princípios do *Lean*.

Inicialmente os resultados são apresentados individualmente, questão a questão. Dessa forma, é possível entender e perceber o nível de utilização de cada técnica e também a percepção de cada profissional a respeito dos princípios do *Lean*. Em seguida, os dados são cruzados para que se possa analisar se há alguma diferença na percepção sob dois aspectos: tempo de experiência e porte de empresa.

Ao final da apresentação é exibido o texto contendo as considerações finais relacionadas à pesquisa e as resultados alcançados.

4.3. Resultados Questão Por Questão

Como já citado anteriormente, as três primeiras questões referem-se ao perfil dos respondentes e as demais compreendem o questionário em si; dessa forma, a análise com base no conteúdo propriamente dito tem início a partir da questão 4.

A questão 4 trata de uma técnica de elicitação de requisitos chamada prototipação. A prototipação consiste em elaborar uma versão simples do sistema para apresentação ao cliente, e a partir dessa versão o cliente aponta novas

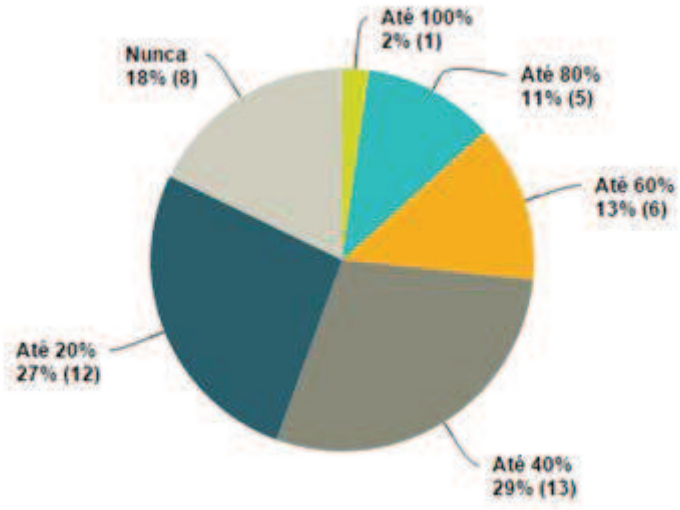
necessidades e melhorias; é possivel ainda interagir com o modelo e perceber como o software irá reagir em diferentes situações. (GAVA, 2009).

A prototipação tem um papel importante na elicitação de requisitos, especialmente em cenários onde o usuário não tem a dimensão real do que deseja que é a criação de um novo software (incluindo novos processos). A interação do usuário com um protótipo contribui para a diminuição dos erros na etapa de elicitação de requisitos. (RANGEL, 2003).

Com base nessas informações a questão 4 foi elaborada da seguinte forma:

Q.4: A prototipação é utilizada frequentemente nos projetos de desenvolvimento de software com o intuito de auxiliar tanto o analista de requisitos quanto o usuário a entender melhor as funcionalidades, a forma como o sistema irá funcionar. Entre os vários tipos de prototipação destaca-se a prototipação funcional (apresenta melhor iteração junto ao usuário, clique de botão, anexar arquivos, preenchimento de campos) e que pode ser reaproveitada no desenvolvimento. Com que frequência você utiliza a prototipação funcional e a reaproveita no desenvolvimento?

Gráfico 4 − A Utilização da Prototipação



Fonte: Elaborado pela Autora

Os resultados da pesquisa mostram que 29% utilizam a prototipação em até 40% do tempo e 27% em até 20% do tempo, o que evidencia que mais da metade dos respondentes, 56% dos participantes, utilizam a prototipação com uma frequência inferior a 40% .

Isso mostra que a prototipação, embora seja uma técnica elucidativa e que pode ajudar na elicitação dos requisitos de maneira a contribuir positivamente, é pouco utilizada pelos profissionais da área.

A prototipação é uma técnica que exige: - tempo para a elaboração dos protótipos; - softwares específicos para sua elaboração; - tempo e disponibilidade do cliente em avaliar os protótipos. (VAZ; VANDERLEI; BARROS, 2011).

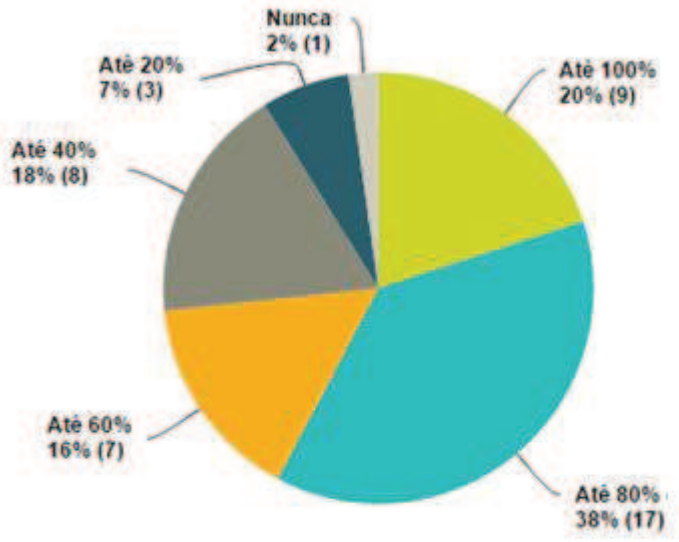
A questão 5 trata do princípio 2 do *Lean Software Development*, que é Integrar Qualidade ou ainda Construir Certo da Primeira Vez. Para Poppendieck (2011) isso significa empregar todo o esforço necessário para realizar a entrega ao cliente com o minimo de erros; para isso são utilizadas duas técnicas de inspeção: inspeção após a ocorrência de erros e a inspeção para prevenir defeitos.

Para alcançar esse estado é necessário testar o software a cada pequeno passo dado e, caso um erro seja localizado, é preciso parar o desenvolvimento e trabalhar para a localização e correção da causa do defeito; dessa forma é possivel assegurar uma entrega com o minimo de defeitos. (POPPENDIECK; POPPENDIECK, 2011).

Por meio desse conceito, foi formulada a questão 5, com resultado apresentado no Gráfico 5.

Q.5: De acordo com Mary e Tom Poppendieck, um dos princípios do Lean Software Development é: Construir certo da primeira vez, que consiste em realizar todo o esforço necessário para que a primeira entrega tenha a menor quantidade de erros possível. Em seus projetos, com qual frequência você utilizou essa linha de raciocinio?

Gráfico 5 − Princípio 2 do *Lean Software Development*



Fonte: Elaborado pela Autora

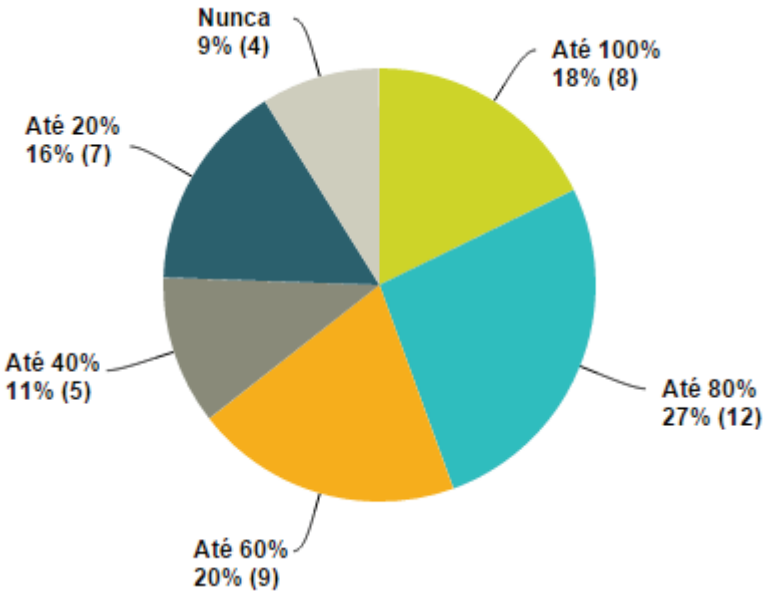
Os resultados mostram que 20% dos particpantes utilizam esse princípio em até 100% do tempo e 38% utilizam em até 80% do tempo, o que demonstra que 58% dos respondentes trabalham com essa técnica. Esse resultado é convergente com os objetivos desta pesquisa a respeito da utilização dos princípios do *Lean Software Development*.

A questão 6 refere-se ao princípio 5 do *Lean,* que trata de entregas rápidas. De acordo com Poppendieck e Poppendieck (2011), essa prática permite entregar software ao cliente para sua pronta utilização e logo no inicio do projeto, além de reduzir a possibilidade do cliente mudar de ideia ou solicitar grandes alterações.

Com essa visão foi elaborada a questão 6 com seus resutados apurados no Gráfico 6.

Q.6: Um projeto complexo e composto de várias fases está sob o seu comando e para facilitar o gerenciamento e a interação entre a equipe, e para facilitar o gerenciamento das atividades do projeto, você decide em conjunto com a equipe realizar pequenas e rápidas entregas ao cliente, de forma que ele receba primeiro os itens que mais agregam valor ao seu processo. Com qual frequência você utiliza essa técnica?

Gráfico 6 − Princípio 5 do *Lean Software Development*



Fonte: Elaborado pela Autora

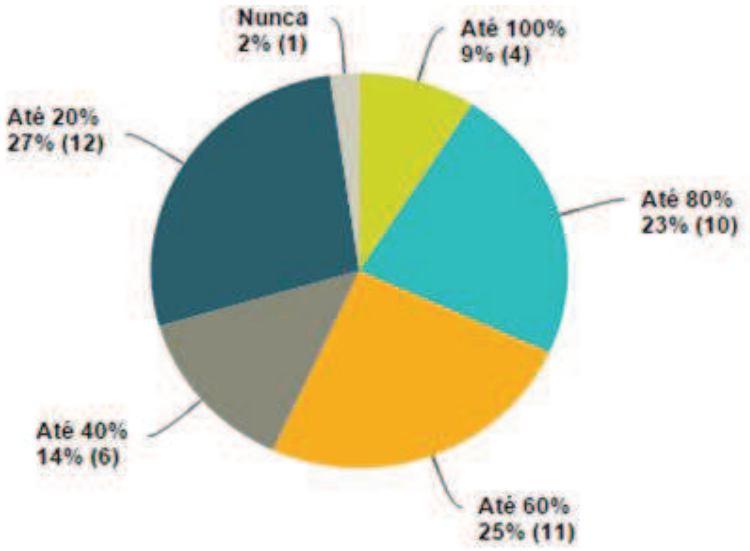
Os resultados mostram que 18% dos participantes utilizam esse princípio em até 100% do tempo e 27% em até 80% do tempo, o que totaliza 45% que atuam com esse princípio entre 80% e 100% do tempo, além de uma siginificativa parcela (20%) dos respondentes que atuam em 60% do tempo, o que reforça os objetivos dessa pesquisa a respeito da utilização dos princípios na engenharia de requisitos de software.

A questão 7 trata da técnica de *brainstorming* que consiste em: reunir os diversos *stakeholders* e incentivá-los a expor ideias e conceitos a respeito do sistema; uma pessoa é responsável por compilar essas informações e disseminar entre os participantes. Essa técnica tem um potencial de sucesso em projetos com muitos *stakeholders* e pouca definição de processo. Os valores apurados são apresentado no Gráfico 7.

Q.7: O Brainstorming é uma prática onde os participantes são estimulados (de maneira organizada) a expressar suas ideias e sugestões a respeito de um determinado assunto de maneira bem livre e inicialmente sem se preocupar com a lógica entre as ideias e, a partir desse conjunto de ideias, as informações são organizadas e compartilhadas com todos os membros da equipe que podem opinar

e sugerir mudanças. Essa prática pode ser utilizada na etapa inicial de elicitação de requisitos, pois traz de uma só vez o ponto de vista de diversos stakeholders. Com que frequência você utilizou essa técnica?

Gráfico 7 − Utilização do *Brainstorming*



Fonte: Elaborado pela Autora

A questão 7 trata da utilização das técnicas de *brainstorming* que é uma técnica de resolução de problemas de baixa a média complexidade. (PINTO, 2007).

Os resultados mostram que a técnica é utilizada de forma significativa: 31% utilizam entre 80% e 100% do tempo e 39% utilizam entre 40% e 60% do tempo, o que mostra uma boa utilização dessa técnica entre os usuários do pesquisa.

A questão 8 trata do princípio 1 do *Lean Software Development*, que é eliminar desperdício. De acordo com Poppendieck e Poppendieck (2011), tal princípio consiste em diminuir a linha entre o pedido do cliente e a entrega do produto final, e a redução dessa linha do tempo só pode ser conseguida por intermédio da redução do desperdício.

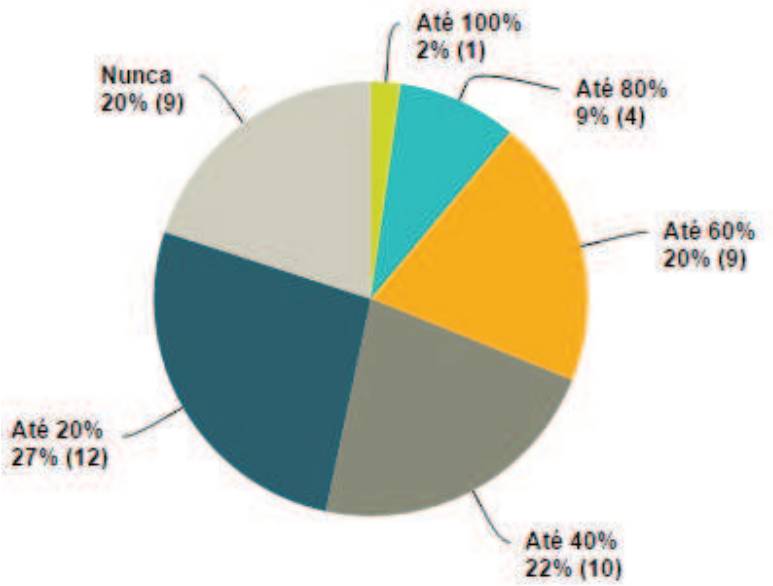
A redução de desperdício em software consiste basicamente em dois passos: identificar tudo aquilo que não agrega valor ao cliente, e ser capaz de identificar e eliminar a causa do desperdício. O desperdício pode ser classificado como qualquer evento ou processo que ocasione atraso ou impeça o desenvolvimento das funcionalidades que agregam valor ao cliente.

Na formulação da questão, o princípio 1 aparece combinado com a técnica de *brainstorming*; essa combinação ocorre porque, para que possa identificar o desperdício, é necessário compreender todo o cenário e através de diversos pontos de vista, ou seja, ter uma visão macro, e pela consolidação das ideias de um *brainstorming* é possivel ter uma noção do cenário apresentado.

Com base nesse contexto a questão 8 foi elaborada e seu resultado apresentado no Gráfico 8.

Q.8: Um dos pilares do Lean Software Development e de toda a filosofia Lean é a eliminação de desperdício nos processos; para o Lean Software qualquer coisa que é desenvolvida e que não agregue valor ao cliente deve ser considerada como desperdício e qualquer atraso que impeça o cliente de obter valor também é considerado desperdício. Quando se trata de desperdício em processos uma das principais causas é a falta de conhecimento sobre o que cada um faz em cada uma das etapas do processo. Os resultados do brainstorming podem ajudar no esclarecimento do funcionamento do processo e, por consequência, na identificação dos desperdícios. Com que frequência você utiliza o resultados do brainstorming para a identificação do desperdício?

Gráfico 8 − Princípio 1 do *Lean Software Development*



Fonte: Elaborado pela Autora

Os resultados mostram que a combinação do princípio 1 com a técnica do *brainstorming* na amostra está dispersa, o que evidencia que não há um consenso entre os participantes a respeito desse cenário.

Uma possível justificativa para essa dispersão é a dificuldade de identificação de desperdício em software, que é uma tarefa bastante complexa, pois, muitas vezes, o desperdício está subentendido em situações que são muito comuns no dia a dia das organizações. Entre essas situações pode-se citar: funcionalidades extras; códigos inacabados ou não testados que são disponibilizados em produção, e antecipação de funcionalidades. Todos esses cenários fazem parte do cotidiano dos profissionais, o que pode dificultar a visão desses itens como desperdício.

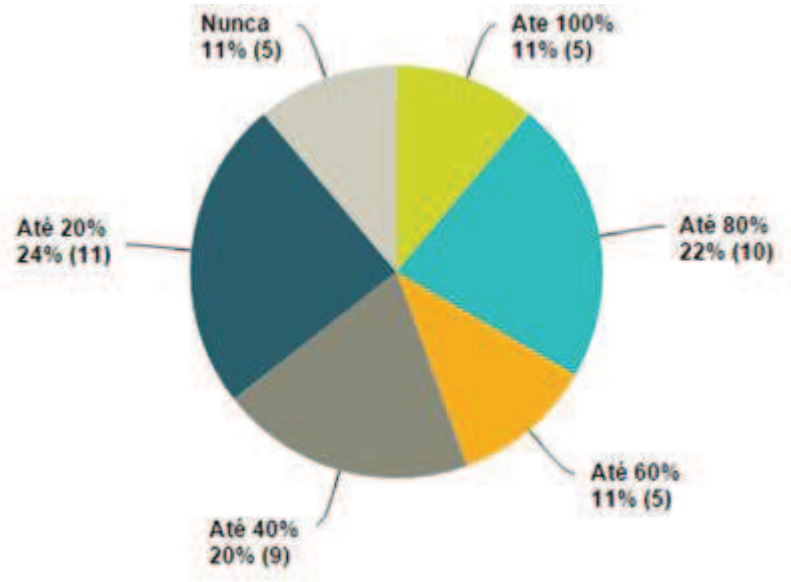
A questão 9 trata do princípio 7 do *Lean Software Development,* que se refere a otimização dos recursos; significa otimizar toda a cadeia de valor da organização e não apenas os processos relacionados ao software. (POPPENDIECK, 2011).

Ainda utilizando o *brainstorming* como técnica de apoio, a questão 9 foi elaborada com o intuito de utilizar essa técnica como ferramenta para auxiliar no entendimento e posterior otimização da cadeia de valor dos clientes, o que inclui a melhoria e otimização de processos sendo eles relacionados ao software ou não.

A partir desse contexto a questão 9 foi elaborada e sua apuração consta do Gráfico 9.

Q.9: Para o Lean Software Development a otimização do todo consiste em otimizar todo o processo do recebimento da demanda até a implantação do software no cliente, incluindo os processos do cliente. Uma das formas de realizar essa otimização é a realização do brainstorming onde todos os envolvidos são incentivados a colocar o seu ponto de vista e todos tomam conhecimento do processo, o que oferece oportunidades de otimização. O brainstorming pode ser utilizado na etapa de elicitação dos requisitos, e essa prática auxilia na identificação dos stakeholders e da forma como cada um enxerga o projeto como um todo. Na etapa de análise e elicitação de requisitos com que frequência você utilizou essa técnica?

Gráfico 9 − Princípio 7 do *Lean Software Development*



Fonte: Elaborado pela Autora

Os resultados desta questão mostram que não há uma predominância na percepção dos participantes; a amostra ficou dispersa; a média ponderada dos resultados mostra que em 39,55% do tempo os participantes se preocupam com a otimização do todo na organização.

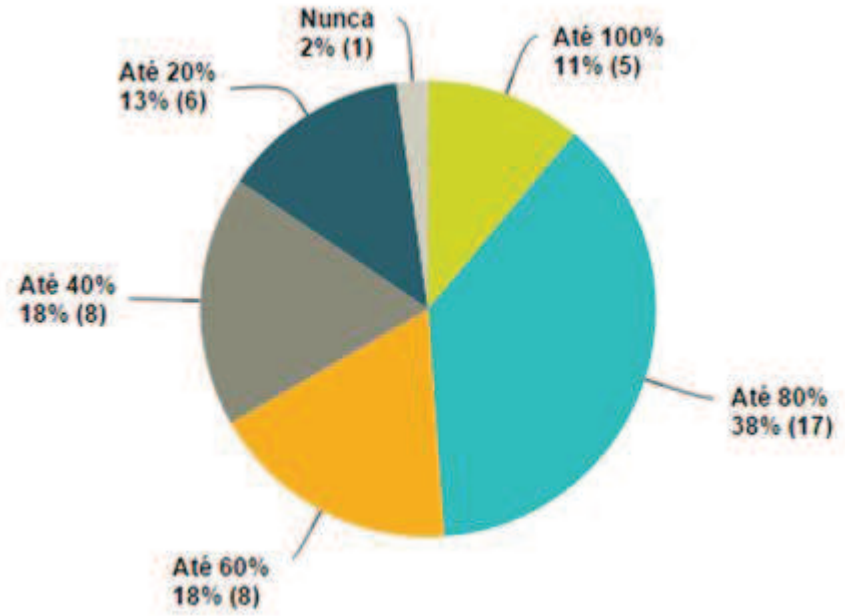
A questão 10 se refere ao princípio 4 do *Lean Software Development* chamado adiar comprometimentos, que consiste em capacitar a equipe para tomar decisões somente após ter o máximo de informações a respeito do cenário; dessa forma a decisão é a mais acertada possível. (POPPENDIECK; POPPENDIECK, 2011).

Para a elaboração da questão o princípio 4 foi combinado com a técnica de entrevista, que é uma técnica de elicitação de requisitos muito utilizada, pois, além de fornecer uma grande quantidade de informações em pouco tempo, auxilia ainda na validação dos requisitos. (FERNÁNDEZ; MORENO; TUBIO, 2013).

A partir desse contexto a questão 10 foi elaborada e seus resultados apresentados no Gráfico 10.

Q.10: De acordo com Mary e Tom Poppendieck um dos princípios (4) refere-se a adiar ao máximo possível uma decisão importante dentro do projeto e, sempre que possível, tomar decisões que sejam facilmente reversíveis. A utilização da técnica de entrevista para realizar a etapa de validação dos requisitos (o último passo antes do inicio do desenvolvimento) e nesse momento tomar as decisões de alta complexidade ou que geraram muitas dúvidas ao longo de todo o processo e realizar a tomada de decisão em conjunto com o usuário e com máximo de informação possível. Com que frequência você utiliza essa técnica?

Gráfico 10 − Princípio 4 do *Lean Software Development*



Fonte: Elaborado pela Autora

Os resultados mostram que 11% dos participantes utilizam essa combinação em até 100% do tempo e 38% utilizam essa técnica em até 80% do tempo, o que totaliza 49% dos participantes que utilizam essa técnica e com essa visão.

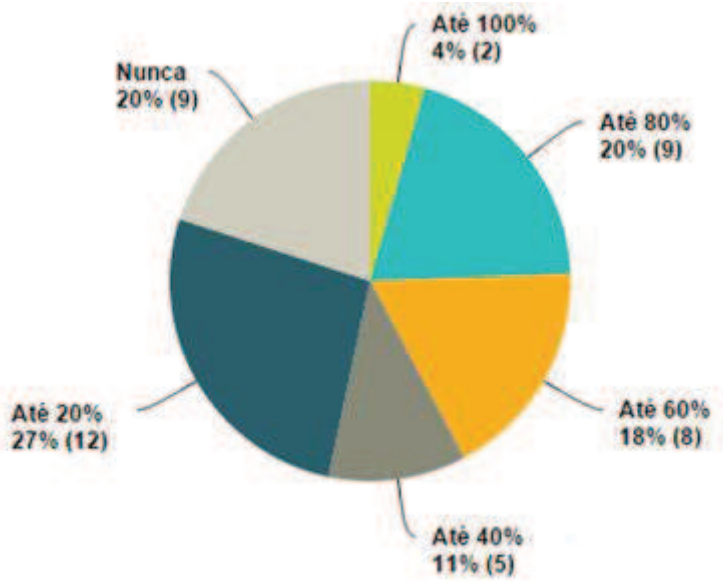
O alto índice de utilização dessa combinação é justificado pelo fato de que a entrevista é uma técnica muito utilizada, que envolve diálogo direto com os *stakeholders* e ajuda a entender como o usuário compreende e conecta as informações dentro do cenário (BORTOLI, 1999). Isso ajuda o entrevistador no momento da validação e também na tomada de decisão.

A questão 11 trata da etnografia − uma técnica de elicitação de requisitos que se refere à observação dos usuários em seu local de trabalho, durante a realização de suas tarefas, sem nenhuma interferência do engenheiro de requisitos, que apenas observa a rotina de trabalho dos usuários. (GALETTI; SPINOLA, 2005).

Com essa definição a questão 11 buscou aferir o grau de aderência dessa técnica junto aos profissionais e sua apuração se encontra no Gráfico 11.

Q.11: A etnografia é uma técnica que consiste na observação do usuário em seu local de trabalho, na execução e acompanhamento das atividades em tempo real e tomando nota das atividades realizadas. Essa técnica é bastante eficiente para o levantamento de requisitos, por fornecer a oportunidade de análise da interação entre os usuários e extrair a informação diretamente da fonte. Durante o seu tempo de atuação na área de análise de requisitos com que frequência você utilizou essa técnica?

Gráfico 11 − Utilização da Etnografia



Fonte: Elaborado pela Autora

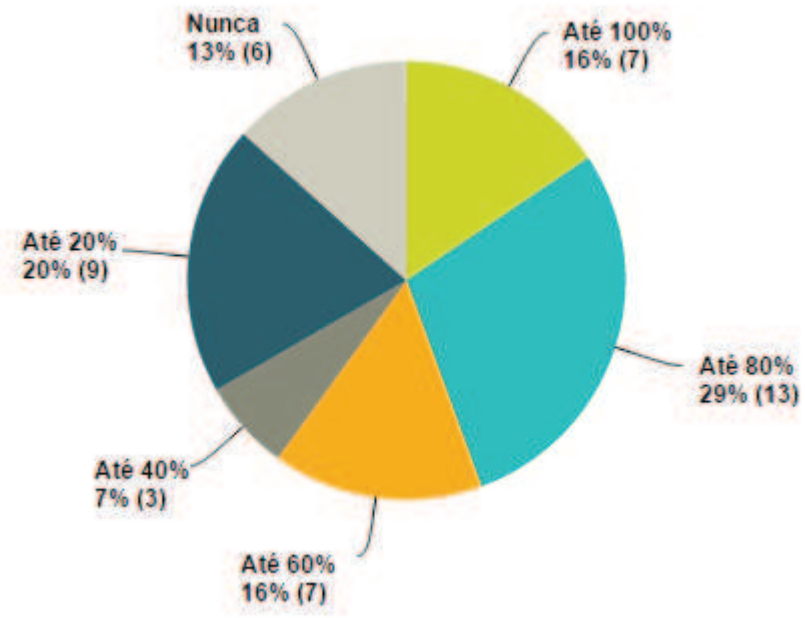
Os resultados mostram que, embora a etnografia seja uma técnica conhecida no mercado, a sua aderência é relativamente baixa: 27% utilizam a técnica em até 20% do tempo e 20% dos respondentes nunca utilizaram a técnica.

A justificativa para essa baixa aderência pode ter causas financeiras, uma vez que a etnografia envolve um custo razoável, pois é preciso manter 1 ou mais profissionais alocados 100% do tempo no cliente por um determinado período, o que acarreta um acréscimo no valor total do projeto. (SILVA JUNIOR; CARVALHO; BORGES, 2008).

A questão 12 trata do princípio 6 do *Lean Software Development*, que se refere a respeitar as pessoas: significa respeitar e valorizar o trabalho dos desenvolvedores, ou seja, entender que se aquela equipe está encarregada de um determinado projeto isso significa que eles são os mais aptos a dar andamento ao projeto; o gestor deve atuar de forma a fornecer o suporte e as condições necessárias para a realização do trabalho além de proteger a equipe de influências externas que não agregam conhecimento ao trabalho realizado. (POPPENDIECK; POPPENDIECK, 2011). A apuração dessa questão é apresentada no Gráfico 12.

Q.12: *De acordo com esse contexto a questão 12 traz as seguintes informações: Para o Lean Software Development respeitar pessoas consiste em respeitar o conhecimento de cada membro e fazer com que todos compartilhem do planejamento e das responsabilidades assumidas ao longo do processo. Outro aspecto importante desse princípio consiste em impedir que interferências externas prejudiquem o trabalho da equipe responsável, ou seja, proteger a equipe de agentes externos que interfiram nas atividades. Uma das formas mais comuns de proteger a equipe especialmente de desenvolvimento consiste em nomear um PO (Product Owner) que deve receber e filtrar todas as demandas e fazer a ponte entre a equipe de desenvolvimento e o cliente. Com que frequência você utiliza essa técnica?*

Gráfico 12 − Princípio 6 do *Lean Software Development*



Fonte: Elaborado pela Autora

Os resultados mostram que 16% dos participantes utilizam esse princípio em até 100% do tempo e 29% utilizam esse princípio em até 80% do tempo, o que totaliza 44% que utilizam esse princípio em 100% e 80% do tempo.

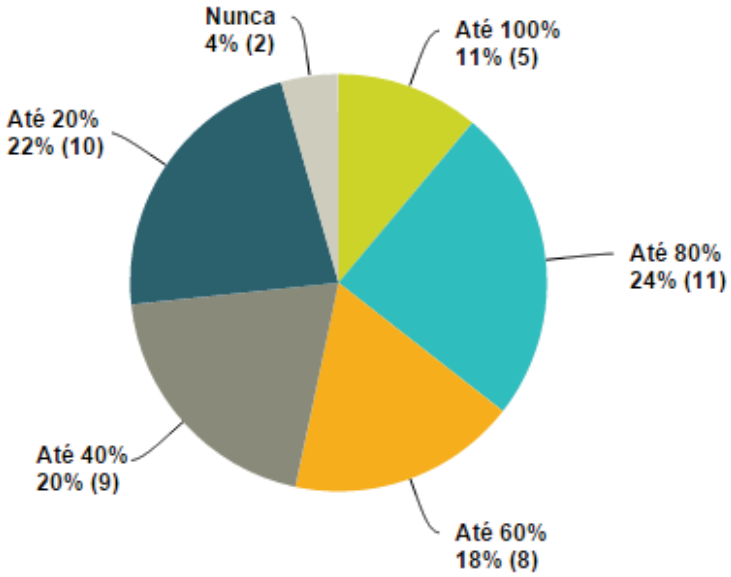
A justificativa para esse índice de utilização está no fato de que esse princípio, embora esteja descrito no *Lean,* está presente também na filosofia das metodologias ágeis de desenvolvimento de software, que atualmente possuem uma vasta utilização. A utilização do PO (*Product Owner*), em especial, é uma prática do SCRUM (metodologia ágil utilizada para o gerenciamento de projetos de software) e possui como premissa o respeito às pessoas e ao trabalho realizado por elas.

A questão 13 se refere ao princípio 3 do *Lean Software Development* chamado criar conhecimento, que consiste em: compreender que o software é muito mais que um amontoado de códigos e documentações, mas que se trata, na verdade, de um organismo vivo que cresce e se modifica de acordo com as tendências do mercado, além do compartilhamento de conhecimentos e experiências dentro da equipe, com o objetivo de acompanhar a dinâmica do software diante das demandas do mercado. (POPPENDIECK; POPPENDIECK, 2011).

A partir desse contexto a questão 13 foi elaborada com seus resultados mostrados no Gráfico 13.

Q.13: Criar conhecimento é um dos princípios do Lean Software Development; este entende que o software não é um amontoado de documentações e procedimentos, e sim um organismo vivo que cresce e evolui à medida que o tempo passa, estando constantemente agregando e criando valor. Ao realizar o desenho de um software é importante não criar processos que engessem o sistema. Nos projetos nos quais você atua ou atuou a sua preocupação com esse aspecto ocorreu com que frequência?

Gráfico 13 − Princípio 3 do *Lean Software Development*



Fonte: Elaborado pela Autora

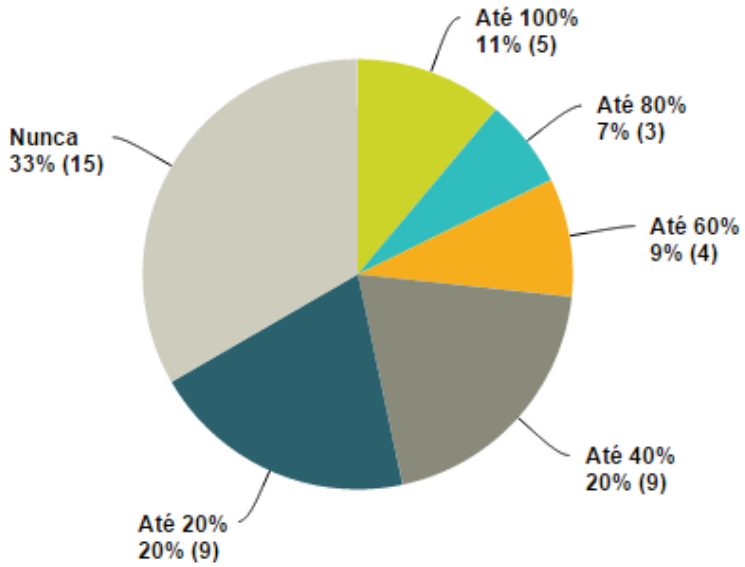
Os resultados mostram que 11% dos respondentes utilizam esse princípio em até 100% do tempo; 24% utilizam em até 80% e 18% utilizam em até 60%, o que totaliza 53% dos respondentes que aplicam o princípio criar conhecimento entre 60% e 100% do tempo, o que demonstra uma preocupação por parte dos participantes em criar e compartilhar conhecimento dentro das organizações.

A questão 14 trata do JAD *(Joint Application Design*), que é uma técnica de elicitação de requisitos desenvolvida pela IBM do Canadá, a partir de 1977, para o levantamento de requisitos. O objetivo do JAD é estabelecer uma reunião estruturada com os principais *stakeholders* do projeto. (BELGAMO; MARTINS, 2000).

Neste contexto foi elaborada a questão 14 cujos resultados são apresentados no Gráfico 14.

Q.14: O JAD (Joint Application Design) é uma técnica que permite um maior alinhamento e cooperação entre a equipe de desenvolvimento e os usuários, por meio de uma técnica de coleta e entendimento de requisitos. O JAD ajuda na captura e organização do conhecimento a respeito do software a ser desenvolvido. Com que frequência você utiliza essa técnica?

Gráfico 14 − Utilização do JAD



Fonte: Elaborado pela Autora

Os resultados mostram que a técnica possui uma baixa aderência entre os profissionais, pois 20% utilizam a técnica em até 20% do tempo e 33% nunca utilizam a técnica.

Dois fatores podem justificar essa baixa utilização: o primeiro é o fato de que para a utilização do JAD é necessário um nível de maturidade alto, tanto por parte do profissional de requisitos quanto pelo *stakeholder;* o segundo é a necessidade de

um tempo maior para a organização e aplicação da técnica, o que em determinados projetos acaba inviabilizando a sua aplicação.

4.4. Análise dos Dados Por Grupos

Após a análise inicial dos dados, que permitiu apurar separadamente a percepção dos profissionais sobre cada um dos itens questionados, surgiram alguns questionamentos:

- O tempo de experiência do profissional exerce algum tipo de influência na percepção dos profissionais?
- O porte da empresa onde o profissional atua exerce algum tipo de influência nas respostas apresentadas?

Para a análise desses questionamentos foi realizado o cálculo da média ponderada de respondentes que utilizaram os recursos apresentados em cada uma das questões considerando os 4 primeiros índices – utiliza de 40 a 100% do tempo. A partir daí foram elaborados dois gráficos para comparação em cada uma das visões: o Gráfico 15 que apresenta a visão por tempo de experiência, e o Gráfico 16 com a verificação por porte da empresa.

Gráfico 15 − Visão por Tempo de Experiência

Fonte: Elaborado pela Autora

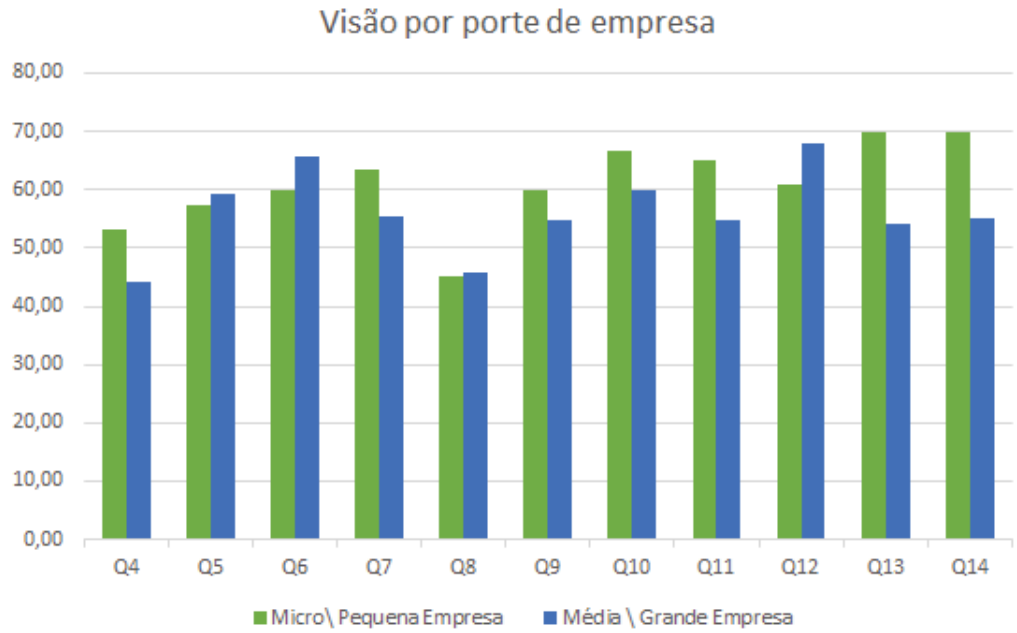
O gráfico mostra que o grupo de respondentes acima de 5 anos de experiência (nesse grupo estão os profissionais com 5 a 10 anos de experiência e acima de 10 anos de experiência) é mais aderente às ferramentas do *Lean Software Development* na engenharia de requisitos, bem como das técnicas de elicitação de requisitos apresentadas ao longo do questionário.

Uma das justificativas para essa tendência é justamente o tempo de experiência do profissional, ou seja, o nível de senioridade do profissional envolvido no projeto possui influência sobre a utilização das técnicas e sua frequência de utilização.

Esse resultado colabora ainda com o objetivo da pesquisa que é: avaliar a percepção dos profissionais sobre a utilização dos princípios do *Lean Software Development* na engenharia de requisitos. Através da visão por tempo de experiência observa-se que os profissionais possuem a percepção de que o *Lean* pode auxiliar na engenharia de requisitos e aplicam os princípios nas ocasiões citadas no questionário.

O segundo cruzamento de dados, apresentado no Gráfico 16, foi realizado a partir do porte da empresa, com o objetivo de verificar se o porte da empresa onde os profissionais atuam exerce algum tipo de influência sobre as respostas.

Gráfico 16 − Visão por Porte de Empresa



Fonte: Elaborado pela Autora

A visão por porte de empresa mostra que há uma maior aderência por parte das empresas de menor porte (micro e pequena empresa) na utilização dos princípios do Lean Software Development. Os demais itens da amostra estão dispersos e não trazem resultados significativos.

4.5. Discussão dos Resultados

Em síntese, a análise dos dados da pesquisa aponta que os profissionais da área de software possuem a percepção de que o *Lean Software Development* pode auxiliar nas práticas de engenharia de requisitos. Entretanto, alguns princípios possuem uma influência maior em detrimento a outros.

Os princípios: 2 - Construir certo da primeira vez (58%); 3 - Criar conhecimento (53%); 4 - Adiar comprometimentos (49%); 5 - Entregas rápidas (45%); 6 - Respeitar as pessoas (45%) estão entre os princípios com o maior índice de utilização entre os profissionais.

Esse fenômeno pode ser explicado pelo fato de que a maioria desses princípios já compõe a filosofia das metodologias ágeis de desenvolvimento de software e também das metodologias clássicas de desenvolvimento de projeto de software, o que facilita a assimilação dos princípios por parte dos profissionais da área, bem como a sua utilização.

Já os princípios: 1 - Eliminar desperdício (47%) e 7 - Otimizar recursos (44%) estão entre os princípios com o menor índice de utilização. A pouca utilização do princípio 1 (Eliminar desperdício) pode ser explicada pelo fato de que a identificação de desperdício em software é bastante complexa, uma vez que a maioria dos desperdícios em software é considerada como situações normais no cotidiano desses profissionais. Entre essas situações pode-se citar: funcionalidades extras; códigos inacabados e antecipação de funcionalidades.

Para o princípio 7 - Otimizar recursos, a dificuldade pode residir na padronização dos processos dentro das organizações, especialmente na área de desenvolvimento de software que trabalha por demanda e com forte pressão, o que dificulta a criação e manutenção de processos que permitam otimizar a área como um todo.

A Tabela 1 mostra os princípios que aparecem com o maior índice de utilização entre os participantes da pesquisa e os princípios com menor índice. Na tabela as linhas marcadas em verde mostram os princípios com maior utilização; as linhas marcadas em vermelho mostram os princípios com menor índice de utilização; as linhas sem marcação mostram os resultados da utilização das ferramentas de forma isolada, ou seja, sem a aplicação dos princípios.

Tabela 1 − Percentual de Utilização dos Princípios do *Lean Software Development*

Questão	Princípio \ Ferramenta	Percentual de Uso
Q4	Prototipação	55% utilizam entre 40% e 100% do tempo
Q5	Princípio 2: Construir certo da primeira vez	58% utilizam entre 80% e 100% do tempo
Q6	Princípio 5: Entrega Rápida	45% utilizam entre 80% e 100% do tempo
Q7	*Brainstorming*	32% utilizam entre 80% e 100% do tempo
Q8	Princípio 1: Eliminar Desperdício e *Brainstorming*	47% utilizam entre 20% do tempo ou nunca
Q9	Princípio 7: Otimizar Recursos e *Brainstorming*	44% utilizam entre 40% e 100% do tempo
Q10	Princípio 4: Adiar Comprometimentos	49% utilizam entre 80% e 100% do tempo
Q11	Etnografia	47% utilizam em até 20% do tempo
Q12	Princípio 6: Respeitar as pessoas	45% utilizam entre 80% e 100% do tempo
Q13	Princípio 3: Criar Conhecimento	53% utilizam entre 60% e 100% do tempo
Q14	JAD	52% utilizam entre 20% do tempo e nunca

Fonte: Elaborado pela Autora

Na análise com o cruzamento dos dados, que avalia se o porte da empresa e o tempo de experiência dos profissionais influenciam na percepção dos princípios, constatou-se por meio do Gráfico 15 que o tempo de experiência exerce sim uma influência sobre as respostas. Os profissionais que possuem entre 5 e 10 anos de experiência e os profissionais que possuem acima de 10 anos de experiência apresentam uma aderência maior aos princípios do *Lean Software Development.*

O Gráfico 16 mostra que o porte da empresa exerce uma pequena influência nas respostas, e que os profissionais que atuam em empresas de pequeno porte

possuem uma aderência maior aos princípios; esse fato pode ser justificado pela observação de que as organizações de menor porte possuem uma estrutura mais flexível, o que facilita a utilização de tais princípios.

Um ponto de atenção é a baixa utilização dos profissionais das técnicas de elicitação apresentadas ao longo do questionário (*brainstorming*; prototipação; JAD; etnografia) que são técnicas que podem auxiliar na eliminação de falhas ao longo do processo de engenharia de requisitos e que devem ser disseminadas e utilizadas pelas organizações, segundo apontam os autores apresentados neste trabalho.

Dessa forma, pode-se considerar que os profissionais da área possuem uma percepção positiva dos princípios do *Lean Software Development* e os utilizam no dia a dia na Engenharia de Requisitos.

O capítulo que segue apresenta as Considerações Finais deste estudo.

5. CONSIDERAÇÕES FINAIS

O presente estudo apresentou como justificativa a intenção de buscar um modelo de trabalho mais assertivo para a área de engenharia de requisitos, por meio da utilização dos princípios apresentados pelo *Lean Software Development*.

Para tanto, o tema foi escolhido, num primeiro momento, pelo fato de que não há relatos suficientes a respeito da relação entre o *Lean* e a engenharia de requisitos e, num segundo momento, como essa relação pode contribuir positivamente com a engenharia de requisitos, de forma que ao tratar os requisitos de software se tenha em mente que mais importante que os recursos tecnológicos a serem utilizados, são na verdade, quais benefícios aquele software irá trazer ao seu cliente, quais necessidades serão satisfeitas e como agregar valor ao negócio do cliente.

Diante dessa motivação, o estudo apresentou como questão de pesquisa: Qual a percepção dos profissionais da área de software quanto à aplicação dos princípios do *Lean Software Development* na engenharia de requisitos?

Para responder a essa questão o trabalho foi estruturado de forma a apresentar os principais conceitos a respeito do processo de desenvolvimento de software; engenharia de requisitos; *Lean Thinking; Lean Production; Lean Software Development;* e como instrumento de pesquisa foi aplicado um questionário para apurar a percepção dos profissionais a respeito dos princípios do *Lean Software Development.*

Os resultados dessa pesquisa, conforme apresentado no capítulo anterior, apontam, como resposta à questão problema apresentada na Introdução, que os profissionais possuem a percepção sobre a utilização dos princípios do *Lean Software Development*, e que essa contribuição é positiva, ou seja, gera ganhos para a engenharia de requisitos.

Essa percepção positiva também colaborou para o alcance do primeiro objetivo especifico desta pesquisa que foi: Analisar a percepção dos profissionais a respeito do *Lean Software Development* sobre a engenharia de requisitos. Os resultados apresentados na análise de grupos por porte da empresa e por tempo de experiência colaborou para o alcance do segundo objetivo especifico que foi: Verificar se o tempo de experiência e o porte de empresa exercem algum tipo de

influência sobre os resultados do questionário; conforme demonstrado no capitulo anterior o tempo de experiência exerce uma pequena influência nos resultados uma vez que os profissionais com mais experiência tem uma inclinação maior a utilizar os princípios do *Lean,* e os profissionais de empresas de menor porte também possuem uma inclinação maior a utilizar os princípios em suas atividades.

Em síntese, a pesquisa atingiu seus objetivos ao demonstrar a percepção positiva dos profissionais a respeito dos princípios do *Lean Software Development.*

Mesmo apresentando resultados positivos a pesquisa possui pontos que poderiam ter um estudo mais aprofundado, por exemplo, a fundamentação teórica foi suficiente para a execução deste trabalho, entretanto, seria interessante um aprofundamento maior em alguns temas, entre eles: *Lean Thinking* - seria interessante ir às suas origens; o processo de desenvolvimento de software poderia ter sido apresentado com um pouco mais de detalhes.

A fundamentação teórica foi útil para a construção do questionário, uma vez que toda a sua construção se deu a partir dos temas abordados na teoria deste trabalho, bem como a análise de resultados foi realizada tendo em vista os conceitos e definições apresentados.

A principal contribuição desta pesquisa é trazer para a área de engenharia de requisitos a visão *Lean*, pois, ao analisar as metodologias de mercado percebe-se que a maioria delas se preocupa muito mais com o processo de desenvolvimento de software e existem poucos estudos dedicados a aperfeiçoar a engenharia de requisitos, cujo papel é fundamental no levantamento e entendimento das necessidades dos usuários de software.

Igualmente existe uma contribuição no que se refere a mostrar para os profissionais da área de engenharia de requisitos uma forma diferente de pensar e administrar os requisitos, por meio de uma visão mais voltada ao cliente e principalmente focada em agregar valor a esse cliente.

Para trabalhos futuros um caminho interessante seria aplicar esses conceitos em um projeto e comparar os resultados com um projeto semelhante no qual a visão *Lean* não foi aplicada e, assim, verificar os ganhos reais, as possíveis restrições na aplicação e quais aspectos podem ser melhorados.

Outra sugestão seria a sistematização do pensamento *Lean* voltado exclusivamente para a engenharia de requisitos; propor um método *Lean* para a engenharia de requisitos, assim como acontece para área de serviços com o *Lean*

Service.

Por fim, a pesquisa além de atingir os resultados esperados abriu novas possibilidades para o desenvolvimento de novos estudos a respeito do tema, bem como, apresentou uma contribuição relevante para a área de pesquisa.

REFERÊNCIAS

ALI, Soh; TORABI, T. *Using Software Engineering Principles to Develop Reusable Business Rules.* ***Information and Communication Technologies***, 2005. *ICICT* 2005. *First International Conference on Publication Year*: 2005.

APPOLINÁRIO, Fábio. **Metodologia da ciência:** filosofia e prática da pesquisa**.** 2ª ed. São Paulo: Cengage Learning, 2012.

ARRUDA, Natasha de Souza. **Engenharia de Requisitos**: como prevenir e reduzir riscos. 2009. Disponível em: <http:\\www.aedb.br/seget/artigos11/30114261.pdf> Acesso em: 01 abr. 2013.

ARRUDA, Ivany Maria; LUNA, Valéria Marcia Silveira. ***Lean Service*****:** uma abordagem do *lean system* aplicada no setor de serviços. 2006. Disponível em: < www.abepro.org.br/biblioteca/ENEGEP2006_TR450301_7264.pdf> Acesso em: 17 out. 2013.

ASFORA, Diego Maciel. **Uma abordagem para priorização de requisitos em ambientes ágeis**. 110 f. Dissertação (Mestrado Ciência da Computação). Universidade Federal de Pernambuco. Recife: UFPE, 2009.

ATLEE, Joanne M; CHENG, Betty, H. C. ***Research Directions in Requirements Engineering.*** 2007. Digital Library. Disponível em: <http://dl.acm.org/citation.cfm?id= 1254725>. Acesso em: 15 out. 2014.

AUDY, Jorge Nicolas Luiz; ESPINDOLA, Rodrigo Santos de; MAJDENBAUM, Azriel. **Uma análise crítica dos desafios para engenharia em manutenção de software.** 2004. Disponível em: <http:\\wer.inf.puc-rio.br/WERpapers/artigos/artigos.../rodrigo_ espindola.pdf> Acesso em: 15 maio 2013.

BASSI FILHO, Dairton Luiz. **Experiências com Desenvolvimento Ágil.** 170 f. Dissertação (Mestrado em Ciências). Instituto de Matemática e Estatística, Universidade de São Paulo. São Paulo: USP, 2008.

BASTANI, Behzad. ***A requirements analysis framework for open systems requirements engineering.*** 2007. *Computer Laboratory, Universityof Cambridge*, Cambridge UK. Disponível em: <http://doi.acm.org/10.1145/1234741.1234753> Acesso em: 04 dez. 2014

BATISTA, Sueleni Mendez. **Uma ferramenta de apoio à definição de requisitos da MDSODI no contexto do ambiente DISEN**. 94 f. Dissertação (Mestrado em Informática). Universidade Estadual de Maringá. Maringá: UEM, 2003.

BECK, Kent. **Manifesto para o Desenvolvimento Ágil de Software.** 2001. Disponível em: http://www.manifestoagil.com.br/ Acesso em 14 ago. 2014.

BELGAMO, Anderson; MARTINS Luís Eduardo Galvão. **Estudo comparativo sobre as técnicas de elicitação de requisitos do software.** 2000. Disponível em: <http://www.unimep.br/viceacad/assessorias/pesquisa/8thCongr/anais/exatas09.htm > Acesso em: 22 jan. 2014.

BORTOLI, Lis Angela. **Um Método de Trabalho para Auxiliar a Definição de Requisitos.** 161 f. Dissertação (Mestrado em Ciências da Computação). Instituto de Informática. Universidade Federal do Rio Grande do Sul. Porto Alegre: UFRGS, 1999.

BRONZE, Reinaldo; MAIA, Felipe; LIMA, Wellington; ROCHA, Paulo. **Levantamento de requisitos de software, uma análise comparativa**. 2009. Disponível em: <http://www3.iesam-pa.edu.br/ojs/index.php/sistemas/article/viewFile/1133/771> Acesso em: 03 dez. 2014.

CAMPOS, Renato; AZEVEDO JUNIOR, Delmir Peixoto de. **Definição de requisitos de software baseada numa arquitetura de modelagem de negócios**. 2008. Disponível em: <http://www.scielo.br/pdf/prod/v18n1/a03v18n1.pdf> Acesso em: 04 dez. 2014.

CASTRO, Amparito V.; REZENDE, Magda. A técnica Delphi e seu uso na pesquisa de enfermagem: revisão bibliográfica. **Revista Mineira de Enfermagem** – REME. Vol. 13.3, 2009.

COSTA, Ricardo Sarmento; JARDIM, Eduardo. G. M. **Os cinco passos do pensamento enxuto (*lean thinking*).** 2010. Disponível em: <http://www.trilhaprojetos.com.br> Acesso em: 17 set. 2013.

COSTA, Jacqueline Santos; MONTEIRO JUNIOR, Aluísio dos Santos; SILVA, Denise Loyola. **Contribuição da metodologia 5S em uma empresa fabricante de embalagem de alumínio**. 2011. Disponível em: <http:\\www.excelenciaemgestao. org/Portals/2/.../cneg7/.../T11_0385_1655.pdf> Acesso em: 22 maio 2013.

COSTA, Luciano Antonio; ZOUCAS, Alessandra Casses; ALVES, João Bosco Mota. **Elicitação de requisitos de software no setor público: lições aprendidas e recomendações para mitigação de riscos.** 2012. Disponível em: <http://www.aedb.br/seget/artigos12/42116413.pdf> Acesso em: 21 jan. 2014.

CRESWELL, John W.; CLARK, Vicki L. P. ***Designing and conducting mixed methods research.*** EUA, Sage Publications, 1ª. Edição, 2007.

CRUZ, Nuno Miguel Pereira. **Implementação de ferramentas *lean manufacturing* no processo de injeção de plástico.** 66 f. Dissertação (Mestrado em Engenharia e Gestão Industrial). Escola de Engenharia. Universidade do Minho. Braga: UMinho, 2013.

DALLA, Werner Duarte; MORAIS, Lucílio Linhares Perdigão. **Produção enxuta:** vantagens e desvantagens competitivas decorrentes da sua implementação em diferentes organizações. 2006. Disponível em: <http:\\www.simpep.feb.unesp.br/ anais/anais_13/artigos/112.pdf> Acesso em: 04 fev. 2014.

ELIAS, Aline Fabíula; RIBEIRO, Bruno Mariani; SILVA, Fernanda Madeira; MARQUES, Daniel José. Implantação do *lean manufacturing*: benefícios na linha de produção. 2013. **Congresso Internacional de Administração**. Disponível em: <http://www.admpg.com.br/2013/down.php?id=270&q=1> Acesso em: 04 fev. 2014.

ENGELSMAN, Wilco; FRANKEN, Henry M.; IACOB, Maria E. ***Architecture-Driven Requirements.*** 2009. Disponível em: <http://delivery.acm.org/10.1145/1530000/1529344/p285-engelsman.pdf> Acesso em: 11 dez. 2014.

FADEL, Aline Cristine; SILVEIRA, Henrique da Mota. **Metodologias ágeis no contexto de desenvolvimento de software:** XP, Scrum e *Lean*. 2010. Disponível em: <http://www.ceset.unicamp.br/liag/Gerenciamento/monografias/Lean%20Agil_v8.pdf > Acesso em: 15 maio 2013.

FAGUNDES, Priscila Bastos. **Framework para comparação e análise de métodos ágeis.** 134 f. Dissertação (Mestrado em Ciências da Computação). Universidade Federal de Santa Catarina. Florianópolis: UFSC, 2005.

FERNÁNDEZ, Marta López; MORENO, Dante Carrizo; TUBIO, Óscar Dieste. ***Uso de amenazas a la validez para replicar experimentos en captura de requisitos de software.*** 2013. Disponível em: <http://dialnet.unirioja.es/servlet/articulo;jsessionid=8893A3564473B1E82F48650510C745F1.dialnet01?codigo=4335437> Acesso em: 03 dez. 2014

FERNANDES, Ulisses; WERNECK, Marcelo; BARBOSA, Glívia. **Análise Comparativa da Validação de Requisitos de Software Especificados por Meio de Duas Técnicas de Especificação de Requisitos.** I ERI-RJ – I Escola Regional de Informática do Rio de Janeiro. Instituto de Matemática, UFRJ, 13 a 15 de Abril de 2010. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/eri-rj/2010/05-63946_1.pdf> Acesso em 11 nov. 2014.

FLICK, Uwe. **Desenho da pesquisa qualitativa.** Coleção Pesquisa Qualitativa. 1ª ed. São Paulo: Bookman, 2009.

FORTES, Claudio Saenger. **Aplicabilidade de *Lean Service* na melhoria serviços de tecnologia da informação (TI)**. 168 f. Dissertação (Mestrado Profissional em Engenharia da Produção). Universidade Federal Rio Grande do Sul. Porto Alegre: UFRGS, 2010.

FRANCETO, Simone. **Especificação e implementação de uma ferramenta para elicitação de requisitos de software baseada na teoria da atividade.** 133 f. Dissertação (Mestrado em Ciência da Computação). Universidade Metodista de Piracicaba. Piracicaba: UNIMEP, 2005.

GALETTI, Italo; SPINOLA, Mauro. Um processo de engenharia de requisitos para assessorar na seleção das técnicas de elicitação de requisitos de software**. XXV Encontro Nac. de Eng. de Produção**. ENEGEP. Porto Alegre, RS. Brasil, 29 out a 01 de nov de 2005. Disponível em: <http://www.abepro.org.br/biblioteca/ENEGEP2005_ Enegep0906_0398.pdf> Acesso em: 21 jan. 2014.

GAVA, Wagner Luiz. **Processo para especificação de requisitos de software com foco de aplicação em trabalho cooperativo**. 288 f. Tese (Doutorado em Engenharia). Escola Politécnica da Universidade de São Paulo. São Paulo; POLI-USP, 2009.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa.** 4ª ed. São Paulo: Atlas, 2002.

GOMES, Alex Sandro; WANDERLEI, Eduardo Garcia. Elicitando requisitos em projetos de software educativo. **IX Workshop de Informática na Escola** – 2003. Disponível em: <http://ceie-sbc.educacao.ws/pub/index.php/wie/article/viewFile/780/766> Acesso em: 31 dez. 2014.

GRANDE, José Inácio; MARTINS, Luiz Eduardo. **SIGERAR:** Uma Ferramenta de Gerenciamento de Requisitos. 2006. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wer/2006/0010.pdf> Acesso em: 02 jan. 2015

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS − IEEE Std. 830. ***IEEE guide to software requirements specification.*** The Institute of Electrical and Electronics Engineers. New York, EUA. 1984.

_____. ***IEEE International Symposium on Requirements Engineering.*** IEEE Computer Society Press. Los Alamitos, Ca, USA, 1997, p. 44-53.

JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. ***The Unified Software Development Process.*** Readding, MA.: Addison-Wesley, 1999.

KOTTER, Jonh P. **Liderando Mudança**. 20º ed. São Paulo: Campos, 1999.

KOTONYA, Gerald; SOMMERVILLE, Ian. ***Requirements engineering ─ processes and techniques.*** John Willy & Sons, 1997.

LAHOZ, Carlos; CAMARGO JUNIOR, João Batista. **Um estudo sobre a atividade de elicitação de requisitos em projetos de software da área espacial**. Disponível em: <htpp://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Trabalho?id=13981> Acesso em: 09 nov. 2013

MARQUIONI, Carlos Eduardo. **Técnico vs. Usuário:** uma análise do processo comunicacional na engenharia de requisitos de software. 221f. Dissertação (Mestrado em Comunicação e Linguagens). Universidade Tuiuti do Paraná. Curitiba: UTP, 2008.

MORESI, Eduardo. **Metodologia da Pesquisa**. 2003. Disponível em: <http://ftp.unisc.br/portal/upload/com_arquivo/1370886616.pdf> Acesso em: 04 nov. 2013.

NEVES, Tainan Rodrigues de Oliveira; SILVA, Ruy Gomes da Silva; SILVA, Thassio Rafael Alvao. **A implantação de ferramentas baseadas na mentalidade enxuta como diferencial competitivo.** 2011. Disponível em: <http:\\www.unimep.br/phpg/mostraacademica/anais/10mostra/1/303.pdf> Acesso em: 08 maio 2013.

NEVES, José Luís. **Pesquisa qualitativa:** características, usos e possibilidades. 1996. Disponível em: <http:\\www.ead.fea.usp.br/Cad-pesq/arquivos/C03-art06.pdf > Acesso em: 25 out. 2014.

NORRMALM, Thomas. ***Achieving Lean Software Development:*** *Implementation of Agile and Lean Practices in a Manufacturing – Oriented Organization*. Uppsala Universitet. Suécia, 2011.

PALUDO, Marco A.; BURNETT Robert C.; LOCH, João M.; REIS, Dálcio. **O Desenvolvimento de Software Aplicando a Técnica *Joint Application Design.*** 2003. Disponível em: <http://www.pg.utfpr.edu.br/ppgep/Ebook/ARTIGOS/56.pdf> Acesso em: 29 dez. 2014.

PEREIRA, Guilherme Vota. **Metodologia Lean de desenvolvimento de software:** uma visão geral. 2010. Disponível em: <http://www.ceavi.udesc.br/arquivos/id_submenu/387/guilherme_metodologia_lean_de_desenvolvimento_de_software___uma_visao_geral.pdf> Acesso em: 05 maio 2013.

PINTO, João Paulo: ***Lean Thinking.*** Introdução ao pensamento magro. São Paulo: Comunidade *LeanThinking*, 2008.

PINTO, Claudia Simões. **Aplicando *Brainstorming* com Apoio de Ferramenta Computacional.** 2007. Disponível em: <http://4semana.uniriotec.br/~pimentel/disciplinas/siscolab20072/Claudia/SC20072ArtigoClaudia.pdf> Acesso em: 31 jan. 2015.

POPPENDIECK, Mary; POPPENDIECK, Tom. ***Lean Software Development***: *An Agile Toolkit. Addison-Wesley Professional,* 2003.

_____. **Implementando o Desenvolvimento Lean de Software** – Do Conceito ao Dinheiro. São Paulo: Bookman, 2011.

PRESSMAN, Roger S. **Engenharia de software** ─ Uma abordagem profissional. 7ª ed. São Paulo: Mc Graw Hill, 2011.

RANGEL, Guilherme Salum. **Pro Tool:** Uma Ferramenta de Prototipação de Software para o Ambiente PROSOFT. 223 f. Dissertação (Mestrado em Ciência da Computação). Universidade Federal do Rio Grande do Sul. Porto Alegre: UFRGS, 2003.

SANTANA, Célio A.; TIMÓTEO, Aline L.; VASCONCELOS, Alexandre M. L. Mapeamento do modelo de melhoria do processo de software brasileiro (MPS.Br) para empresas que utilizam Extreme Programming (XP) como metodologia de desenvolvimento. 2006. **V Simpósio Brasileiro de Qualidade de Software** ─ SBQS2006. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2006/009.pdf> Acesso em: 28 jan. 2014.

SANTANDER, Victor Francisco Araya**. Integrando modelagem organizacional com modelagem funcional.** 221 f. Tese (Doutorado em Ciência da Computação). Universidade Federal de Pernambuco. Recife: UFPE, 2002.

SANTOS, Victon Malcolm Rodrigues; CORREIA NETO, Jorge da Silva. Adaptação da Metodologia JAD para o Desenvolvimento de Games. 2009. **IX Jornada de Ensino Pesquisa e Extensão (JEPEX) da UFRPE**. Disponível em: <http://www.eventosufrpe.com.br/jepex2009/cd/resumos/R1152-1.pdf> Acesso em: 29 dez. 2014.

SELLITO, Miguel Afonso; BORCHARDT, Miriam; PEREIRA, Giancarlo Pereira. **Presença dos princípios da mentalidade enxuta e como introduzi-los nas práticas de gestão das empresas de transporte coletivo de Porto Alegre**. 2010. Disponível em: <http://www.prod.org.br/doi/10.1590/S0103-65132010005000009> Acesso em: 11 fev. 2014.

SERVIÇO BRASILEIRO DE APOIO ÀS MICRO E PEQUENAS EMPRESAS – SEBRAE. **Critérios de Classificação de Empresas**: EI – ME – EPP. 2006. Disponível em: <http://www.sebrae-sc.com.br/leis/default.asp?vcdtexto=4154> Acesso em 12 dez. 2014.

SILVA, Marco Aurélio Graciotto. **Uma ferramenta web colaborativa para apoiar a engenharia de requisitos em software livre**. 164 f. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional). ICMC – Universidade de São Paulo. São Paulo: USP, 2006.

SILVA JUNIOR, Luiz Carlos Lopes; CARVALHO, Paulo Victor Rodrigues de; BORGES, Marcos Roberto da Silva. **Etnografia Colaborativa:** Uma Abordagem para a Elicitação de Requisitos Cognitivos de Equipes. 2008. Disponível em: **<**http://www.abergo.org.br/revista/index.php/ae/article/view/66>. Acesso em 18 out. 2014.

SOARES, Bruno C. **Requisitos para a utilização de prototipagem evolutiva nos processos de desenvolvimento de software para Web.** 2003. Disponível em: http://homepages.dcc.ufmg.br/~rodolfo/dcc823-2-07/ Bruno3.pdf Acesso em: 26 dez 2014

SOARES, Felipe S. Furtado; MARIZ, Leila M. Rodrigues de Sousa; CAVALCANTI, Yguaratã C.; RODRIGUES, Joseane P.; NETO, Mário G.; BASTOS, Petrus R; ALMEIDA, Ana Carina M.; PEREIRA, Daniel Thiago V.; ARAÚJO, Thierry da Silva; CORREIA, Rafael S.M.; ALBUQUERQUE, Jones. **Adoção de Scrum em uma fábrica de desenvolvimento distribuído de software.** 2007. Disponível em: <http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Trabalho?id=6894> Acesso em: 28 jan. 2014.

SOARES, Michel dos Santos. Metodologias Ágeis: *Extreme Programming* e Scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas da Informação**. 2010. Disponível em: <http://revistas.facecla.com.br/index.php/reinfo/article/view/146/0 >Acesso em: 28 jan. 2014.

SOMMERVILLE, Ian. **Engenharia de software**. 8ª ed. São Paulo: Pearson Addison Wesley, 2008.

STAATS, Bradley R.; UPTON, David M. ***Lean Principles, Learning, and Software Production:*** *Evidence From Indian Software Services.* Harvard Business School, 2009.

VAZ, Fabiano Amorim; VANDERLEI, Igor M.; BARROS, Roberto S.M. **Java on Road:** Protótipos Exploratórios e Evolutivos de Alta Fidelidade. Centro de Informática. Universidade Federal de Pernambuco. 2011. Disponível em: <http://www.researchgate.net/publication/230867346_Java_on_Road_Prottipos_Exploratrios_e_Evolutivos_de_Alta_Fidelidade/file/9fcfd5058a1ae6f377.pdf> Acesso em: 26 dez. 2014

WOMACK, James; JONES, Daniel. **A mentalidade enxuta nas empresas**. Rio de Janeiro: Campus, 2004.

ZAVE, Pamela. ***Classification of Research Efforts in Requirements Engineering.*** ACM Computer Surveys, vol 29, n°4, 1997.

APÊNDICE

Questionário - A Utilização dos Princípios do *Lean Software Development* na Engenharia de Requisitos

Protocolo de Pesquisa

O objetivo do questionário é verificar a percepção dos profissionais da área de processos de desenvolvimento de software a respeito da utilização dos princípios do *Lean Software Development* na Engenharia de Requisitos.

Informações Importantes:

1. As respostas são de caráter sigiloso e confidencial.
2. Os dados da pesquisa serão utilizados exclusivamente para fins acadêmicos.

1. Porte da empresa onde atua.

- Microempresa - até 19 funcionários
- Pequena Empresa - até 49 funcionários
- Média Empresa - até 99 funcionários
- Grande Empresa - acima de 99 funcionários

2. Segmento onde a sua empresa atua.

- Indústria
- Comércio
- Serviços

3. Quanto tempo de experiência você possui na área de engenharia de requisitos de software?

- 0 a 1 ano
- 1 a 3 anos
- 3 a 5 anos
- 5 a 10 anos
- Acima de 10 anos

4. A prototipação é utilizada frequentemente nos projetos de desenvolvimento de software com o intuito de auxiliar tanto o analista de requisitos quanto o usuário a entender melhor as funcionalidades, a forma como o sistema irá funcionar. Entre os vários tipos de prototipação destaca-se a prototipação funcional (apresenta melhor iteração junto ao usuário, clique de botão, anexar arquivos, preenchimento de campos) e que pode ser reaproveitada no desenvolvimento. Com que frequência você utiliza a prototipação funcional e a reaproveita no desenvolvimento?

- Até 80% do tempo
- Até 60% do tempo

- Até 40% do tempo
- Até 20% do tempo
- Nunca

5. De acordo com Mary e Tom Poppendieck, um dos princípios do *Lean Software Development* é: Construir certo da primeira vez, que consiste em realizar todo o esforço necessário para que a primeira entrega tenha a menor quantidade de erros possível. Em seus projetos, com qual frequência você utilizou essa linha de raciocínio?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

6. Um projeto complexo e composto de várias fases está sob o seu comando e para facilitar o gerenciamento e a interação entre a equipe, e para facilitar o gerenciamento das atividades do projeto, você decide em conjunto com a equipe realizar pequenas e rápidas entregas ao cliente, de forma que ele receba primeiro os itens que mais agregam valor ao seu processo. Com qual frequência você utiliza essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

7. O *Brainstorming* é uma prática onde os participantes são estimulados (de maneira organizada) a expressar suas ideias e sugestões a respeito de um determinado assunto de maneira bem livre e inicialmente sem se preocupar com a lógica entre as ideias e, a partir desse conjunto de ideias, as informações são organizadas e compartilhadas com todos os membros da equipe que podem opinar e sugerir mudanças. Essa prática pode ser utilizada na etapa inicial de elicitação de requisitos pois traz de uma só vez o ponto de vista de diversos *stakeholders*. Com que frequência você utilizou essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

8. Um dos pilares do *Lean Software Development* e de toda a filosofia Lean é a eliminação de desperdício nos processos; para o *Lean Software* qualquer coisa que é desenvolvida e que não agregue valor ao cliente deve ser considerada como desperdício e qualquer atraso que impeça o cliente de obter valor também é considerado desperdício. Quando se trata de desperdício em processos uma das principais causas é a falta de conhecimento sobre o que cada um faz em cada uma das etapas do processo. Os resultados do *brainstorming* podem ajudar no esclarecimento do funcionamento do processo e, por consequência, na identificação dos desperdícios. Com que frequência você utiliza o resultados do *brainstorming* para a identificação do desperdício?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

9. Para o *Lean Software Development* a otimização do todo consiste em otimizar todo o processo do recebimento da demanda até a implantação do software no cliente, incluindo os processos do cliente. Uma das formas de realizar essa otimização é a realização do *brainstorming* onde todos os envolvidos são incentivados a colocar o seu ponto de vista e todos tomam conhecimento do processo, o que oferece oportunidades de otimização. O *brainstorming* pode ser utilizado na etapa de elicitação dos requisitos, e essa prática auxilia na identificação dos stakeholders e da forma como cada um enxerga o projeto como um todo. Na etapa de análise e elicitação de requisitos com que frequência você utilizou essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

10. De acordo com Mary e Tom Poppendieck um dos princípios (4) refere-se a adiar ao máximo possível uma decisão importante dentro do projeto e, sempre que possível, tomar decisões que sejam facilmente reversíveis. A utilização da técnica de entrevista para realizar a etapa de validação dos requisitos (o último passo antes do inicio do desenvolvimento) e nesse momento tomar as decisões de alta complexidade ou que geraram muitas dúvidas ao longo de todo o processo e realizar a tomada de decisão em conjunto com o usuário e com máximo de informação possível. Com que frequência você utiliza essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo

- Até 20% do tempo
- Nunca

11. A etnografia é uma técnica que consiste na observação do usuário em seu local de trabalho, na execução e o acompanhamento das atividades em tempo real e tomando nota das atividades realizadas. Essa técnica é bastante eficiente para o levantamento de requisitos, por fornecer a oportunidade de análise da interação entre os usuários e extrair a informação diretamente da fonte. Durante o seu tempo de atuação na área de análise de requisitos com que frequência você utilizou essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

12. Para o *Lean Software Development* respeitar pessoas consiste em respeitar o conhecimento de cada membro e fazer com que todos compartilhem do planejamento e das responsabilidades assumidas ao longo do processo. Outro aspecto importante desse princípio consiste em impedir que interferências externas prejudiquem o trabalho da equipe responsável, ou seja, proteger a equipe de agentes externos que interfiram nas atividades. Uma das formas mais comuns de proteger a equipe especialmente de desenvolvimento consiste em nomear um PO (*Product Owner*) que deve receber e filtrar todas as demandas e fazer a ponte entre a equipe de desenvolvimento e o cliente. Com que frequência você utiliza essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca

13. Criar conhecimento é um dos princípios do *Lean Software Development:* este entende que o software não é um amontoado de documentações e procedimentos e sim um organismo vivo que cresce e evolui a medida que o tempo passa, estando constantemente agregando e criando valor. Ao realizar o desenho de um software é importante não criar processos que engessem o sistema. Nos projetos nos quais você atua ou atuou a sua preocupação com esse aspecto ocorreu com que frequência?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo

- Até 20% do tempo
- Nunca

14. O JAD (*Joint Application Design*) é uma técnica que permite um maior alinhamento e cooperação entre a equipe de desenvolvimento e os usuários, por meio de uma técnica de coleta e entendimento de requisitos. O JAD ajuda na captura e organização do conhecimento a respeito do software a ser desenvolvido. Com que frequência você utiliza essa técnica?

- Até 80% do tempo
- Até 60% do tempo
- Até 40% do tempo
- Até 20% do tempo
- Nunca