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

RODRIGO DA SILVA PINTO

A FILOSOFIA LEAN E SUA APLICAÇÃO NO DESENVOLVIMENTO DE SOFTWARE: UM CASO DE SUCESSO

São Paulo
Abril/2016

RODRIGO DA SILVA PINTO

A FILOSOFIA LEAN E SUA APLICAÇÃO NO DESENVOLVIMENTO DE SOFTWARE: UM CASO DE SUCESSO

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
Abril/2016

Pinto, Rodrigo da Silva

P659f A filosofia lean e sua aplicação no desenvolvimento de software: um caso de sucesso / Rodrigo da Silva Pinto. – São Paulo : CEETEPS, 2016.

112 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, 2016.

1. Lean software development. 2. Lean thinking. 3. Kanban. I. Azevedo, Marilia Macorin de. II. Centro Estadual de Educação Tecnológica Paula Souza. III. Título.

RODRIGO DA SILVA PINTO

A FILOSOFIA LEAN E SUA APLICAÇÃO NO DESENVOLVIMENTO DE SOFTWARE: UM CASO DE SUCESSO

Profa. Dra. Marilia Macorin de Azevedo

Prof. Dr. Antonio Cesar Galhardi

Prof. Dr. Mauro de Mesquita Spínola

São Paulo, 27 de abril de 2016

AGRADECIMENTOS

Muitas pessoas cooperaram direta e indiretamente para a realização desse trabalho.

Gostaria agradecer a minha orientadora Profa. Dra. Marilia Macorin de Azevedo pelas horas empenhadas em aulas, reuniões, apresentações e revisões. Sua parceria foi fundamental para a produção dos artigos, dessa dissertação e o principal, agregaram grande conhecimento.

Aos membros da banca, Prof. Dr. Antonio Cesar Galhardi e Prof. Dr. Mauro de Mesquita Spínola. As recomendações dadas pelos senhores demonstraram competência e perícia no assunto, sendo de grande valia na composição desta publicação.

Aos gestores da empresa UOL SA: Márcio Drumond Araújo, Daniele Gemignani e Renato Silveira pela autorização na divulgação dos dados contidos nesse estudo e pelo horário flexível que me permitiu atender às aulas do mestrado. Isso foi crucial!

Aos amigos Romeu Ivolela, Carlos Idoeta, membros do ShoppingUOL e principalmente ao time Riviera: obrigado por me ensinarem em todo esse tempo!

Aos colegas e membros da Unidade de Pós-graduação do Centro Paula Souza pelo apoio nessa jornada.

E por fim, à minha família: meus filhos Gabriel e Rafaela, meus pais Hélvio e Eclair, e minha esposa Michelle. Vocês são meu apoio, obrigado pelos incentivos e pela paciência.

A todos, meus sinceros agradecimentos.

It is not enough to do your best; you must
know what to do, and then do your best.
(W. Edwards Deming)

RESUMO

PINTO, R. S. **A filosofia lean e sua aplicação no desenvolvimento de software: um caso de sucesso** 112 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 identificar, por meio da pesquisa empírica, como a Filosofia *Lean* pode ser incorporada ao desenvolvimento de software. A metodologia usada é a Pesquisa-Ação e o universo de pesquisa, um time de desenvolvedores alocado na empresa UOL SA. Além de definir os princípios e práticas *Lean* vindos da indústria de manufatura, o estudo aponta como podem ser praticados nas atividades cotidianas de desenvolvimento de um time. Por fim, a pesquisa apresenta uma análise qualitativa dos dados, evidenciando o aumento da eficiência no processo, embasado em conceitos empíricos desenvolvidos por décadas na indústria de manufatura japonesa e que demonstraram-se aplicáveis à engenharia de software.

Palavras-chave: Lean Software Development. Lean Thinking. Kanban

ABSTRACT

PINTO, R. S. **The lean philosophy and its application in software development: a success story** 112 f. Dissertation (Mestrado Profissional em Gestão e Tecnologia em Sistemas Produtivos). Centro Estadual de Educação Tecnológica Paula Souza, São Paulo, 2015.

This study aims to identify, through empirical research, how the Lean Philosophy can be applied to software development. The methodology used is the Action Research and the universe of research is a team of developers allocated in the company UOL SA. In addition to defining the Lean principles and practices, which come from the manufacturing industry, the study shows how they can be practiced in the daily activities of a development team. Finally, the research presents a qualitative analysis of the data, showing the increase of process efficiency, based on empirical concepts developed for decades in the Japanese manufacturing industry and that proved to be applicable to software engineering.

Keywords: Lean Software Development. Lean Thinking. Kanban

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE SIGLAS

CFD	Cumulative Flow Diagram
CMMI	Capability Maturity Model Integration
DMS	Daily Management System
IC	Integração Contínua
IMVP	International Motor Vehicle Program
JIT	Just-in-time
LSD	Lean Software Development
MIT	Massachusetts Institute of Technology
PDCA	Plan-Do-Check-Act
P&D	Pesquisa e Desenvolvimento
PU	Processo Unificado
QCO	Quick Change Over
ROI	Return Over Investment
SMED	Single Minute Exchange of Dies
SOPK	System of Profound Knowledge
TDD	Test Driven Development
TOC	Theory of Constraints
TPM	Total Productive Maintenance
TPS	Toyota Production System
VSM	Value Stream Map
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

A Engenharia de Software surgiu em meados da década de 1960 com o objetivo de prover um método estruturado voltado à especificação, desenvolvimento e manutenção de sistemas de software, com uso de tecnologias, práticas gerenciais e outras disciplinas, visando organização, produtividade e qualidade (MAGELA, 2006; PRESSMAN, 2006).

O surgimento dessa engenharia se fez em resposta a "crise do software", termo cunhado por Edsger Dijkstra, que pretendia expressar as dificuldades vivenciadas pelo desenvolvimento de software frente ao rápido crescimento da demanda, da complexidade dos problemas a serem resolvidos e da inexistência de técnicas e métodos estruturados para esse fim (DIJKSTRA, 1972; MAFFEO e SILVA, 1992).

Num período conceptual como aquele, é comum as novas áreas do conhecimento sofrerem influência de outras mais desenvolvidas. Nesse caso, os primeiros passos da Engenharia de Software foram baseados em conceitos e tecnicalidades da Engenharia da Produção, sendo especial a influência do setor automotivo, que era um ramo estabelecido por décadas de um processo evolucionário contínuo (POPPENDIECK e POPPENDIECK, 2003).

O setor automotivo contribuiu de forma significativa para o aprimoramento do sistema produtivo e da indústria como um todo. Historicamente, sua evolução possui dois momentos marcantes, ambos pautados pela melhoria disruptiva do processo fabril. O primeiro aconteceu no início do século XX com Henry Ford e a Produção em Massa, enquanto o segundo ganhou notoriedade a partir de 1950, com a montadora japonesa Toyota e sua filosofia gerencial, posteriormente batizada de *Lean* (KRAFCIK, 1988).

Lean é um modelo gerencial aplicado por meio de princípios e técnicas operacionais, que tem como objetivo a redução do desperdício, a melhoria da qualidade e a maximização do valor entregue ao cliente. Em sua essência, é uma filosofia orientada à eficiência e eficácia de processos, centrada em criar mais valor com menos trabalho (WOMACK e JONES, 2010).

O principal fator para o sucesso do *Lean* na manufatura foi combinar as vantagens da Produção Artesanal e em Massa ao mesmo tempo em que evitava seus pontos fracos. A produção artesanal possuía uma grande capacidade de customização e adaptação porém a um custo de execução elevado. Isso era devido a reduzida padronização dos produtos e do processo fabril. Por sua vez, a Produção em Massa reduziu o custo da produção enrijecendo a linha de montagem e as opções aos clientes. Nesse quesito, o *Lean* propunha técnicas inovadoras que possibilitavam produzir grande variedade, em grandes volumes, a um preço competitivo (WOMACK, JONES e ROOS, 1991).

Os benefícios colhidos pela implantação do *Lean* na manufatura foram tão expressivos que sua influência extrapolou os limites da indústria automotiva. Como vetor de inovação, produtividade e qualidade, esse ferramental foi aplicado não somente em segmentos industriais, mas também em setores como hospitalar, escritórios, serviços, chegando ao desenvolvimento de software (KEYTE e LOCHER, 2004).

Na década de 1990, a indústria de software passava novamente por uma crise. Enquanto em 1960 a queixa era a falta de método, em 1990 os métodos tradicionais de desenvolvimento estavam sendo contestados quanto a sua efetividade devido aos recorrentes insucessos (MELO, *et al.*, 2013; STANDISH GROUP, 1995).

Como alternativa ao modelo e métodos vigentes, num mesmo momento em que se expandia uma nova categoria chamada de Métodos Ágeis, surgiram os primeiros relatos da criação e aplicação da Filosofia *Lean* no desenvolvimento de sistemas de informação. Assim, por terem surgido no mesmo movimento e serem constituídos de princípios em comum, métodos *Lean* e Ágeis são considerados parte da mesma família (VILKKI e ERDOGMUS, 2012; HIGHSMITH, 2002).

No início, esses métodos alternativos sofreram grande resistência por questionarem muitas práticas dos métodos tradicionais. Por exemplo, observando princípios como foco no valor ao cliente, lotes pequenos e eliminação de desperdício, métodos *Lean* e Ágeis enfatizavam mais a comunicação, pequenas iterações e adaptabilidade, do que uso extensivo de documentação, grande hierarquia de comunicação e planejamento rígido das atividades (BECK, *et al.*, 2001).

Com o decorrer dos anos, projetos *Lean* e Ágeis vêm apresentando bons resultados e conquistando mais adeptos (LO GIUDICE, *et al.*, 2015; LO GIUDICE, KISKER e ANGEL, 2014; VERSION ONE, 2014; STANDISH GROUP, 2014; WEST, *et al.*, 2010). De igual modo, esses métodos estão capturando a atenção das comunidades científicas, haja vista a quantidade crescente de periódicos e congressos nos quais grande número de publicações são encontradas abordando o assunto (JONSSON, 2012; PERNSTÅL, FELDT e GORSCHEK, 2013; RODRÍGUEZ, 2013).

1.1 Questão de pesquisa

Como pode ser visto, a Filosofia *Lean* é advinda da indústria, do conhecimento empírico/tácito de empreendedores e colaboradores, não havendo, necessariamente, um embasamento teórico-científico. Assim, é importante compor um estudo sobre como princípios e práticas da indústria automotiva podem ser aplicados no desenvolvimento de software. Dessa forma, nesta dissertação busca-se responder a seguinte questão de pesquisa:

Como a Filosofia *Lean* pode ser aplicada em um projeto de desenvolvimento de software?

1.2 Objetivos

O objetivo geral do trabalho é identificar, por meio de um estudo empírico, como a Filosofia *Lean* pode ser aplicada ao desenvolvimento de software.

Objetivos específicos:

- Descrever a Filosofia *Lean*, sua definição, origem, princípios e ferramentas
- Posicionar o *Lean* na evolução histórica dos métodos de desenvolvimento de Sistemas de Informação
- Descrever a aplicação da Filosofia *Lean* na engenharia de software por meio de métodos já conhecidos: Lean Software Development e Método Kanban
- Por meio do estudo empírico, apresentar como podem ser aplicadas as ferramentas abstraídas do *Lean Manufacturing* em um projeto de desenvolvimento de software

1.3 Justificativa

Os fatores que motivaram essa pesquisa estão na importância e relevância do assunto em relação à quantidade e qualidade das publicações científicas do gênero.

Segundo relatório da Associação Brasileira das Empresas de Software (2015), o mercado de tecnologia da informação, que inclui hardware, software e serviços, movimentou em todo o mundo 2,09 trilhões de dólares em 2014. No Brasil essa parcela foi de 60 bilhões de dólares, o que representou 2,6% do PIB brasileiro, um resultado superior aos números do ano anterior. Deste valor, 11,2 bilhões vieram do mercado de software e 13,9 bilhões do mercado de serviços, sendo a soma dos dois segmentos superior a 40% do mercado total de TI, um forte indicativo da passagem do Brasil para o grupo de economias que privilegiam o desenvolvimento de soluções e sistemas (ABES, 2015).

É um consenso entre diversos autores que os métodos gerenciais utilizados na produção de sistemas de informação são vitais para o sucesso do projeto (SOMMERVILLE, 2007; PRESSMAN, 2006; APPLEGATE, MCFARLAN e MCKENNEY, 1996). Haja vista a grandeza desse mercado, o aumento tanto dos investimentos no setor quanto da quantidade de sistemas sendo produzidos, cresce a relevância do tópico em pesquisa tanto por fatores acadêmicos quanto de mercado.

Seguindo a mesma tendência de crescimento, pesquisas de mercado de diversos institutos apontaram o aumento na utilização dos métodos *Lean* para desenvolvimento de software[*] (LO GIUDICE, *et al.*, 2015; LO GIUDICE, KISKER e ANGEL, 2014; VERSION ONE, 2014; STANDISH GROUP, 2014; WEST, *et al.*, 2010). Da mesma forma, no âmbito acadêmico, as revisões da literatura publicadas por Jonsson (2012) e Pernstal (2013) indicaram o interesse da comunidade científica, assim como apresentaram uma lacuna de pesquisa no tópico desenvolvimento de software *Lean.*

A Tabela 1 mostra o levantamento bibliométrico feito em janeiro de 2016 na base de publicações Web of Science (2016). A busca utilizou os termos ***lean*** e **software** nos campos título, resumo e palavra-chave. O resultado foi filtrado pelo domínio de pesquisa SCIENCE TECHNOLOGY, pela área de pesquisa COMPUTER SCIENCE, o tipo de publicação

---

[*] Entende-se Métodos *Lean:* Lean Software Development, Kanban e métodos híbridos que contenham práticas centrais do *Lean* como o Scrumban.

ARTICLE e o intervalo de data de publicação nos últimos 20 anos (1995-2015) e também nos últimos 5 anos (2010-2015).

Tabela 1 – Publicações e citações sobre Lean Software nos últimos 20 anos

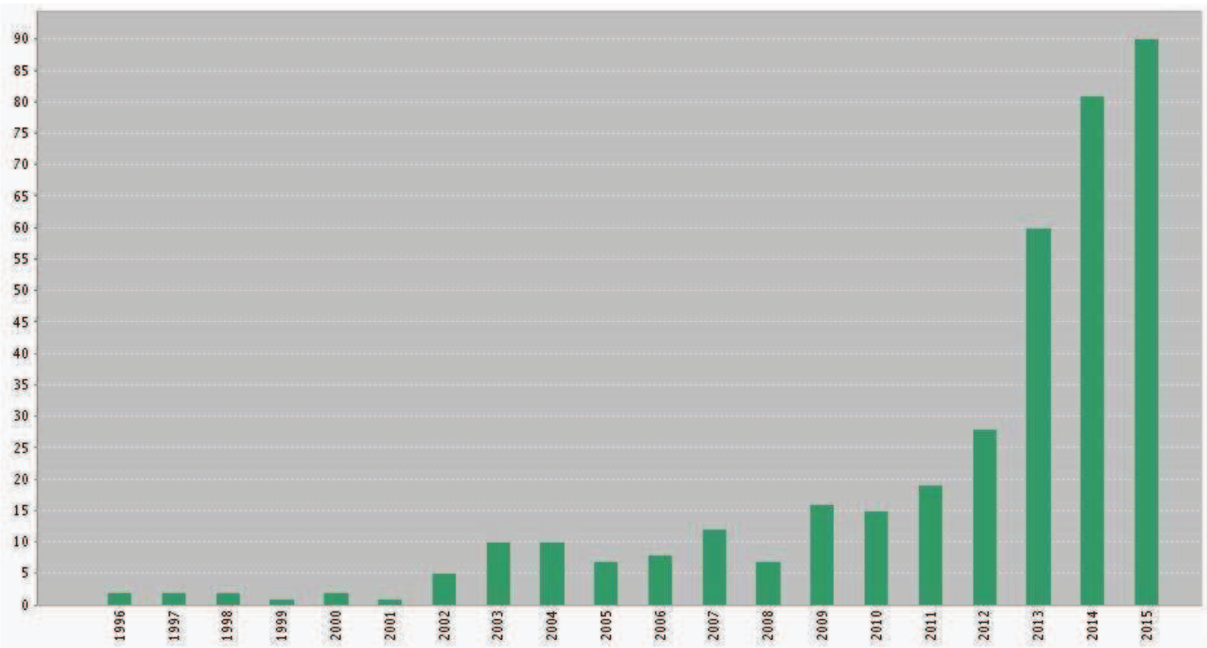
	1995 - 2015	**2010 - 2015**
Artigos Publicados	61	37
Número de citações	378	293
Artigos que fizeram a citação	341	256

Fonte: adaptado de Web of Science (THOMSON REUTERS, 2016)

Dos 172 artigos encontrados, 111 foram descartados após análise do título e do resumo. Tais pesquisas estavam arroladas nas áreas de saúde, manufatura e computação, possuíam os termos software e *lean,* porém não estavam associadas ao lean software.

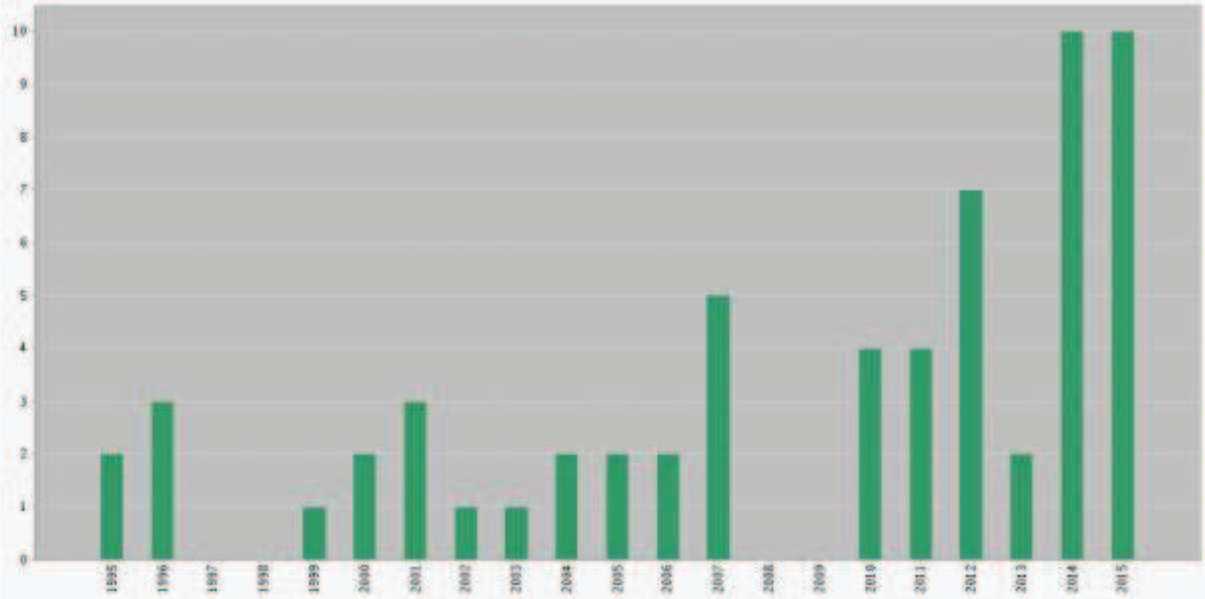
Dessa forma, aferiu-se que de 1995 a 2015 foram publicados 61 artigos com a temática em questão e que os mesmos foram referenciados por 341 publicações, num total de 378 citações. Ademais, atestou-se que mais de 60% desse total de publicações e citações estão concentradas nos últimos 5 anos (37 artigos e 293 citações), o que pode ser visto nas Figura 1 e Figura 2.

Figura 1 – Publicações sobre Lean Software nos últimos 20 anos



Fonte: adaptado de Web of Science (THOMSON REUTERS, 2016)

Figura 2 – Citações sobre Lean Software nos últimos 20 anos



Fonte: adaptado de Web of Science (THOMSON REUTERS, 2016)

Baseado no estudo bibliométrico feito por Jonsson (2012), usou-se também o Google Acadêmico para contabilizar o número de citações das principais publicações sobre *Lean* e Lean Software Development utilizadas nessa pesquisa, o que incluiu os principais livros do gênero, assim como um artigo e uma dissertação. Como pode ser visto nas Tabela 2 e Tabela **3**, a quantidade de citações dessas publicações praticamente duplicou nos últimos três anos.

Tabela 2 – Citações sobre *Lean* e Lean Software no Google Acadêmico em 2012

Referência	**Citações em 2012**
(WOMACK, JONES e ROOS, 1991)	8942
(WOMACK e JONES, 2010)	3694
(LIKER, 2004)	1723
(OHNO, 1988)	2588
(POPPENDIECK e POPPENDIECK, 2003)	525
(POPPENDIECK e POPPENDIECK, 2006)	183
(POPPENDIECK e POPPENDIECK, 2010)	27
(ANDERSON, 2010)	58
(COPLIEN e BJORNVIG, 2010)	27
(MORGAN, 1998)	4

Fonte: adaptado de Jonsson (JONSSON, 2012, p. 3)

Tabela 3 – Variação nas citações sobre *Lean* e Lean Software no Google Acadêmico 2012/2015

Referência	Citações em 2015	Variação (2012/2015)
(WOMACK, JONES e ROOS, 1991)	12772	+42,83%
(WOMACK e JONES, 2010)	6530	+76,77%
(LIKER, 2004)	3650	+111,84%
(OHNO, 1988)	4249	+64,18%
(POPPENDIECK e POPPENDIECK, 2003)	923	+75,81%
(POPPENDIECK e POPPENDIECK, 2006)	343	+87,43%
(POPPENDIECK e POPPENDIECK, 2010)	72	+166,67%
(ANDERSON, 2010)	272	+368,97%
(COPLIEN e BJORNVIG, 2010)	64	+137,04%
(MORGAN, 1998)	11	+175,00%

Fonte: elaborado pelo autor

Dessa forma, nota-se o interesse crescente intensificado nos últimos anos, tanto do mercado quanto da comunidade científica, por um assunto que movimenta grande parte da economia mundial.

A importância deste trabalho consiste em, além de corroborar com a afirmação que a Filosofia *Lean* é aplicável e beneficia o desenvolvimento de software, pretende-se demonstrar como isso é feito mediante exemplos de métodos conhecidos extraídos de referências bibliográficas e de um estudo empírico no qual pode-se visualizar detalhadamente os princípios e práticas estudados, assim como os resultados dessa aplicação.

1.4 Organização do Trabalho

O presente trabalho é composto de 5 capítulos. O **capítulo 1** apresenta uma introdução, na qual há a contextualização do tema, seguido da questão de pesquisa, ou seja, a pergunta que o estudo almeja responder, os objetivos geral e específicos, a justificativa e, por fim, como o trabalho está estruturado.

O **capítulo 2** é a fundamentação teórica e apresenta o embasamento teórico-científico do estudo. Esse capítulo se divide em duas seções: a primeira descreve o *Lean* no ambiente da Engenharia da Produção (Lean Manufacturing), enquanto a segunda está voltada à Engenharia de Software e a implementação do Lean neste contexto (Lean Software Development).

O **capítulo 3** contém a metodologia utilizada na elaboração do estudo, ou seja, apresenta a classificação da pesquisa científica, as etapas do seu planejamento, o detalhamento do processo e os canais de comunicação utilizados pelo pesquisador.

O **capítulo 4** é um estudo de caso que relata como uma célula da empresa UOL S/A, empresa líder de seu segmento, utilizou os métodos explanados no capítulo 2, assim como a aplicação customizada dos princípios *Lean* no desenvolvimento do sistema de informação para um projeto interno. Esse capítulo retrata o cenário do estudo, o processo utilizado nas atividades diárias, os problemas vivenciados, as soluções *Lean* praticadas e os resultados obtidos.

Por fim, o **capítulo 5** é composto pelas considerações finais do trabalho e uma proposta de sequência para a pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica está dividida em duas seções. A primeira seção desse capítulo está relacionada à Engenharia da Produção, traz um panorama histórico e apresenta os princípios e as práticas do *Lean,* filosofia de gerenciamento da produção que revolucionou a indústria e em especial o setor automotivo.

A segunda seção relata as inovações e melhorias trazidas pela Filosofia *Lean* e como foram aplicadas no desenvolvimento de software. Para isso, são expostos dois métodos de desenvolvimento *Lean*: Lean Software Development conhecido como a primeira aplicação do *Lean* no desenvolvimento de sistemas; e o Método Kanban que segundo pesquisa foi o método de software que mais cresceu em utilização no ano de 2014 (VERSION ONE, 2014; LO GIUDICE, KISKER e ANGEL, 2014).

2.1 Engenharia da Produção

Em meados de 1900 quando se desejava adquirir um automóvel, era necessário encomendá-lo. A força de trabalho empregada na época era altamente qualificada em todas as fases da produção: design, operações de máquinas, ferramental e montagem. A grande maioria dos trabalhadores era formada por meio de um processo de aprendizado interno, obtendo destreza em um conjunto completo de habilidades manuais e artesanais.

As organizações eram extremamente descentralizadas embora concentradas em uma única localidade. A maioria das peças e muito do design do veículo vinham de pequenas oficinas mecânicas fornecedoras. O sistema era coordenado pela empresa montadora em contato direto com todos os envolvidos (clientes, empregadores e fornecedores). As máquinas e ferramentas utilizadas eram de propósito geral e executavam corte, perfuração, moagem soldagem e outras operações em matérias primas como metal, madeira e borracha.

O sistema possuía um volume de produção muito baixo (1000 ou menos automóveis por ano), e poucos (cinquenta ou menos) eram construídos com o mesmo design. Mesmo entre esses, nenhum era exatamente igual pois as técnicas artesanais inerentes ao processo acabavam gerando variações no produto final (WOMACK, JONES e ROOS, 1991).

Por volta de 1905, quando Henry Ford iniciava seu trabalho, a indústria artesanal havia encontrado um platô que não conseguia superar. As limitações estavam relacionadas à:

- produtividade, pois não conseguia suprir as demandas em tempo hábil;
- qualidade, pois a confiabilidade e durabilidade dos produtos eram questionáveis;
- inovação, uma vez que os pequenos artesãos não conseguiam produzir e investir em pesquisas e no desenvolvimento tecnológico.

Nesse momento, Ford conseguiu superar os problemas referentes à produção artesanal com um novo sistema de produção, que empregava técnicas de redução drástica de custos e aumento da qualidade do produto. A esse sistema inovador ele deu o nome de Produção em Massa (WOMACK, JONES e ROOS, 1991).

O segredo da Produção em Massa não era, como muitos acreditavam, a movimentação das partes numa linha de montagem contínua. Ao invés disso, a permutabilidade completa e consistente das peças e a simplicidade de montá-las entre si foram as inovações na fabricação que tornaram a linha de montagem possível, sendo um grande diferencial competitivo (WOMACK, JONES e ROOS, 1991).

As peças intercambiáveis, por sua vez, foram desenvolvidas graças ao aparecimento de novas ferramentas de precisão para o corte e prensa de metais. Ford reduziu drasticamente o tempo de *setup* por meio do desenvolvimento de máquinas de finalidade específica, que faziam uma função de cada vez. Um trabalhador não tão qualificado apenas posicionava a peça e apertava um botão em uma máquina dedicada para executar a tarefa desejada.

Quanto a disposição e atuação das pessoas, existia nas plantas da Ford a figura central do engenheiro chefe, que planejava todo funcionamento da fábrica. Ele dizia como as peças se encaixariam, as tarefas que cada montador deveria executar, os trabalhadores que deveriam suprir a linha de produção com peças, os ajudantes gerais que estariam periodicamente arrumando e limpando as áreas de trabalho, os reparadores que consertavam e repunham as ferramentas dos montadores, os profissionais da qualidade que revisavam todo o trabalho realizado, e os retificadores que refaziam o que não fora aprovado pela qualidade. Com essa segmentação por funções específicas, pouco tempo de treinamento era necessário para a formação de um novo trabalhador (WOMACK, JONES e ROOS, 1991).

Assim, em conjunto, maquinário específico, permutabilidade, simplicidade e facilidade de conectar as peças deram a Ford uma tremenda vantagem competitiva sobre os concorrentes, mesmo em um sistema com deficiências. Entre 1914 e 1924, essas inovações industriais praticamente acabaram com a produção artesanal de veículos automotores, fazendo com que o número de empresas norte americanas de automóveis caísse de mais de 100 para pouco mais de uma dezena. Dentre elas Ford, General Motors e Chrysler representavam mais de 90% de todo o mercado (WOMACK, JONES e ROOS, 1991).

A partir da década de 1950, a indústria automotiva japonesa e em especial a montadora Toyota iniciou uma expansão global movida por resultados muito superiores aos apresentados por montadoras europeias e americanas, sendo reconhecida pela maioria dos especialistas como o produtor de veículos mais eficiente e de maior qualidade em todo o mundo, com desempenho e números que progrediam para superar os americanos em escala mundial (WOMACK, JONES e ROOS, 1991).

Ao fim dá década de 1980, pesquisas acadêmicas atestaram algo até então imperceptível aos gestores ocidentais. Muitos pensavam que o sucesso da Toyota estava atrelado ao mercado consumidor japonês, incentivos governamentais, fiscais ou puramente a automação. Todavia, esses estudos comprovaram que existia por trás dessa empresa uma filosofia gerencial e um sistema de produção que era mais eficiente e eficaz que os demais, sendo o agente propulsor dos grandes resultados (KRAFCIK, 1988; WOMACK e JONES, 2010).

2.1.1 História da Toyota

A história da Toyota começou com Sakichi Toyoda, um pensador e inventor que cresceu no fim do século XIX em uma comunidade rural nas proximidades de Nagoya. Naquele tempo, a tecelagem era a maior indústria do Japão e o governo desejoso de promover seu crescimento, encorajou o desenvolvimento da indústria a partir da criação de pequenas oficinas. A Toyota iniciou suas atividades no ano de 1896 e seus primeiros produtos foram teares mecânicos (MASS e ROBERTSON, 1996).

O espírito inventor de Sakichi representou um diferencial desde os primeiros anos. Sua primeira inovação foi um mecanismo especial que parava automaticamente o tear assim que o fio se quebrasse. Esse invento foi popularmente batizado de "tear mecânico a prova de erros" e tipificou em grande forma a disposição da empresa em desenvolver seus produtos.

No início dos anos 1920, Sakichi incluiu seu filho Kiichiro Toyoda no conselho administrativo da empresa. Uma década depois, a companhia mudou completamente seu mercado migrando para o setor automotivo com o objetivo de suprir a demanda de caminhões para uso militar. Kiichiro foi enviado à prestigiada Universidade Imperial de Tokio para estudar engenharia mecânica com enfoque na tecnologia de motores (REINGOLD, 1999).

Os membros da família Toyoda cresceram dentro de uma filosofia semelhante, aprendendo na prática, com um espírito inovador e entendendo que a empresa deve contribuir com a sociedade. Além disso, todos desejavam criar uma companhia com visão de longo prazo. Depois de Kiichiro Toyoda, outro membro da família que liderou e influenciou em grande forma foi Eiji Toyoda, sobrinho de Sakichi. Eiji Toyoda também estudou engenharia mecânica na Universidade Imperial de Tokio em 1933. Assim que se formou, seu primo Kiichiro lhe incumbiu de criar um "centro de pesquisa" em uma garagem no distrito de Shibaura (TOYODA, 1987). Eiji trabalhou sozinho durante meses, depois constituiu um time de 10 colaboradores, estabeleceu parcerias com diversos fornecedores da região e criou protótipos. E assim foi a fase embrionária da primeira fábrica da Toyota com a finalidade única de produção de veículos, construída na cidade de Nagoya (FUJIMOTO, 1999).

Por volta de 1930 Kiichiro liderou um grupo em visita as fábricas da Ford em Michigan com o objetivo de aprender mais sobre a indústria automotiva americana e, de igual modo, trazer à recém-criada Toyota, ferramentas e técnicas que aperfeiçoassem seu processo produtivo. Anos se passaram e após a Segunda Guerra, em 1950, Eiji e um grupo de gerentes realizaram outra excursão de 12 semanas por diversas plantas da Ford e GM com o intuito de assimilar as inovações desenvolvidas por seus competidores nesse período. Porém, a comitiva foi surpreendida ao encontrar o mesmo cenário da primeira visita 20 anos antes (LIKER, 2004).

De fato eles estavam aprendendo sobre como não produzir um veículo. As mesmas técnicas da produção em massa estavam sendo implementadas: muitos equipamentos produzindo grandes quantidades de produtos que eram armazenados em estoques, para só depois serem movidos para outro departamento, onde grande maquinário processaria o mesmo e assim por diante. Eles perceberam que o princípio por trás da produção em massa estava na eficiência individual de cada etapa e não na exploração do todo.

Após as visitas às plantas automobilísticas americanas, os gestores da Toyota concluíram que o sistema de Ford não poderia ser aplicado à realidade do Japão pós Segunda Guerra Mundial (KATAYAMA, 2010), pois:

- O Sistema de Produção em Massa estava projetado para ganhos de escala, isto é, dependia da fabricação de poucos modelos em grande quantidade. No entanto, o mercado doméstico japonês era limitado e exigia uma variedade grande de modelos – carros de luxo para oficiais do governo, grandes caminhões de carga para as redes mercadistas, caminhões de pequeno porte visando pequenos agricultores, carros pequenos adequados às cidades japoneses muito populosas;
- A economia japonesa estava devastada pela guerra, em grande necessidade de capital e sua moeda estava enfraquecida. Isso fazia com que a aquisição de maquinário de última geração vindos do ocidente fosse descartada;
- O espaço territorial japonês era muito reduzido, contrastivo ao espaço de armazenamento abundante nas empresas americanas.

2.1.2 Taiichi Ohno

Um dos grandes idealizadores do que posteriormente foi chamado de Sistema de Produção Toyota (do inglês *Toyota Production System*, TPS), foi o engenheiro chefe da montadora por muitos anos, Taiichi Ohno.

Em sua obra, Ohno também expressou a necessidade de adaptar as práticas do Fordismo à realidade japonesa, ou os intentos de Sakichi Toyoda seriam frustrados. Além disso, nas visitas às fábricas da Ford em Detroit, percebeu que aquele sistema de produção estava repleto de *muda*, palavra japonesa utilizada para expressar o desperdício de tempo, material e esforço (OHNO, 1988).

Ele constatou que, teoricamente, nenhum dos especialistas acima do trabalhador da linha de montagem adicionava valor ao veículo que estava sendo produzido. Além do mais, achava que os trabalhadores poderiam desempenhar a maioria das funções de um especialista e com qualidade superior, pois possuíam um contato mais direto com o sistema de produção, ainda que os mesmos trabalhadores ocupassem papeis de menor destaque dentro da fábrica. Em contraste, nas fábricas ocidentais, gerentes diziam aos trabalhadores que eles só eram necessários pois a automação ainda não conseguira substituí-los (WOMACK, JONES e ROOS, 1991).

Assim que voltou das viagens à América, Taiichi Ohno iniciou uma série de experimentos. O primeiro passo foi organizar trabalhadores em times com um líder ao invés de um capataz. Cada time era responsável por um conjunto de etapas de montagem (uma parte da linha) e era permitido que se organizassem da melhor maneira possível para executar as operações. O líder trabalharia na linha de montagem, na coordenação do time e também ocuparia o lugar de algum trabalhador que faltasse (conceitos inovadores e não vistos na produção em massa).

Em seguida, destinou ao time as funções de organizar e limpar seu espaço, executar pequenos reparos em ferramentas e realizar controle de qualidade. E finalmente, como último passo, depois das equipes estarem operando sem problemas, ele passou a reservar periodicamente um período para que o time sugerisse formas de melhorar o processo. Essa prática de melhoria contínua e incremental era chamada de *kaizen*, palavra japonesa que significa “mudança para melhor”.

Ao falar de retrabalho, os conceitos e ações de Ohno foram realmente revolucionários. Ele percebeu que a prática da produção em massa de permitir a passagem dos defeitos a frente com a justificativa de manter a linha ativa, causava a multiplicação de problemas de forma interminável. Os trabalhadores pensavam que os erros seriam reparados de qualquer forma na etapa de qualidade, mas ações que parassem a linha de produção seriam motivo de punição. Como a falha só seria descoberta no fim da linha, um número alto de veículos com defeitos similares era produzido até que o problema fosse notado (WOMACK, JONES e ROOS, 1991).

Em contraste com as fábricas de produção em massa, onde parar a linha era responsabilidade do gerente de linha sênior, Ohno posicionou uma corda acima de todas as estações de trabalho e instruiu os trabalhadores a parar a linha de produção imediatamente se ocorresse um problema que eles não pudessem resolver. Dessa forma, todo o time trabalharia para solucionar o mais rápido possível aquele problema.

Taiichi Ohno ainda foi além. Nas fábricas de produção em massa, os problemas eram tratados como eventos aleatórios. A ideia era simplesmente reparar cada erro e esperar que ele não ocorresse mais. Ohno por sua vez, instituiu um sistema de resolução de problemas chamado de “5 porquês” (5W). Os trabalhadores da linha eram ensinados a rastrear sistematicamente cada erro de volta a sua causa inicial (chamada causa raiz), perguntando o porquê assim que cada camada do problema fosse descoberta, de forma a elaborar a correção e uma ação para que aquele erro não acontecesse mais. Assim que Ohno experimentou essas ideias, sua linha de produção começou a parar a todo o instante. Todavia, assim que os times ganharam experiência identificando e rastreando os problemas à sua causa raiz, o número de erros começou a cair drasticamente (OHNO, 1988).

Hoje, nas fábricas da Toyota, onde qualquer trabalhador pode parar a linha, a mesma roda ininterrupta em quase 100% do tempo. Em contraste, nas fábricas de produção em massa, somente o gerente da linha pode pará-la, e ainda assim o fazem constantemente. Isso ocorre não para retificar erros (pois eles só são feitos ao final), mas sim para lidar com suprimento de material e problemas de coordenação. Ainda mais impressionante foi o que ocorreu no fim da linha de montagem. Assim que o sistema de Taiichi Ohno acertou seu passo, a quantidade de retrabalho caiu continuamente e a qualidade dos veículos entregues foi maior (WOMACK, JONES e ROOS, 1991).

Ohno também desenvolveu uma nova forma de coordenar o fluxo das partes e peças dentro do sistema de suprimento no dia-a-dia da operação, utilizando um sistema chamado internamente de *kanban*. Sua ideia era simplesmente de converter o vasto grupo de fornecedores e sua própria fábrica em uma grande cadeia interconectada, em que peças só seriam produzidas para suprir uma demanda iminente do passo subsequente (OHNO, 1988).

Essa ideia simples era muito difícil de ser implementada pois eliminava praticamente todo o estoque, indicando que quando uma pequena porção do vasto sistema de produção falhava, todo o sistema vinha a parar. Na visão de Taiichi Ohno, o real intuito dessa prática era promover em cada membro da cadeia de produção o foco em antecipar problemas antes que eles se tornassem sérios o bastante para parar a linha de produção.

Um outro intento era promover a análise do processo de manufatura por meio dos olhos do cliente. A primeira pergunta a ser feita era: "o quê o cliente deseja desse processo?", seja ele um cliente interno ou externo. Isso fazia com que fosse definido o valor, observando o processo pelos olhos do cliente e separando as etapas geradoras e as etapas não geradoras de valor, que são os desperdícios.

Quanto a isso, a obra de Taiichi Ohno (1988) é um clássico pois apontou os 7 maiores desperdícios no desempenho das atividades diárias de produção. Posteriormente, Jeffrey Liker (2004) adicionou o oitavo desperdício. São eles:

1- Superprodução: significa produzir itens mesmo quando não há pedido, o que demanda mais funcionários, um lugar maior para armazenagem, custos com transporte e o dispêndio de toda a força produtiva sem razão para tal.

2- Espera: acontece quando trabalhadores têm de aguardar uma etapa do processo, uma ferramenta, fornecedor ou parte. Isso pode ocorrer por não terem trabalho dada falta de itens em estoque, pela espera por grandes lotes, por falhas ou manutenções em equipamentos ou por gargalos de processamento.

3- Transporte: deslocar peças, equipamentos e ferramentas entre as etapas do processo por longas distâncias cria ineficiência pois consome recursos e tempo.

4- Superprocessamento: ocorre pela existência de etapas que atuam sobre o produto sem a efetividade que a justifiquem. Isso pode ser consequência da má qualidade ou má utilização do ferramental, ou mesmo quando são fabricados produtos ou peças com mais qualidade do que o necessário.

5- Estoque: o excesso de matéria-prima, material em processo ou produto acabado causa o aumento do *lead time*, da obsolescência e custos com transporte, armazenamento; além de representar "dinheiro parado".

6- Movimentação: qualquer movimentação desnecessária dos trabalhadores durante o curso da produção, como procurar um material, se locomover para levar ou buscar uma ferramenta, é um desperdício.

7- Defeitos: produzir produtos ou partes defeituosas é o desperdício mais evidente da cadeia produtiva. O retrabalho por meio de reparos deve ser evitado a todo o custo.

8- Subutilização: não utilizar o potencial humano dos trabalhadores é um desperdício que pode ser comparado ao custo de oportunidade. A empresa está em desperdício se não explorar a criatividade, habilidades, ideias, melhorias e outras capacidades existentes em seus colaboradores.

Ohno considerava a superprodução o desperdício fundamental por ser o causador da maioria dos demais. Produzir mais do que o cliente deseja acarreta a criação de estoque, filas e atrasos nas etapas da operação. Outro problema decorrente de grandes lotes e estoques é o comportamento “relaxado” que reduz a motivação pela melhoria contínua. Por quê preocupar-se com manutenção preventiva nos equipamentos quando paradas não afetam imediatamente a linha de produção? Por quê preocupar-se com alguns erros de qualidade quando se pode descartar peças defeituosas? Pessoas tem dificuldade em perceber que, ao construir um sistema “sem folgas”, ficam evidentes os pontos que devem ser melhorados. Isso gera um ciclo virtuoso que não é explorado na Produção em Massa pois os grandes *buffers* mascaram as imperfeições do processo (LIKER, 2004).

Ao atacar os desperdícios do processo como baixa qualidade, produzir além no necessário e tempo de espera, Ohno diminuiu a quantidade de trabalho despendida na produção, sendo fator determinante no aumento de produtividade, como visto na Equação Simples da Produtividade (OHNO, 1988).

$$Produtividade = \frac{Qtd\ Produtos}{Qtd\ Trabalho} \qquad (1)$$

Enquanto Ford colocou o foco no aumento da produção (quantidade total de produtos), o engenheiro da Toyota absorveu as evoluções da Produção em Massa, mas concentrou-se em diminuir os esforços (quantidade de trabalho aplicado) e na otimização do processo (OHNO, 1988).

Ohno também assimilou fortemente os conhecimentos do americano pioneiro no movimento da Qualidade Total, W. Edwards Deming, que ministrou palestras educativas no Japão com o tema qualidade e produtividade. Ele ensinou que suprir e exceder as expectativas do cliente era tarefa de todos dentro da organização e alargou a definição de cliente para incluir requisitantes internos e externos, ou seja, a etapa ou departamento subsequente na linha de produção se tornou cliente da anterior e deveria ser suprida na quantidade e qualidade estabelecidas, e no tempo esperado. Por fim, Deming encorajou os japoneses a adotar uma abordagem sistemática para resolução de problemas, que posteriormente ficou conhecida como o Ciclo de Deming, ou *Plan-Do-Check-Act* (PDCA), uma prática central da melhoria contínua (WOMACK, JONES e ROOS, 1991).

Porém, essas mudanças foram gradativas e elaboradas de forma empírica, sendo necessários mais de 20 anos para implementar o conjunto de ideias na cadeia de produção da Toyota. Ao final, o intento foi um sucesso, com consequências extraordinárias para produtividade, qualidade do produto e responsividade a demandas do mercado (WOMACK, JONES e ROOS, 1991).

2.1.3 Sistema de Produção Toyota

Os números da Toyota demonstram uma empresa consistente, resiliente, com um desempenho superior, que a coloca entre as companhias mais bem sucedidas do mundo:

- Em 2006 enquanto a GM e Ford reportaram prejuízo, obteve $13,7 bilhões de dólares em lucro (OSONO, SHIMIZU e TAKEUCHI, 2008)
- Seu valor de mercado em maio de 2007 foi 1,5 vezes maior que GM, Ford e Daimler-Chrysler juntas (LARMAN e VODDE, 2009)
- Em 2008 superou a GM e se tornou a maior montadora de veículos em vendas e lucro (LARMAN e VODDE, 2009)
- J. D. Power e Associados classificou a Toyota como o produtor de veículos em alto volume de maior qualidade. Seu modelo de alto padrão Lexus esteve em primeiro lugar no quesito segurança por 13 anos consecutivos, a frente de Mercedes e BMW (OSONO, SHIMIZU e TAKEUCHI, 2008)

A Toyota chamava a atenção do mundo desde a década de 1980, quando além de números crescentes em vendas e *marketshare*, demonstrava uma eficiência produtiva muito acima de seus concorrentes. Em média, ela despendia menos da metade do tempo na montagem dos veículos, com aproximadamente 3 vezes menos defeitos, em espaço reduzido e utilizando menos estoque (como visto na Tabela 4). Isso se refletia na qualidade dos carros japoneses, que duravam mais que os americanos e com menos necessidade de reparos (WOMACK, JONES e ROOS, 1991).

Tabela 4 – Comparação das fábricas da GM e Toyota em 1987

	GM Framingham	**Toyota Takaoka**
Horas de montagem por carro	31	16
Defeitos por 100 carros	135	45
Espaço de montagem por carro	8,1	4,8
Estoque médio de partes	2 semanas	2 horas

Fonte: Womack, Jones e Roos (1991, p. 83)

Na década seguinte ficou ainda mais aparente que havia algo de especial na Toyota, mesmo se comparada com outras montadoras japonesas. Não era uma questão de design ou performance dos carros, era a forma como a Toyota projetava e construía os carros que os levou a uma consistência incomparável em termos de processo e produto. A Toyota projetava carros mais rápido, com mais confiabilidade, com custo competitivo mesmo pagando os salários relativamente altos dos trabalhadores japoneses. Igualmente, toda vez que a Toyota parecia enfraquecida ou vulnerável aos competidores, ela miraculosamente revertia o problema e reaparecia mais forte (LIKER, 2004).

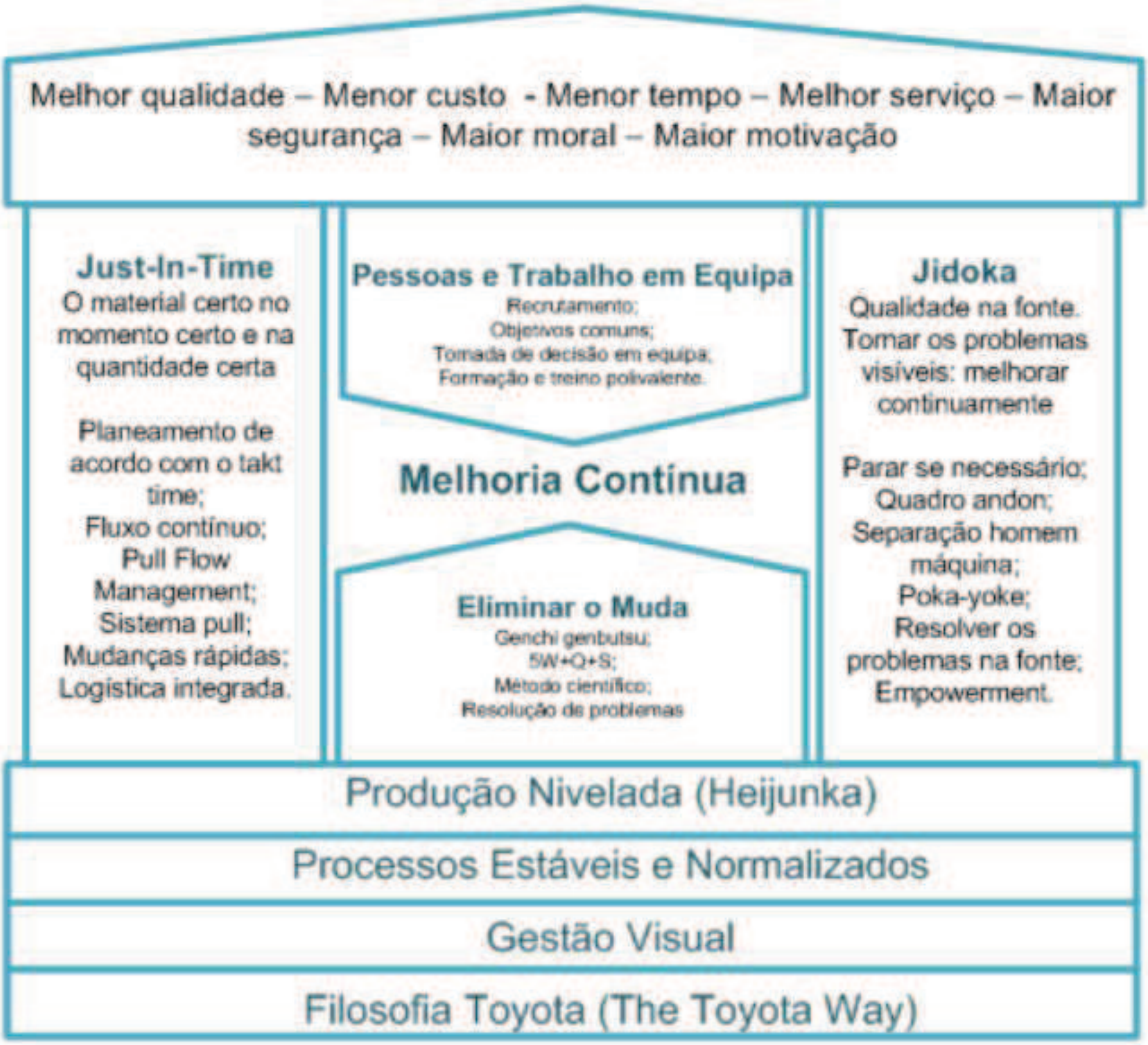
A performance consistente da Toyota era o resultado direto de sua excelência operacional, que se tornou um ativo de grande valor e sua arma estratégica. Essa excelência operacional estava embasada parte em ferramentas e métodos de eficiência da produção e melhoria da qualidade que ficaram conhecidas na indústria da manufatura com termos nas línguas inglesa e japonesa como *just-in-time*, *kaizen*, *one-piece flow*, *jidoka* e *heijunka*. Mas ferramentas e técnicas não eram as armas secretas que transformaram seu negócio. O sucesso contínuo da Toyota em implementar essas ferramentas e técnicas vinha de uma filosofia muito mais profunda, baseada no entendimento do ser humano e suas motivações. Isso era decorrente do desenvolvimento de uma cultura de liderança, de times autônomos, de criar estratégias disruptivas, de construir relacionamento com os fornecedores e manter uma organização em constante aprendizagem.

Por décadas a Toyota esteve aplicando e melhorando esse sistema de produção no dia-a-dia do chão de fábrica, porém sem documentá-lo. Trabalhadores e gerentes iam aprendendo e renovando suas práticas constantemente e a comunicação informal dessas melhores práticas se espalhava pelos departamentos, fábricas e fornecedores pois ainda se tratava de uma estrutura pequena. Mas com a maturidade do processo e o crescimento estrutural, ficou claro que um trabalho de formação era imprescindível e interminável. Para isso foi formalizado o Sistema de Produção Toyota (do inglês *Toyota Production System*, TPS) (OHNO, 1988).

Uma representação conceitual muito conhecida do TPS é a casa (que é um bom símbolo pois estabelece relação com uma estrutura sólida). Ela só se mantém de pé se o telhado, pilares e fundação forem consolidados e bem arregimentados. Existem diferentes versões da casa do TPS, mas o princípio se mantém o mesmo. O telhado representa os objetivos de melhor qualidade, baixo custo e o menor *lead time*. Os pilares laterais estabelecem o *just-in-time*, provavelmente a mais visível e publicada característica do TPS e *jidoka*, que em resumo significa não permitir que um defeito passe para a próxima etapa e também liberar as pessoas das máquinas por meio da automação com um toque humano (LIKER, 2004).

No centro do sistema estão as pessoas, que fazem tudo acontecer, e a melhoria contínua, força motriz do processo. Para sustentar toda a estrutura estão os elementos que compõem os fundamentos da casa. São eles a necessidade de um processo padronizado, estável e confiável, e também o *heijunka*, que simboliza o nivelamento da produção em volume e variedade. Uma agenda de produção nivelada é necessária para manter o sistema estável e permitir o mínimo de estoque. Grandes picos de produção de certos produtos e a falta de outros criam a escassez de peças a não ser que um grande estoque seja adicionado ao sistema (LIKER, 2004).

Figura 3 – A casa do *Lean* representando o Sistema de Produção Toyota



Fonte: Jeffrey Liker (2004, p. 33)

Os elementos da casa reforçam uns aos outros, estabelecendo uma relação sinérgica e crítica ao seu funcionamento. O *just-in-time* (JIT) estabelece a remoção máxima do estoque utilizado como *buffer* nas operações. O ideal buscado é o *one-piece flow*, ou seja, o fluxo de uma única peça de cada vez. Usar *buffers* menores significa remover a rede de proteção, ou dizer que defeitos e problemas na qualidade se tornarão imediatamente evidentes. Isso reforça o *jidoka*, o processo de produção que estabelece que o trabalhador deve resolver o problema imediatamente para que a produção recomece. O requisito de trabalhar com estoque mínimo e paradas na produção toda vez que surge um problema causa instabilidade, eleva o senso de urgência dos trabalhadores e o desejo de melhoria contínua para que aquilo não mais se repita (LIKER, 2004).

Na produção em massa, quando uma máquina parava, não havia senso de urgência pois o departamento de manutenção era chamado para consertá-la enquanto o estoque era consumindo e mantinha a produção operante. Em contraste, no TPS, quando um operador desliga um equipamento para repará-lo, outras operações em breve irão parar, gerando uma crise. Portanto, há sempre um senso de urgência em todos para reparar os problemas em conjunto no mínimo espaço de tempo pois do contrário toda a linha será impossibilitada de operar. Se o mesmo problema ocorrer repetidamente, a gestão rapidamente concluirá que essa é uma situação crítica e que providências iminentes devem ser tomadas em caráter improrrogável (LIKER, 2004).

Não obstante todas as ferramentas apresentadas, é importante notar que o TPS não é uma caixa de ferramentas contendo *just-in-time*, *jidoka*, 5S, *kanban*, *andon* e outras. O TPS é um sofisticado sistema de produção no qual todas as partes contribuem para o todo. A raiz de todo o sistema está em encorajar e suportar as pessoas a continuamente melhorar o processo em que estão inseridas. Autores e publicações que apresentam o TPS como um conjunto de ferramentas, interpretaram de forma equivocada o sistema e não se aprofundaram a ponto de entender que esse não é um modelo a ser copiado em suas ações, mas nos princípios que direcionam a prática. Quando um gerente busca implementar uma operação enxuta em sua empresa copiando as ferramentas utilizadas pela Toyota sem observar o perfil filosófico que direciona a prática, ele perde o propósito que está por trás das ferramentas e não consegue absorver todo o potencial da mudança (LIKER, 2004).

Liker (2004), ao apresentar o Sistema de Produção Toyota, deixa clara a série de princípios que regem as práticas, influenciando a tomada de decisão tanto no pensamento estratégico quanto nas escolhas operacionais. Fica evidente o enfoque de pensamento a longo prazo, com o objetivo de agregar valor aos clientes e à sociedade por meio da construção de uma organização de aprendizagem, que se adapta às mudanças no ambiente externo e se perpetua como uma organização produtiva. Isso esclarece que a empresa tem um senso filosófico que suplanta qualquer decisão de curto prazo, trabalhando e crescendo em torno de um propósito que é maior do que simplesmente ganhar dinheiro.

É possível empregar a variedade de ferramentas utilizadas na Toyota seguindo poucos princípios do TPS. O resultado é um salto de performance discreto, de curto-prazo e não sustentável. Em contrapartida, as organizações que seguem verdadeiramente o conjunto de princípios do TPS se posicionam para desenvolver uma vantagem competitiva perene, duradoura. Essa filosofia que é a chave do sucesso da Toyota é chamada de *Lean Thinking* ou simplesmente *Lean*.

2.1.4 Lean Thinking

O termo *Lean* foi usado pela primeira vez por John Krafcik (1988) no artigo "*Triumph of the Lean Production System*" (do inglês, O Triunfo do Sistema de Produção *Lean*), texto base de seu mestrado no Massachusetts Institute of Technology (MIT). Krafcik, um engenheiro de qualidade do setor automotivo, havia trabalhado numa *joint venture* da Toyota/GM localizada na Califórnia e usou os anos de experiência nessa montadora para desenvolver sua pesquisa acadêmica.

Posteriormente, em 1991, esse material foi utilizado pelo grupo IMVP (International Motor Vehicle Program), também do MIT, para produção do best-seller "A Máquina que Mudou o Mundo" (WOMACK, JONES e ROOS, 1991). Essa foi a publicação de maior importância para a disseminação dos princípios *Lean* e sua implementação na indústria.

O termo *Lean* (do inglês, enxuto) foi escolhido porque o objetivo da filosofia era sempre usar menos de tudo: dinheiro, matéria-prima, maquinário, ferramentas, espaço, pessoas e tempo. Para isso, era necessário que fossem usados menos recursos não só na produção, mas na concepção dos produtos, na manutenção do parque industrial, na retificação de peças, enfim, em todos os setores da empresa.

Em 1995, Womack e Jones escreveram um segundo livro intitulado "*Lean Thinking: banish waste and create wealth in your corporation*" (WOMACK e JONES, 2010). Essa publicação enumerou os conceitos básicos da filosofia *Lean*, dispostos em 5 princípios (valor, fluxo de valor, fluxo contínuo, produção puxada e perfeição) e a aplicação dos mesmos não somente à manufatura, mas a diversas outras áreas de uma empresa.

O pensamento *Lean* deve ser iniciado pela definição exata do que é valor para o cliente em termos do produto, recursos oferecidos e preço. Especificar o valor da forma correta é crucial e o primeiro princípio do pensamento *Lean*. Falhar nesse passo observando todos os outros, seria prover o produto ou serviço errado da maneira correta, ou seja, um grande desperdício.

Em seguida, é necessário mapear o fluxo do valor, isto é, reconhecer, entender e classificar individualmente cada atividade do processo. Pode-se categorizar as etapas de três formas. Existem as atividades-chave, que sem dúvida criam valor, sendo as mais importantes de todo o processo e que devem ser cultivadas tendo em vista o aprimoramento contínuo de sua preparação e execução. Em segundo lugar, existem as atividades que não agregam valor diretamente ao produto, mas que são necessárias dado o processo, os ativos e as tecnologias envolvidas na produção. A esse tipo de atividade, dado que não agregam valor porém são indispensáveis, é correto agir em prol de sua atenuação. Por último, estão as atividades que não adicionam nenhum valor ao produto, que devem ser sumariamente eliminadas (WOMACK e JONES, 2010).

A atenção de gerentes historicamente esteve em coordenar e supervisionar as pessoas, mas o que realmente importa é gerir o fluxo de valor do produto ou serviço. Se as atividades não forem precisamente identificadas e analisadas, elas não podem ser aperfeiçoadas, contestadas ou eliminadas.

Quanto a classificação das atividades, pode-se fazer uma analogia com uma corrida de Fórmula 1:

1. Gera valor: acelerar dentro do traçado mais eficiente
2. Não gera valor: frear onde não precisa ou dirigir fora do traçado
3. Não gera valor, porém é necessário: parada no pit stop

Uma vez que o valor foi devidamente especificado, o fluxo de valor totalmente mapeado (e as etapas de desperdício eliminadas), é hora de dar o próximo passo no *Lean Thinking*: gerenciar o fluxo contínuo.

Os modelos de produção tradicionais falharam sobremodo ao propagar o conceito que performar tarefas em grandes lotes é melhor. Taiichi Ohno atribuiu aos primeiros agricultores o modo de pensar em filas e lotes, uma vez que o modelo anterior de caça presava por executar uma atividade de cada vez. Assim que a agricultura começou a se expandir, o pensamento em lote (colheita uma vez por ano) e estoques (depósito de grãos) se tornaram padrão (OHNO, 1988).

O pensamento *Lean*, por sua vez, tinha o desafio de criar um fluxo contínuo numa produção de lotes pequenos (e, se possível, unitários), por saber que é um modelo mais eficiente. O problema é que o pensamento de fluxo contínuo em lotes pequenos é contra intuitivo. Parece muito óbvio para a maioria das pessoas que o trabalho deve ser organizado em departamentos e em lotes. Então, uma vez que departamentos e equipamentos especializados para a manufatura em lote e alta velocidade são organizados, as aspirações dos funcionários em departamentos e os cálculos do gerente de produção visando a utilização máxima dos equipamentos, trabalham contra a mudança para fluxo contínuo (WOMACK e JONES, 2010).

A alternativa *Lean* é redefinir o funcionamento de funções e departamentos para que possam gerar uma contribuição positiva à criação de valor e atender as reais necessidades dos colaboradores em qualquer ponto da cadeia, suscitando o interesse pela fluidez do processo. O primeiro efeito visual da conversão de departamentos e lotes para times de produtos e fluxo é que o tempo despendido para ir do conceito ao lançamento, da venda à entrega, da matéria-prima ao produto acabado, cai drasticamente. Quando o fluxo é introduzido, produtos que demandariam anos para serem projetados são feitos em meses, ordens que levariam dias são completadas em horas.

O princípio da produção ou sistema puxado significa que ninguém na cadeia de produção deve fabricar um produto até que o cliente demande por ele. A melhor forma de entender a lógica do sistema puxado é começar com um cliente real expressando uma demanda por um produto e trabalhar em todas as etapas de produção até a condução do produto acabado de volta ao cliente. Dessa forma, o consumidor passa a puxar o fluxo de valor, reduzindo a necessidade de estoque (WOMACK e JONES, 2010).

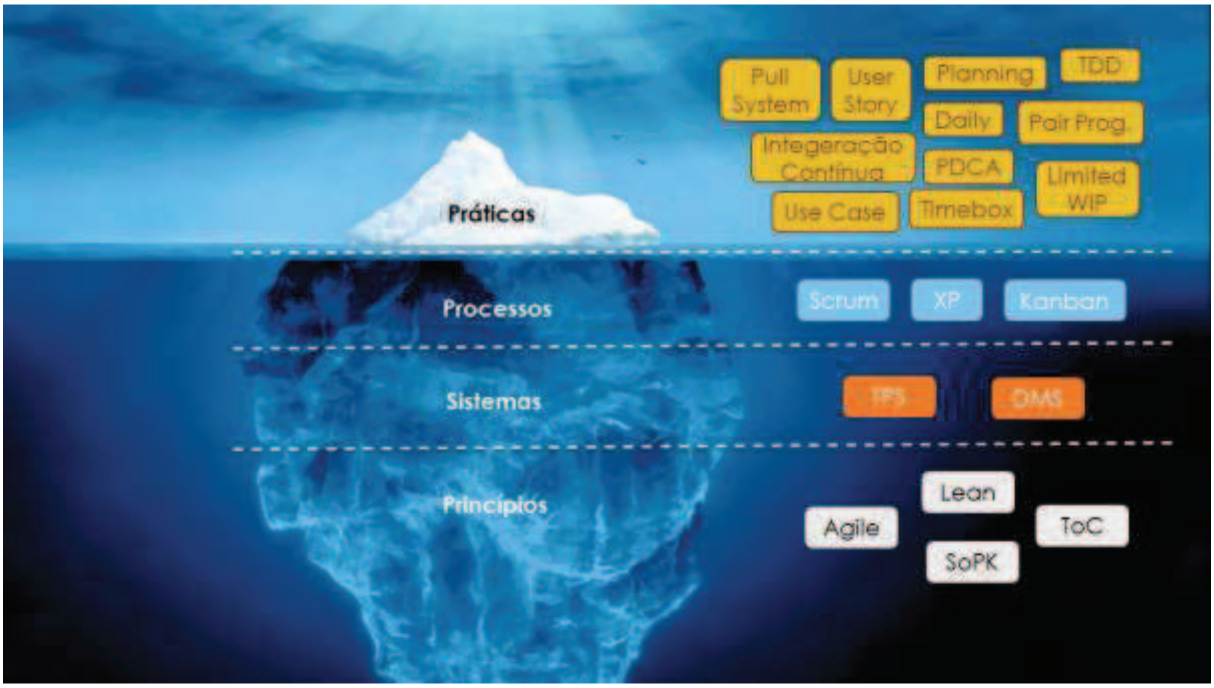
Assim que as organizações passam a especificar valor de forma adequada, identificar o fluxo de valor, promover um fluxo contínuo, e implantar um sistema puxado, algo diferente passa a acontecer. Os envolvidos passam a entender que não há fim no processo de redução de esforços, tempo, espaço, custo e erros, ao oferecer um produto que vai se tornando cada vez mais aderente ao que o cliente de fato deseja. Isso significa uma busca contínua e interminável pela perfeição, o quinto e último princípio do pensamento *Lean*.

Isso acontece pois os quatro princípios iniciais interagem entre si num ciclo virtuoso. Gerar valor num fluxo de forma mais acelerada sempre expõe desperdícios escondidos no processo. Quanto mais intenso o sistema puxado, mais os impedimentos ao fluxo são revelados. Times de produtos dedicados, em contato direto com os clientes, sempre encontram uma forma mais eficaz de entregar valor e suprir sua real necessidade. A busca pela melhoria contínua em direção a um estado ideal deve nortear todos os esforços da empresa em processos transparentes que são conhecidos e promovidos por todos os membros da cadeia (WOMACK e JONES, 2010).

2.1.5 Práticas Lean

A filosofia *Lean* é composta por princípios que direcionam práticas. Porém, o que fica mais evidente no processo produtivo de empresas como a Toyota são as técnicas e ferramentas que se materializam nas práticas do dia-a-dia. Michael Stoecklein (2005) faz uma alegoria pertinente ao descrever a teoria do pensamento sistêmico de Deming. Numa associação com a figura de um iceberg, ele descreve ferramentas e práticas como a parte visível do bloco de gelo, enquanto os princípios e processos são a base da peça, que ficam invisíveis aos olhos, servem como base de sustentação e compõem a parte mais relevante e representativa do todo. Assim, entende-se que, mesmo as práticas sendo os agentes mais evidentes no sistema, pouco valem se não estiverem alicerçados nos princípios corretos.

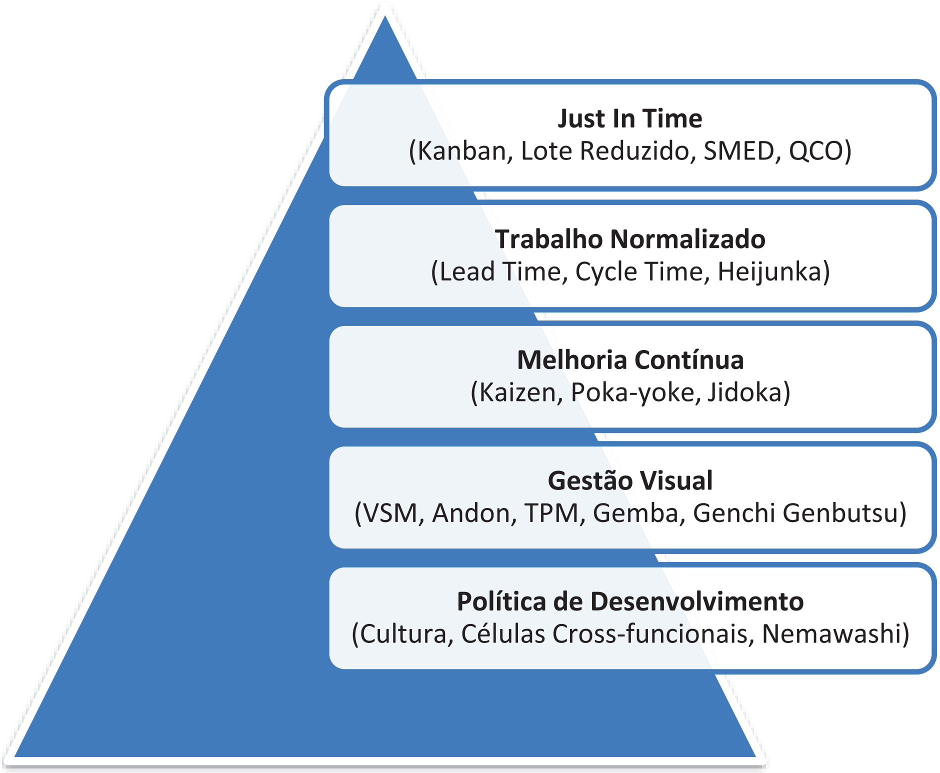
Figura 4 – Iceberg de Stoecklein no contexto-exemplo de uma empresa de software



Fonte: adaptado de Michael Stoecklein (2005)

Para apresentação dos princípio e práticas *Lean,* será utilizado o modelo adaptado de Goforth, como visto na Figura 5:

Figura 5 – Modelo de Princípios e Práticas *Lean*



Fonte: adaptado de Goforth (2008)

Para melhor compreensão, é importante definir os princípios e práticas *Lean*:

- *Kaizen*

Kaizen é o termo japonês utilizado no TPS para representar uma série de melhorias contínuas e incrementais que visam aprimorar o processo, eliminando o desperdício e aperfeiçoando as atividades geradoras de valor. O *kaizen* promove o auto-conhecimento, o desenvolvimento de habilidades para trabalhar efetivamente em grupo, solucionar problemas, avaliar e aprimorar processos, coletar e analisar dados. Além do mais, ele traz o poder da proposta e muitas vezes o poder da decisão para os colaboradores, feito por meio da discussão aberta e do consenso. O *kaizen* carrega a filosofia da busca pela perfeição, o que sustenta o TPS no dia-a-dia (WOMACK e JONES, 2010).

- *Jidoka*

Entre as inovações de Sakichi Toyoda estava um mecanismo especial que parava automaticamente o tear assim que o fio se quebrava, uma invenção que evoluiu para um sistema mais amplo, sendo um pilar do sistema de produção da Toyota chamado de *jidoka*. Essencialmente, *jidoka* refere-se a construção com qualidade e ao desenho de operações e equipamentos de forma que os trabalhadores não fiquem presos às máquinas e estejam livres para executar trabalho de alto valor agregado (TOYODA, 1987).

- *Poka-yoke*

Poka-yoke é um termo japonês que significa "a prova de erros" e, no *lean manufacturing*, pode ser visto em qualquer mecanismo que auxilia um operador a evitar erros. Seu propósito é eliminar defeitos por meio da prevenção, correção e o direcionamento da atenção para os erros assim que eles ocorrem. O conceito foi originalmente descrito com o nome *baka-yoke,* ou "a prova de tolo" e deixa implícito o conceito de buscar excluir a possibilidade do erro humano a todo custo (SHINGO, 1996).

Um exemplo prático cotidiano é o mecanismo existente em veículos para evitar que faróis, lanternas e ar condicionado fiquem ligados por descuido após o desligamento do veículo. Um mecanismo *poka-yoke* pode eliminar ou mitigar esse risco desligando o sistema elétrico do carro ou simplesmente emitindo um sinal sonóro ao condutor.

- Células Cross-funcionais

O arranjo da operação e das pessoas em times ou células de trabalho também foi uma inovação trazida pelo *Lean* frente a tradicional linha de montagem retilínea. Em termos práticos, a organização em células aprimora a comunicação e incentiva a cooperação entre os indivíduos. Outra característica importante é a utilização de times cross-funcionais, ou seja, times de generalistas que podem desempenhar igualmente qualquer função. Isso é de suma importância quando se deseja diminuir a variabilidade de demanda entre as etapas do fluxo de trabalho e útil para suprir a falta de um membro em qualquer função (WOMACK, JONES e ROOS, 1991).

- Gestão Visual

Os controles visuais habilitam qualquer pessoa, que entre no ambiente de trabalho, a saber num curto espaço de tempo o que está acontecendo na agenda da produção, qual o fluxo de trabalho, quais são as próximas demandas, qual o nível do estoque, qual o nível de utilização dos recursos e a qualidade. Esses controles devem ser eficientes, auto-reguláveis, gerenciados pelo trabalhador e podem incluir cartões *kanban,* luzes, quadros, painéis, ferramentas assinaladas por cores, áreas delimitadas com fitas adesivas, etc (KILPATRICK, 2003).

- *Andon*

Andon refere-se a um sistema de notificação da gestão e dos times sobre problemas de processo e qualidade. O conceito central é a gestão visual, que é feita por meio de um placar que incorpora luzes de sinalização e indica qual estação de trabalho tem o problema. O alerta pode ser ativado manualmente por um trabalhador que usa uma corda ou botão, ou automaticamente pelo próprio equipamento da produção. Um sistema *andon* é um dos principais elementos do método *jidoka* de controle de qualidade. Ele dá ao trabalhador a possibilidade de parar a produção quando um defeito é encontrado para iniciar imediatamente o reparo (KATAYAMA, 2010).

Figura 6 – Exemplo de Painel Andon

Fonte: (DORUK OTOMASYON, 2010)

A Figura 6 mostra um Painel Andon, no qual fica visível a todos os colaboradores e gestores que as estações de trabalho assinaladas em vermelho estão com algum problema (por exemplo, falta de matéria-prima, defeito na fabricação, maquinário danificado).

- *Kanban*

O *kanban* é um sistema de gerenciamento da produção do ponto de vista logístico, onde se estabelece o limite de processamento de cada etapa do sistema. Isso é feito com a utilização de *kanbans* (cartões) para sinalizar a capacidade produtiva do sistema e assim favorecer o controle de estoque por meio do limite do trabalho em progresso (do inglês *work-in-progress*, WIP).

Após voltar da série de viagens à América, Taiichi Ohno desenvolveu o sistema *kanban* inspirado no modelo de reposição visto nos supermercados americanos. O funcionamento básico desse sistema estabelecia que a reposição de itens às gôndulas era determinado pelo consumo. Em termos aplicados, um veículo só seria reposto no estoque assim que vendido, peças só seriam produzidas assim que o veículo saísse do estoque, e assim por diante, num efeito cascata que permeava até as primeiras etapas da produção, inclusive na aquisição de fornecedores. O *kanban* é o método essencial que habilita o *just-in-time* (WOMACK, JONES e ROOS, 1991).

- *Nemawashi*

Pode-se perceber que empresas praticantes da filosofia *Lean* devem aprender a conviver com mudanças constantes, pois isso é parte integral do processo de trabalho e das implementações de melhoria contínua. Nesse contexto, o *nemawashi* denota um procedimento informal que estabelece as bases para um processo de mudança. Isso compreende falar com as pessoas, reunir apoio, colher *feedbacks* e opiniões, sendo um elemento importante em grandes alterações pois possibilita o consenso dos envolvidos e diminui os efeitos da resistência à mudança. *Nemawashi* pode ser traduzido como dar a volta na raiz, ou cavar ao redor das raízes de uma árvore para prepará-la para um transplante.

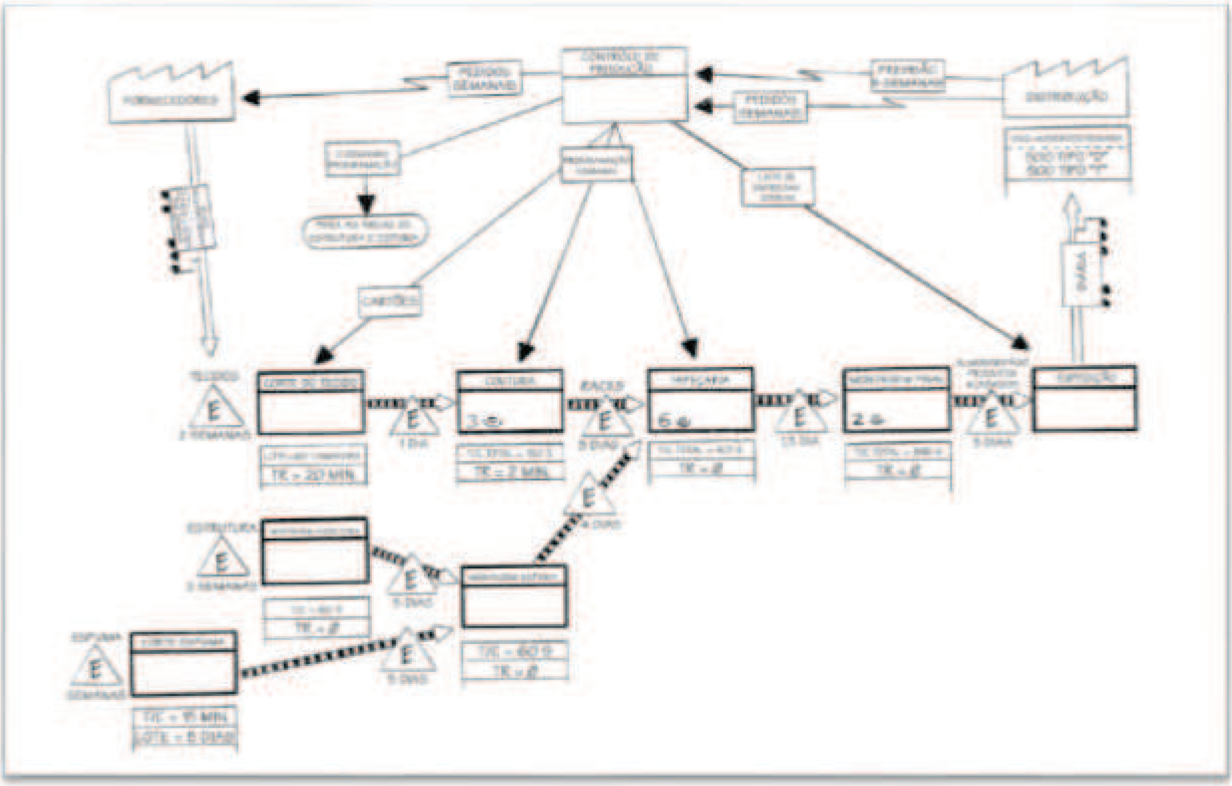
- *Gemba / Genchi Genbutsu*

O significado do termo japonês g*emba* é o "local real", e numa empresa representa o lugar onde o valor é criado. A ideia é que gerentes têm uma percepção parcial e superficial ao se basear somente em informações obtidas em planilhas dentro de seus escritórios. É necessário ir ao g*emba,* ver o processo em execução, entender o trabalho, esclarecer dúvidas e aprender como tudo é feito. Enquanto *gemba* significa o local onde tudo acontece, a atitude de ir e ver é representada por outro termo, o *genchi genbutsu*.

- *Value Stream Map* (VSM)

O *Value Stream Map* (VSM), ou mapa de fluxo de valor, é um modelo que representa todas as etapas do processo produtivo e seus envolvidos, oriundo do princípio *Lean* de mapeamento do fluxo de valor. O mapa tem grande utilidade para analisar o estado atual, pontos de otimização, desperdícios, filas e também projetar o estado futuro da cadeia produtiva. Dessa forma, são derivadas uma série de ações para implementar o estado almejado. A Figura 7 apresenta um exemplo de VSM.

Figura 7 – Exemplo de Mapa de Fluxo de Valor (VSM)



Fonte: (LEAN TI, 2013)

- *Total Productive Management* (TPM)

O princípio do TPM está na dinâmica proativa e progressiva de manutenção de maquinário e ferramentas, estelecido sobre o conhecimento e cooperação dos operadores, vendedores de equipamentos, engenheiros e pessoal de suporte para otimizar o desempenho das máquinas. Resultados desse trabalho incluem eliminação de quebras de tempo de parada (planejado e não planejado), melhoria na utilização, maior *throughput*, melhor qualidade do produto final, menores custos operacionais, vida útil do equipamento estendida e menor custo final de manutenção (KILPATRICK, 2003).

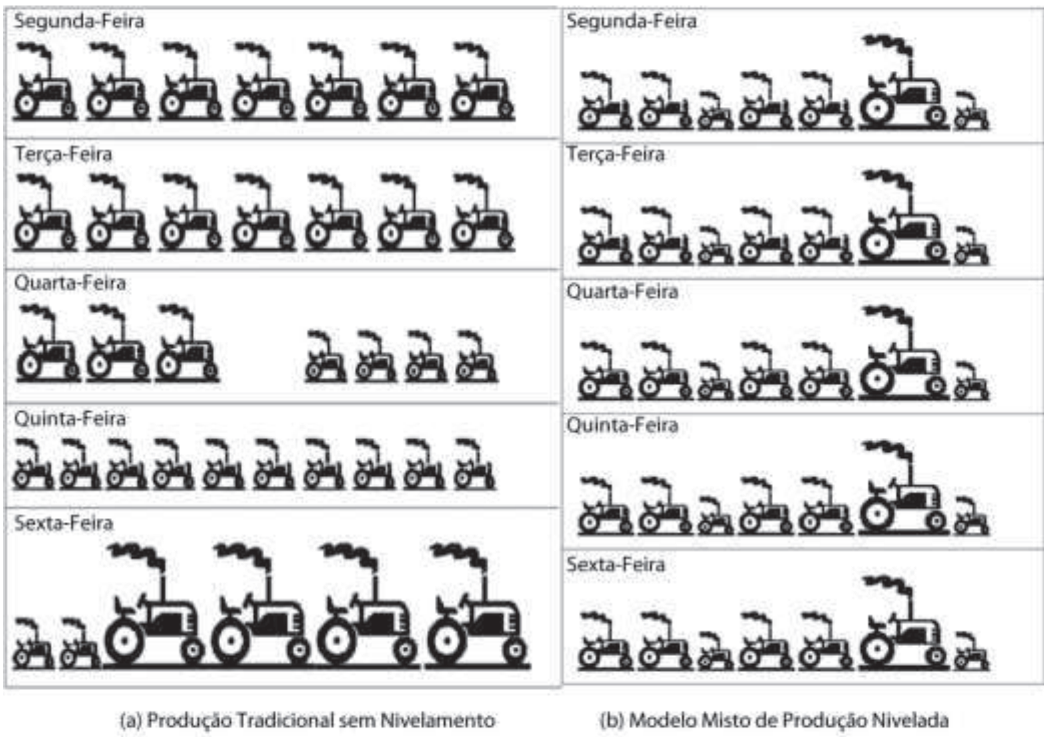
- *Lead Time / Cycle Time*

Lead time e *cycle time* são métricas de tempo num processo *Lean,* essenciais para a análise e avaliação do processo, da qualidade das ações de *kaizen* e dos pontos de desperdício e melhoria. *Lead time* compreende o intervalo entre o início e o término de uma atividade, contando os passos internos do processo e incluindo o tempo de espera entre eles. *Cycle time* é a média do intervalo entre o término das atividades. Para ilustrar, considerando-se um fluxo em uma tubulação, o *cycle time* é o tempo médio entre cada unidade que sai do tubo.

- *Heijunka*

Heijunka ou nivelamento da produção é um conceito relacionado a planejamento da produção e é obtido pelo sequenciamento dos pedidos e pela eficiência das trocas de *setup*. O *heijunka* converte a instabilidade da demanda dos clientes em um nivelado e previsível processo de manufatura, e é geralmente usado em combinação com outras técnicas *lean* de produção para estabilizar o fluxo de valor. As suas vantagens são a fabricação estar preparada para variedade de quantidade e tipo, maior disponibilidade de escolha aos clientes, diminuição de estoques e da variabilidade do processo (GALGANO, 2004). A Figura 8 mostra o planejamento semanal da produção na forma tradicional (sequencial) e com *heijunka* (nivelada).

Figura 8 – Produção Tradicional x Produção Nivelada (heijunka)



Fonte: (Heijunka - flexibilizar e nivelar a produção, 2013)

- Lote Reduzido

Historicamente, empresas de manufatura operavam com grande lote de trabalho com o objetivo de maximizar a utilização do maquinário, assumindo que os tempos de troca eram fixos e não poderiam ser reduzidos. Baseados em outros princípios *Lean* como a eliminação do estoque e o *one-piece flow,* pode-se afirmar que o tamanho ideal do lote é 1. Todavia, um lote de tamanho 1 nem sempre é praticável, por isso o objetivo é praticar a melhoria contínua para reduzir o tamanho do lote o máximo possível (KILPATRICK, 2003).

- Troca Rápida (SMED e QCO)

Ainda que o sistema de produção fosse hábil em construir produtos para pedidos específicos, ele não era tão bom em lidar com surtos de demanda, sendo o *heijunka* a técnica empregada para mitigar problemas relacionados a isso. Porém, a produção em grande variedade gerava um outro problema quando os artefatos não pudessem ser produzidos utilizando o mesmo *setup* ou as mesmas ferramentas (por exemplo, entre veículos de grande e pequeno porte, ou entre carros e caminhões). As siglas SMED (*Single Minute Exchange of Dies*) e QCO (*Quick Change Over*) são utilizadas para representar as práticas *Lean* que promovem a troca rápida de ferramentas ou *setups*, fator crucial para o fluxo produtivo com grande variedade (*heijunka*) (WOMACK, JONES e ROOS, 1991).

- *Just-in-time* / *One-piece-flow* / Sistema puxado

Just-in-Time (JIT) e *one-piece-flow* foram inovações lançadas pela Toyota na década de 1950, como método de facilitação à fluidez da produção. Ambos só puderam ser implantados efetivamente pois o tempo de troca do maquinário foi reduzido drasticamente (por meio do SMED e QCO), para que as etapas produzissem pequenas quantidades e só fabricassem novamente quando esse montante inicial fosse consumido pela etapa subsequente.

Um processo *Lean* deve desenvolver os conceitos do JIT na busca pelo estado ideal do fluxo contínuo que é trabalhar as peças uma-a-uma (*one-piece-flow*). Para isso, é essencial que o maquinário seja dimensionado da maneira correta, o que frequentemente significa usar máquinas mais simples, menos automatizadas e mais lentas, porém mais precisas e repetíveis que as tradicionais.

Um segundo ponto é laborar num sistema puxado, onde o trabalho é iniciado na ponta mais extrema da cadeia produtiva, o time de vendas. Nessa configuração, o pedido do vendedor é a primeira etapa do sistema produtivo, enviando ordem para a fábrica construir um veículo específico e fazendo com que as etapas internas puxem as subsequentes e assim sucessivamente. Dessa maneira, para-se de produzir carros de forma antecipada e para compradores desconhecidos e passa-se a trabalhar num sistema sob encomenda (WOMACK, JONES e ROOS, 1991).

2.2 Engenharia de Software

Desde a década de 1960, a evolução das plataformas computacionais e a invenção do computador pessoal impulsionaram a demanda por softwares mais dinâmicos, abrangentes, robustos e confiáveis. Em termos práticos isso acarretou em aumento de escopo, maior quantidade de requisitos, maior complexidade arquitetural, aumento do número de linhas de código e a consequente necessidade de coordenar grupos de pessoas cada vez maiores para implantação de tais demandas (LEHMAN, 1980; NORRMALM, 2011).

Quanto a relevância do software para as organizações, percebe-se um movimento crescente. No início, o uso estava destinado somente a automação de atividades manuais, passando a exercer o papel de diferencial competitivo frente aos concorrentes. Porém, essa relevância cresceu ainda mais, tornando os sistemas de informação ativos críticos ao negócio (APPLEGATE, MCFARLAN e MCKENNEY, 1996), ou, em alguns momentos, o próprio negócio (por exemplo, um site de *e-commerce*).

Diante deste cenário, produzir e manter o software dentro de custos, prazos e critérios de qualidade adequados tornaram-se requisitos obrigatórios, sendo necessário um robusto e eficiente processo de desenvolvimento de software (SOMMERVILLE, 2007).

A partir da década de 1970, diversas abordagens foram formalizadas, delimitando procedimentos para organização de tarefas, prescrição de artefatos e coordenação de pessoas, tendo em vista a concepção de um sistema de informação. Esses métodos chamados tradicionais tinham em comum o extenso uso documental, o planejamento detalhado das atividades e a segregação das pessoas em papéis específicos, sendo os mais conhecidos o

Modelo Cascata (ROYCE, 1970), o Modelo Espiral (BOEHM, 1988) e o Processo Unificado (JACOBSON, BOOCH e RUMBAUGH, 1999).

Em oposição aos métodos tradicionais, a partir da década de 1990, começaram a surgir métodos alternativos (posteriormente batizados de Métodos Ágeis), que possuíam características comum marcantes como a rápida resposta à mudança, a flexibilidade do plano, a ênfase na comunicação entre time e cliente, e as entregas rápidas e incrementais.

Pressman (2009) faz uma clara distinção entre os métodos tradicionais, que ele categoriza como prescritivos, e os Métodos Ágeis, chamados de adaptativos. De modo semelhante, Mary e Tom Poppendieck trazem essa distinção, porém utilizando outra classificação:

> O desenvolvimento de software é uma forma de desenvolvimento de produto. O desenvolvimento é um processo de transformação de ideias em produtos. Há duas escolas de pensamento sobre como fazer essa transformação: a escola determinística e a empírica. A escola determinística começa criando uma definição completa do produto e então cria uma realização daquela definição. A escola empírica inicia-se com um conceito alto-nível do produto e então estabelece ciclos de feedback bem definidos que ajustam as atividades para criar uma interpretação ótima do conceito (POPPENDIECK e POPPENDIECK, 2006, p. 21).

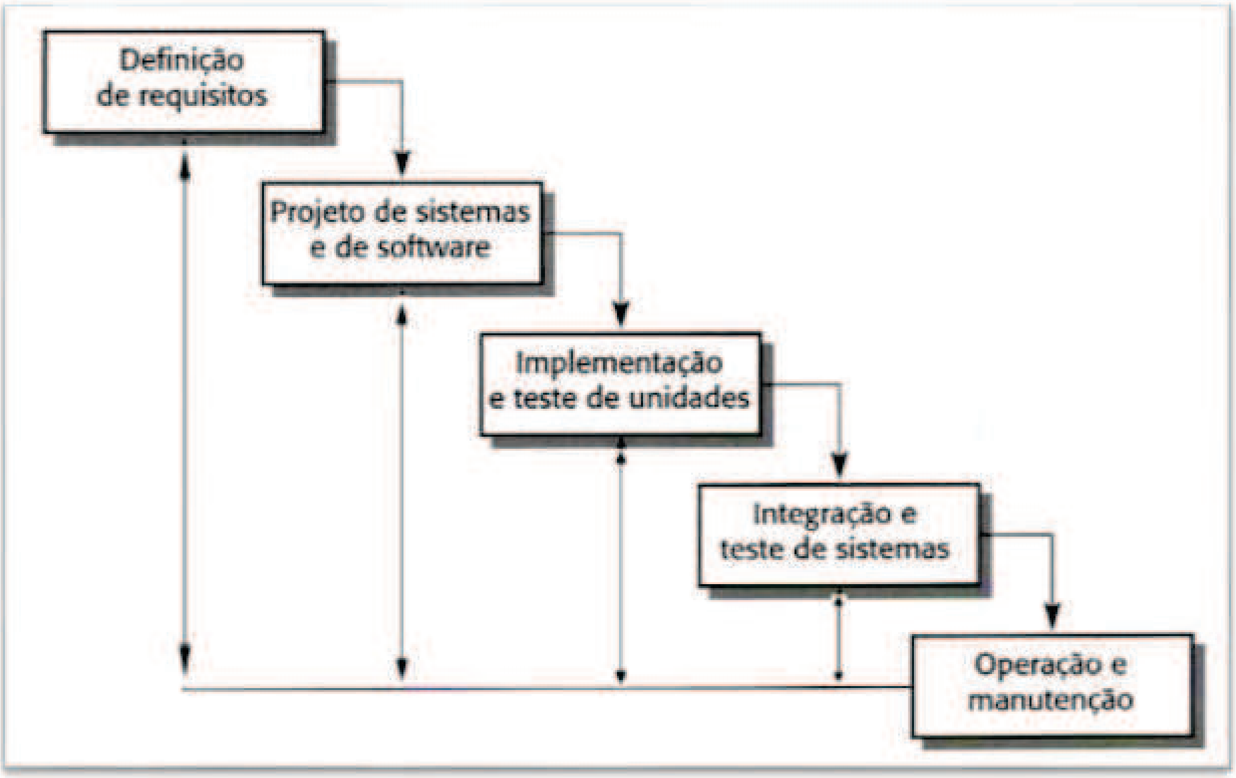
Entende-se que os Métodos Tradicionais, por serem altamente orientados a composição e execução de um plano, são seguidores da escola de pensamento determinística. Em contrapartida, os Métodos Ágeis, por estarem mais preparados à adaptação aos acontecimentos, estão alinhados com a escola empírica.

2.2.1 Modelo Cascata

Os métodos chamados tradicionais foram baseados na engenharia tradicional, estruturados em fases bem definidas nas quais é preciso aguardar a finalização de uma para o início da próxima. Mudanças e alterações nos artefatos do projeto resultam em retornar as fases anteriores para refazer todo um lote de trabalho, o que pode significar atraso e desperdício. O Modelo Cascata é um método tradicional no qual todas as atividades são consideradas fundamentais e acontecem de forma sequencial, da especificação até a implementação (ROYCE, 1970).

O Modelo Cascata de acordo com Sommerville (2008) é dividido em estágios, como visto na Figura 9.

Figura 9 – Estágios do Modelo Cascata



Fonte: Sommerville (2008, p. 66)

Os estágios do Modelo Cascata são:

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.

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.

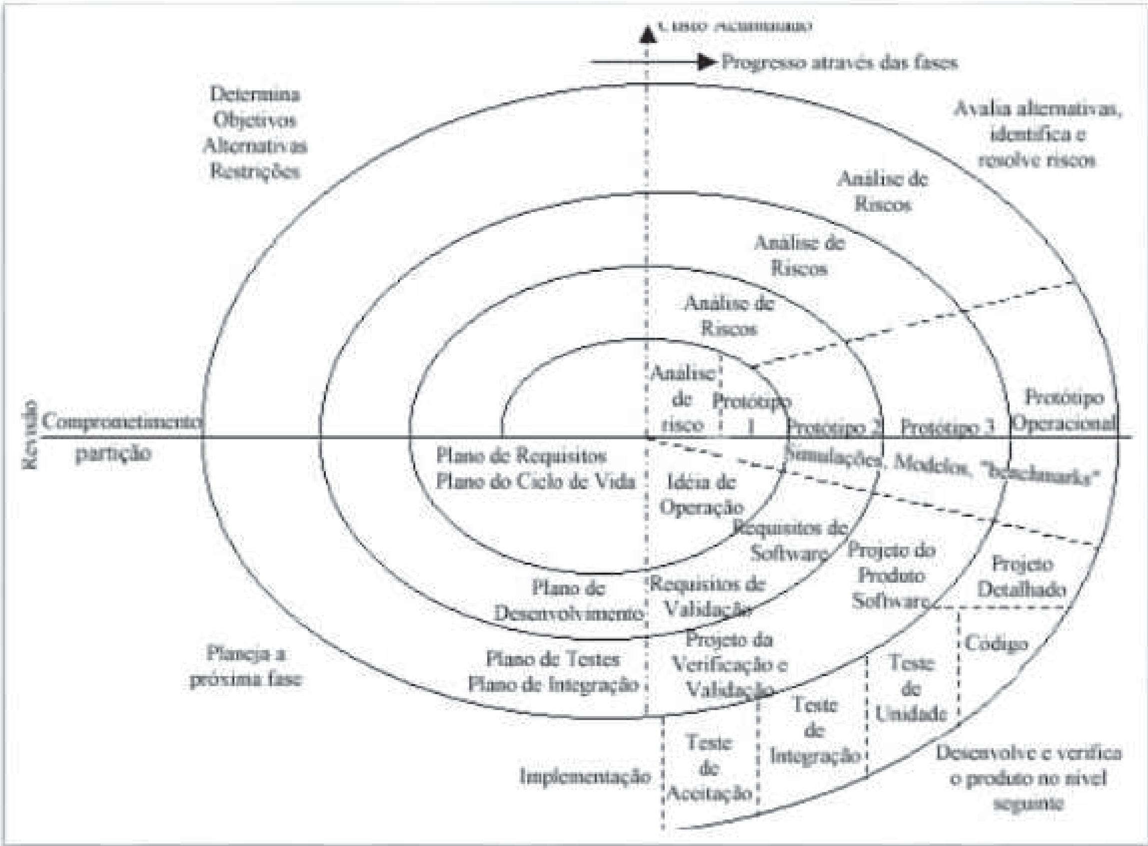
c) O cliente deve ser paciente, pois uma versão inicial do sistema não estará disponível até o final do projeto, dessa forma, se ocorrer algum erro grave na codificação ou nos requisitos, esses erros serão detectados e corrigidos no final do projeto.

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.

2.2.2 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. Como pode ser visto na Figura 10, 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.

Figura 10 – Representação do Modelo Espiral



Fonte: Boehm (1988, p. 68)

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 da 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).

2.2.3 PU – Processo Unificado

O Processo Unificado não é simplesmente um processo, mas sim um *framework* extensível que deve ser personalizado para organizações ou projetos específicos. O *framework* foi proposto inicialmente por Ivar Jacobson, Grady Booch e James Rumbaugh (1999), focado em linguagens orientadas a objeto e se utilizando de modelagem UML (*Unified Modeling Language*) para planejamento e documentação dos artefatos.

Seus autores discutiram a necessidade de um processo de software iterativo e incremental, direcionado por casos de uso e centrado numa arquitetura (JACOBSON, BOOCH e RUMBAUGH, 1999). 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)

2.2.4 Lean e os Métodos Ágeis

Ao final do século XX, diversos especialistas questionaram os pontos fracos dos métodos tradicionais e passaram a organizar métodos independentes (por exemplo, XP e Scrum), que deram origem ao movimento dos Métodos Ágeis. O enfoque principal dessa abordagem era: valorização da colaboração, da comunicação e do indivíduo, atenção maior ao sistema e menor às formalidades e a resposta imediata à mudança (BECK, *et al.*, 2001).

Kent Beck, criador do Extreme Programming (XP), organizou no ano 2000 um encontro para líderes e pessoas ligadas a métodos de desenvolvimento que ele classificava como "leve". Na mesma época, alguns artigos foram publicados pela comunidade de software, referenciando uma categoria de práticas e denominando-as como "peso leve".

Robert Martin, CEO e fundador da Object Mentor, uma consultoria especializada em processos de desenvolvimento de software, iniciou em setembro daquele ano uma convocação à todos esses entusiastas de metodologias não convencionais para uma série de reuniões em Utah e que culminou na edição, em fevereiro de 2001, de um manifesto.

O Manifesto Ágil é uma declaração que fundamenta o desenvolvimento de software seguindo princípios, valores e práticas tidas como ideais aos 17 signatários originais.

Valores do Manifesto Ágil:

- Indivíduos e interações mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

Princípios do Manifesto Ágil:

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- Software funcionando é a medida primária de progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Contínua atenção à excelência técnica e bom design aumenta a agilidade.
- Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.
- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Os autores do Manifesto eram considerados referência em processos de desenvolvimento de software, traziam consigo seus próprios métodos e se tornaram embaixadores dessas práticas. Os métodos em questão, seus criadores e colaboradores que estavam presentes no Manifesto Ágil:

- Scrum: Ken Schwaber, Jeff Sutherland, Mike Beedle;
- Dynamic Systems Development Method (DSDM): Arie van Bennekum;
- Crystal Methods: Alistair Cockburn;
- Feature-Driven Development (FDD): Jon Kern;
- Extreme Programming (XP): Kent Beck, Ward Cunningham, Ron Jeffries, Martin Fowler;
- Adaptive Software Development (ASD): Jim Highsmith.

Em 1993, Robert Charette iniciou o estudo da aplicação dos princípios *Lean* à engenharia de Sistemas de Informação, criando um conjunto de práticas que deu o nome Lean Development. Ele também questionava a ineficiência dos métodos tradicionais utilizados para concepção e manutenção de sistemas e passou a utilizar os valores da filosofia *Lean* para balizar um método próprio.

Charette foi convidado para o encontro de assinatura do Manifesto Ágil mas não pôde estar presente (ANDERSON, 2011). Em dois de seus livros, Jim Highsmith (2002; 2000) que é um dos signatários do Manifesto, categoriza o Lean Development como um método ágil por entender que trazia os mesmos ideais do Scrum, FDD, XP, etc, e que estava totalmente alinhado com o conceito de agilidade: “a habilidade de criar e responder a mudança de forma a obter lucro em um ambiente de negócios turbulento” (HIGHSMITH, 2002, p. xxiii).

O Lean Development é composto de 12 princípios, os quais possuem diversas similaridades com o Manifesto Ágil (CAWLEY, WANG e RICHARDSON, 2010). O primeiro princípio diz que a satisfação ao cliente é a maior prioridade da organização. Isso significa que um time que implementa o Lean Development deve se concentrar em determinar a proposição de valor do negócio, com o objetivo de maximizar a satisfação do cliente por meio da criação de valor em um nível aceitável de qualidade técnica. Nesse caso, não corresponder às expectativas de valor do cliente significa o fracasso do projeto.

O segundo princípio é sempre proporcionar o melhor valor em retorno ao que é pago pelo cliente. Os sistemas de informação devem ajudar a eliminar o risco de um cliente, resolver um problema, ou fornecer uma nova oportunidade a um custo razoável. O valor é uma combinação de características do produto que atendem às necessidades do cliente em um momento específico, em uma situação específica, a um determinado preço. Esse conceito está relacionado ao ROI de um projeto de software (HIGHSMITH, 2002).

Terceiro princípio diz que o sucesso depende da participação ativa do cliente. A participação ativa deve ser um esforço conjunto de colaboração de todos os envolvidos, sendo essencial a participação do cliente para a tomada de decisão. Esse princípio é muito similar ao princípio ágil de comunicação e participação ativa do cliente (BECK, *et al.*, 2001).

O trabalho em equipes multidisciplinares é o quarto princípio do Lean Development e encoraja o uso desses times ao invés de indivíduos isolados, por acreditar que a diversidade, além de fomentar a inovação, é essencial para a redução do tempo de ciclo de entrega (*cycle time).* Essa prática também é bastante difundida pelo Scrum, o mais utilizado dos métodos ágeis (LO GIUDICE, KISKER e ANGEL, 2014; VERSION ONE, 2014; MELO, *et al.*, 2013).

O quinto princípio denota outra característica chave dos métodos ágeis, a suscetibilidade à mudança (BECK, *et al.*, 2001). Para sistemas de software serem úteis, eles devem atender rapidamente às mudanças nas condições do negócio. O Lean Development assume que mudanças de requisitos serão constantes e que se adaptar a elas é uma estratégia mais interessante do que tentar controlá-las. O plano de projeto não deve tentar responder a todas as perguntas de forma rígida, mas estar preparado para se adaptar às perguntas que venham a surgir.

O Lean Development apresenta no sexto princípio que soluções de domínio são prefiríveis a soluções pontuais. Isso é devido a sistemas de software específicos serem muito caros na maioria dos casos, enquanto sistemas de software que são aplicáveis em vários domínios, empresas, mercados e produtos, pulverizam o custo e contribuem para a equação de valor.

A construção de sistemas de software é demorada, custosa e propensa a erros. Comprar ao invés de construir é uma estratégia viável para a maioria dos grupos de desenvolvimento de aplicações. O sétimo princípio diz que, sempre que possível, deve-se usar elementos de software pré-existentes ou pré-fabricados para minimizar os custos de oportunidade da organização.

O oitavo princípio refere-se a agilidade de resposta ao mercado, e sintetiza a idéia principal de outro método *lean*, o Lean Startup. Os mercados estão se movendo muito rápido para que o time gaste tempo fornecendo soluções 100% prontas aos clientes (RIES, 2011).

O Lean Development, como diz o nono princípio, busca eliminar a sobrecarga desnecessária e o desperdício, minimizando a burocracia, mantendo equipes de desenvolvimento pequenas e co-localizadas, e mantendo o foco do escopo do produto. Uma característica minimalista, também vista nos métodos ágeis (BECK, *et al.*, 2001).

O décimo princípio diz que se deve priorizar os objetivos do projeto e então a tecnologia para suportá-lo, e não vice-versa. As opções tecnológicas hoje são tão grandes que é fácil passar mais tempo migrando tecnologias do que criando valor para o negócio do cliente.

Uma outra prática ágil muito comum é vista no penúltimo princípio. A medida primária do progresso no Lean Development é o software em funcionamento, e não o crescimento no número de linhas de código (BECK, *et al.*, 2001). Ao avaliar as funcionalidades, o time deve sempre considerar como a dinâmica do negócio pode mudar e como será afetada pelo aplicativo de software.

Porfim, Robert Charette conclui com o último princípio: "o Lean Development visa reduzir custos e aumentar os lucros por meio da criação de maior valor para o cliente, porém se há formas melhores de fazê-lo, faça" (CHARETTE, 2002).

2.2.5 Métodos Lean Aplicados

A expansão originada na revolução industrial contribuiu para que, desde o fim do século XIX, áreas como a manufatura, a construção civil e a logística estruturassem grandes arcabouços de conhecimento sobre como melhorar o desempenho da produção. A engenharia de software tentou usufruir desse conhecimento modelando suas práticas como uma metáfora da engenharia civil e da manufatura. Porém, os resultados obtidos em projetos de software que seguiam essa aplicação eram inconsistentes, pois ora se mostravam satisfatórios e ora desastrosos. Isso aconteceu em parte por um entendimento limitado da real natureza dessas disciplinas e também pela falha em reconhecer até onde essa metáfora era válida (POPPENDIECK e POPPENDIECK, 2003).

Nesse contexto, uma metáfora significa aplicar conceitos bem-sucedidos de uma disciplina em outra, a partir do compartilhamento e transferência de princípios e práticas. Princípios são ideias e conceitos que guiam uma disciplina, enquanto práticas são atividades executadas para implementar os princípios (SENGE, 1990). Princípios são universais e generalistas, mas não são necessariamente fáceis de aplicar em determinado ambiente.

Práticas, por sua vez, expressam orientações específicas do que fazer, mas que necessitam de uma adaptação a um domínio. Uma "melhor prática" só pode ser encarada dessa forma para um determinado contexto. Mudando-se o contexto ou o domínio, a "melhor prática" perde sua força, sendo necessária sua revalidação (LARMAN e VODDE, 2010). De fato, problemas que são gerados da aplicação de metáforas para o desenvolvimento de software são usualmente resultado da tentativa de transferir as práticas de outra disciplina ao invés dos princípios (POPPENDIECK e POPPENDIECK, 2006).

2.2.6 Lean Software Development

Para Mary Poppendieck, uma programadora e gerente de TI, o desenvolvimento de software era similar ao desenvolvimento de produtos em diversos aspectos e a indústria de software poderia aprender muito ao examinar os princípios da engenharia de produtos e as melhorias que foram criadas e aplicadas a esse processo. Mary pôde ver o desenvolvimento de produtos *Lean* na prática durante os muitos anos em que trabalhou na empresa 3M.

Os princípios do desenvolvimento de produtos *Lean* foram testados e aprovados na indústria automotiva, que era parte de um ambiente complexo em que as etapas produtivas eram dependentes e altamente acopladas. A teoria por trás desse desenvolvimento em empresas como a Toyota estava pautada nos mesmos princípios regentes em sua linha de produção, e por isso essas empresas galgavam resultados tão acima da concorrência em ambas as disciplinas: desenvolvimento de produto (*Lean Development*) e fabricação do produto (*Lean Manufacturing*).

Em 2003, Mary e Tom Poppendieck escreveram um livro sobre como aplicar os princípios *Lean* de fabricação e criação de produtos no desenvolvimento de sistemas de informação. Seu livro "*Lean Software Development: An Agile Toolkit*" se tornou a principal publicação do gênero e deu origem ao método de desenvolvimento de software com o mesmo nome.

Como defendido pelos autores, o LSD (Lean Software Development) tem um grande enfoque em princípios ao invés de práticas. Os autores, em poucos momentos, fazem menção a técnicas de implementação e aplicações práticas, tornando seu método pouco preditivo. Os 7 princípios do LSD estão listados a seguir e percebe-se neles a paridade com o *Lean Thinking*:

Quadro 1 – Os 7 princípios do Lean Software Development

Eliminar Desperdício
Integrar Qualidade
Criar Conhecimento
Adiar Comprometimentos
Entregar Rápido
Respeitar as Pessoas
Otimizar o Todo

Fonte: (POPPENDIECK e POPPENDIECK, 2003, p. 66)

Desperdício é tudo aquilo que não adiciona ao produto um valor que é percebido pelo cliente. Se um componente fica em uma prateleira juntando poeira e esperando algo acontecer, isso é desperdício. Se uma fábrica produz mais peças do que o imediatamente necessário, é desperdício. Se um desenvolvedor de software codifica mais funcionalidades do que as imediatamente necessárias, é desperdício. Na manufatura, mover o produto de um lado para o outro é desperdício. No software, passar o código de uma pessoa ou grupo para outra, é também desperdício. O ideal é conhecer o que o cliente deseja e entregar exatamente aquilo, se possível imediatamente. Tudo o que impede isso de ser implementado rapidamente é desperdício.

A Toyota transferiu seu conceito de desperdício da produção para o desenvolvimento de produto. Quando o desenvolvimento de um produto é iniciado, o objetivo é completá-lo o mais rápido possível e a forma mais eficiente de fazê-lo é eliminando atividades ou etapas inteiras que não adicionam valor ao cliente. Em resumo, "se dá para fazer sem, então é desperdício" (POPPENDIECK e POPPENDIECK, 2006).

Aprender a enxergar o desperdício é o primeiro passo na implementação do LSD. A exemplo disso, mesmo antes da criação do LSD, Winston Royce escreveu que enquanto "muitas etapas adicionais de desenvolvimento [de software] são requeridas, nenhuma delas contribui tão diretamente para o produto final quanto análise e codificação" (ROYCE, 1970). Dessa forma, ele indicava que das etapas do modelo *waterfall*, análise e codificação eram as que mais agregavam valor, enquanto as demais imputavam certo grau de desperdício ao processo.

Os idealizadores do TPS, por sua vez, classificaram sete tipos de desperdício na produção (OHNO, 1988), o que serviu como guia e ajudou muitos gerentes a identificar perdas onde nunca haviam pensado existir. Com o mesmo intuito, Poppendieck fez uma adaptação, nomeando os sete desperdícios na engenharia de software, conforme Quadro 2.

Quadro 2 – Relação entre os desperdícios da Produção e da Engenharia de Software

Os 7 desperdícios da Produção	**Os 7 desperdícios da Engenharia de Software**
Superprodução	Funcionalidades extras
Espera	Espera
Transporte	Alternância de tarefas
Superprocessamento	Processos extras
Estoque	Trabalho Parcial

Movimentação	Movimentação
Defeitos	Defeitos

Fonte: Poppendieck (2006, p. 4)

Em meio ao processo de desenvolvimento parece uma boa ideia adicionar uma funcionalidade não planejada ao sistema para o caso do cliente necessitá-la, ou seja, uma funcionalidade extra. Isso pode parecer uma ação inofensiva, porém é um desperdício. Toda porção de código que é escrita deve ser mapeada, compilada, testada, integrada e manutenida por todo o ciclo de vida do sistema. Cada nova linha de código inserida, aumenta a complexidade do software e potenciais pontos de falha. Há uma grande chance da funcionalidade extra se tornar obsoleta pois não houve uma requisição real para tal. Se o código não é necessário agora, integrá-lo ao sistema é um desperdício.

Ao observar o conceito de Espera, pode-se compreender porque é considerada um dos maiores desperdícios no desenvolvimento de software. Espera no início de um projeto, na formação do pessoal, em revisões e aprovações, nos testes e no desenvolvimento são desperdícios. É natural pensar que o tempo de resposta e consequentemente a entrega de valor ao cliente são afetados em decorrência da espera e dos atrasos sistêmicos dentro do ciclo de desenvolvimento. No momento que uma tarefa entra em uma fila para ser processada, um atraso é adicionado ao prazo de finalização da mesma. Na engenharia de software, isso toma proporções maiores dado que a natureza do processo é cíclica e com etapas interdependentes.

O conceito de desperdício por alternância de tarefas pode ser entendido quando são delegadas múltiplas tarefas ou projetos a uma pessoa. Cada vez que um desenvolvedor de software alterna a execução de tarefas, decorre um tempo significativo de troca de contexto e agrupamento de ideias para entrar no fluxo da nova demanda (DEMARCO e LISTER, 1999).

A maneira mais rápida de completar dois projetos que usam os mesmos recursos é fazê-los com foco, um de cada vez. Por exemplo, um determinado projeto é executado em duas semanas. Se dois projetos são executados em sequência, ao findar a segunda semana um projeto estará finalizado e ao término da quarta semana o segundo. No entanto, se os dois projetos forem iniciados juntos alternando-se a execução das tarefas, nenhum dos dois estará pronto em duas semanas, postergando a entrega de valor ao cliente. Além disso, o tempo desperdiçado com a alternância fará com que o prazo final da execução seja próximo de cinco semanas (GOLDRATT, 1997). Trabalha-se mais rápido quando se estabelece um fluxo de atividades sendo executadas uma-a-uma, o que reforça o princípio *Lean* do *one-piece flow*.

Portanto, como afirmado por Mary e Tom Poppendieck (2006), a dimuição do estoque intermediário e das tarefas executadas ao mesmo tempo, expõem as imperfeições do processo, da mesma forma que o nível de um rio quando está baixo expõe tudo que está alojado em seu leito que impede a livre navegação.

A produção de documentação deve sempre ser questionada. Criar diagramas, documentos explicativos e descritivos consome tempo e recursos que poderiam ser empregados na produção direta de valor ao cliente. Não é descartada totalmente a utilização de documentos, porém o uso excessivo é indevido pois promove ativos com grande tendência a ficarem desatualizados, obsoletos e a caírem no esquecimento por serem pouco consultados, o que denota a necessidade de avaliar tais processos extras.

Muitas vezes as documentações fazem parte da entrega e do acordo comercial com o cliente. Nesse caso é necessário um trabalho de conscientização de forma a concentrar esforços e recursos para a entrega software funcionando em detrimento de papeladas que serão pouco ou quase nunca utilizadas.

O software desenvolvido parcialmente tem a tendência a se tornar obsoleto e ocupa a capacidade produtiva do time sem entregar valor ao cliente. Até que a funcionalidade seja integrada ao sistema, não há a certeza de sua qualidade e nem a ativação do valor que ela propõe, tornando-se um "estoque" de software, um produto não acabado. Minimizar o desenvolvimento parcial de software é uma estratégia de redução de risco assim como redução de desperdício e uso efetivo da capacidade produtiva.

No caso de movimentação, quando um desenvolvedor tem uma questão, quanto esforço ele precisa fazer para encontrar a resposta? Os stakeholders estão à disposição para sanar dúvidas ou definições técnicas? O cliente está disponível para tomar decisões de negócio sobre as funcionalidades e o comportamento do sistema? Na maioria das vezes, essa movimentação não é física. Perde-se tempo com a falta da informação em mãos e a movimentação acontece na forma de e-mails, telefonemas e reuniões. Os artefatos de software também se movimentam. Os requisitos se movimentam dos analistas aos arquitetos, que movimentam documentos aos desenvolvedores, que por sua vez movimentam código fonte aos testadores e assim por diante. Cada *hand-off* de um artefato é uma oportunidade para o aparecimento de filas, espera e desperdício.

O desperdício causado por um defeito está relacionado ao impacto e ao tempo que ficou sem ser identificado. Um defeito crítico que foi identificado em 3 minutos não é uma causa grande de desperdício. Um defeito menor que ficou sem ser descoberto por semanas é causador de um desperdício muito maior. Para mitigar o impacto dos defeitos é necessário prover mecanismos para detectá-los assim que ocorrem. Assim, a maneira de reduzir o desperdício com defeitos, além de fazer certo da primeira vez, é testar imediatamente, integrar o código com frequência e lançar em produção o mais breve possível.

O objetivo do Lean Software Development é construir a qualidade dentro do código desde o início e não testar a qualidade ao final do processo. O foco não é relacionar o máximo de defeitos e depois corrigí-los, mas sim evitar de criá-los. De acordo com Shingeo Shingo (1996), existem dois tipos de inspeção: inspeção depois de os defeitos ocorrerem e inspeção para prevenir os defeitos. É prioritário controlar as condições para não permitir que os defeitos ocorram, evitando o desperdício com retrabalho e reparo. Se não for possível evitá-los com eficiência, uma forma de proteger o processo é inspecionar o produto a cada pequeno passo, de forma a capturar o defeito imediatamente após sua ocorrência. A sequência ideal à ocorrência de um defeito é sua correção imediata e o estudo da causa raiz que o originou.

Desenvolvimento é um exercício de descobrimentos, enquanto produção é um exercício de redução da variação. Para esse propósito, a abordagem *Lean* para desenvolvimento resulta em práticas um tanto diferentes das práticas *Lean* de produção. Desenvolvimento é como criar uma receita enquanto a produção é como seguir uma receita. Receitas são criadas por cozinheiros experientes que desenvolveram um instinto do que funciona e da capacidade de adaptar ingredientes para cada ocasião. No entanto, os grandes chefs produzem diversas variações de um novo prato enquanto aprendem a receita. Isso é parte integrante do processo de aprendizado e permite que cada chef coloque sua marca pessoal no prato. O desenvolvimento de software é melhor concebido como um processo de aprendizado similar, porém com um desafio adicional de os times de desenvolvimento serem maiores e os resultados muito mais complexos que uma receita de cozinha.

Práticas de desenvolvimento que promovem a tomada de decisão tardia são efetivas em domínios que envolvem incertezas, porque compõe uma abordagem orientada à opções. Em face a incerteza, muitos mercados provêem formas de o investidor evitar estar preso a uma decisão sobre o futuro, até que esteja mais próximo e mais fácil de prever. Postergar decisões é favorável pois melhores decisões podem ser tomadas quando baseadas em fatos e não em especulação. Num mercado em evolução, manter as opções do projeto em aberto é mais valioso do que o comprometimento prévio. Uma estratégia chave para o atraso no comprometimento ao desenvolver sistemas complexos é a promoção da capacidade de adaptação do sistema.

Até pouco tempo atrás, o desenvolvimento rápido de software não era valorizado, fazendo com que uma abordagem de não cometer erros fosse mais importante. O desenvolvimento rápido traz diversas vantagens. Sem a velocidade não é possível atrasar as decisões, nem obter respostas rápidas e confiáveis. Em desenvolvimento, o ciclo de descobrimento é crítico ao aprendizado: projetar, implementar, avaliar e melhorar. Quanto mais curtos são esses ciclos, maior é o aprendizado. A velocidade garante que os consumidores recebam aquilo que eles precisam agora, não o que precisavam ontem. E também permite que eles atrasem a tomada de decisão sobre o que realmente desejam até que tenham mais entendimento e visão. Comprimir o fluxo de valor o máximo possível é uma estratégia fundamental para a eliminação do desperdício.

A execução de alto nível baseia-se na implementação correta dos pequenos detalhes e ninguém conhece melhor os detalhes do que pessoas que trabalham com isso diariamente. Envolver os desenvolvedores nos detalhes das decisões técnicas é fundamental para alcançar a excelência. O grupo de pessoas que estão na linha de frente combina o conhecimento dos mínimos detalhes com o poder do consenso. Quando equipados com o conhecimento necessário e guiados por um líder, eles vão tomar decisões técnicas e de processo melhor do que qualquer um. As decisões sendo tardias e a execução sendo rápida, não é possível a uma autoridade centralizada orquestrar as atividades dos trabalhadores. Assim, as práticas *Lean* usam técnicas de trabalho puxado que contém mecanismos que assinalam localmente o que precisa ser feito de forma que todos os integrantes saibam o que precisa ser feito. No Lean Software Development o sistema puxado é um acordo para entregar versões de software constantemente em intervalos regulares. Exemplos de gestão à vista por meio de sinalização visual podem ser por quadros, gráficos, reuniões diárias, integração contínua, dentre outros.

Uma organização *Lean* otimiza todo o fluxo de valor da hora em que recebe o pedido de um cliente até o software ser entregue e a necessidade do cliente suprida. Se a organização focar em otimizar algo menor do que toda a cadeia de valor, o que é chamado de subotimização, o benefício potencial ao cliente estará sendo desperdiçado dentro do próprio fluxo.

2.2.7 Método Kanban

Como visto, os princípios *Lean* foram desenvolvidos na Toyota tendo grande expansão na década de 1950. Dentre as técnicas implementadas nesse aperfeiçoamento do processo produtivo, estava o sistema *kanban*. *Kanban* é uma palavra japonesa que significa cartão visual e representa uma ferramenta de processo da indústria de manufatura para gerenciar a execução de atividades, sendo um mecanismo de controle do fluxo puxado e base da produção *just-in-time*.

> "... *Kanban* (K maiúsculo) é um método de mudança evolucionário que utiliza o sistema puxado kanban (K minúsculo), a visualização e outras ferramentas para catalizar a introdução de ideias *Lean* no desenvolvimento de tecnologia e operações de TI." (ANDERSON, 2010)

O Método Kanban na engenharia de software surgiu no ano de 2004. David Anderson, um gerente de desenvolvimento que atuou nas empresas Motorola, Microsoft e Corbis, desde 2002 buscava por metodologia própria uma maneira eficiente de aumentar a produtividade e a qualidade do seu time, de forma a cumprir as demandas de negócio (ANDERSON, 2010).

Para isso, estabeleceu um quadro visual que fornecia visibilidade de todo o processo de software: as demandas que estavam por vir, os tipos de atividades, o trabalho que estava sendo executado por cada integrante do time, a comunicação clara de prioridades e a rápida exposição de bloqueios e gargalos.

Para o desenvolvimento do Método Kanban, Anderson contou com diversas influências dentre elas os Métodos Ágeis de desenvolvimento de software (BECK, *et al.*, 2001), a Teoria das Restrições (TOC) (GOLDRATT, 1990), a Teoria das Filas (LITTLE, 1961), o pensamento sistêmico (SENGE, 1990; DEMING, 1998) e, principalmente, os valores do *Lean Thinking* (WOMACK e JONES, 2010).

David enumerou 6 valores básicos que guiam a implementação do Método Kanban. Como visto no Quadro 3, de forma clara pode-se traçar uma relação dos valores do Kanban com princípios e ferramentas presentes no *Lean Thinking*.

Quadro 3 – Alinhamento entre o Método Kanban e princípios e ferramentas *Lean*

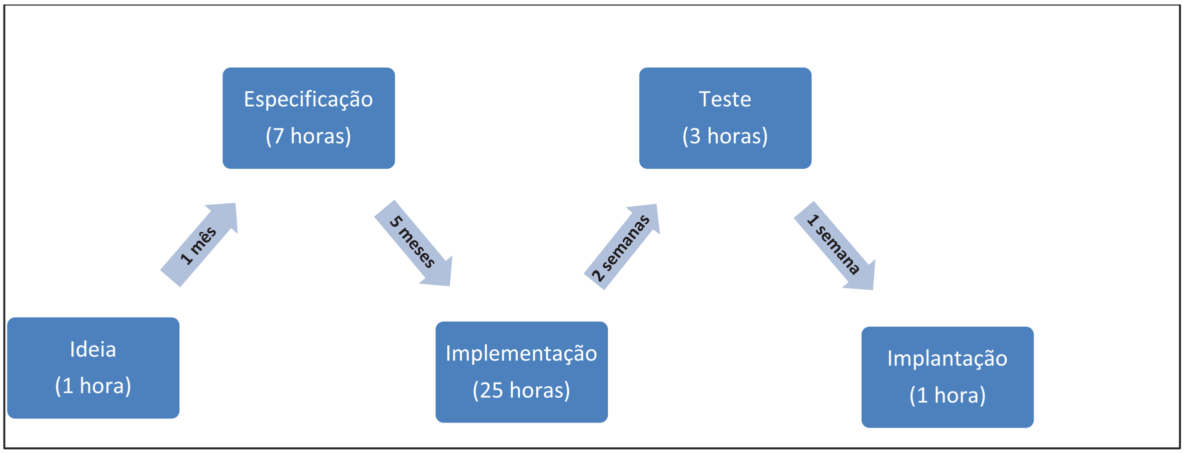
Valores do Método Kanban	Princípios *Lean Thinking*
Visualizar Fluxo de Trabalho	Fluxo de Valor (VSM)
Limitar Trabalho em Progresso	*One-piece Flow* e Sistema Puxado (JIT)
Medir e Gerenciar Fluxo	Fluxo Contínuo
Explicitar Políticas	Valor e Fluxo de Valor (VSM)
Implementar Loops de Feedback	Perfeição (Kaizen)
Melhorar de Forma Colaborativa	Perfeição (Kaizen)

Fonte: elaborado pelo autor

O primeiro valor do Método Kanban é a visualização do fluxo de trabalho, pois em atividades do conhecimento como desenvolvimento de software essa é uma prática de difícil percepção. Ter a ciência de todas etapas do trabalho e torná-lo visível é fundamental para o entendimento mais profundo de como ele é feito, além de facilitar o processo de tomada de decisão.

A visualização do fluxo de trabalho é totalmente análoga ao princípio *Lean* do Mapeamento de Fluxo de Valor (VSM), que em sua forma mais simples é a exposição dos estágios de execução do trabalho desde a matéria-prima ao produto acabado, ou, no caso do desenvolvimento de software, de uma vaga ideia à uma funcionalidade sendo usada pelo cliente, conforme apresentado na Figura 11.

Figura 11 – Exemplo de Fluxo de Trabalho (desenvolvimento de software)



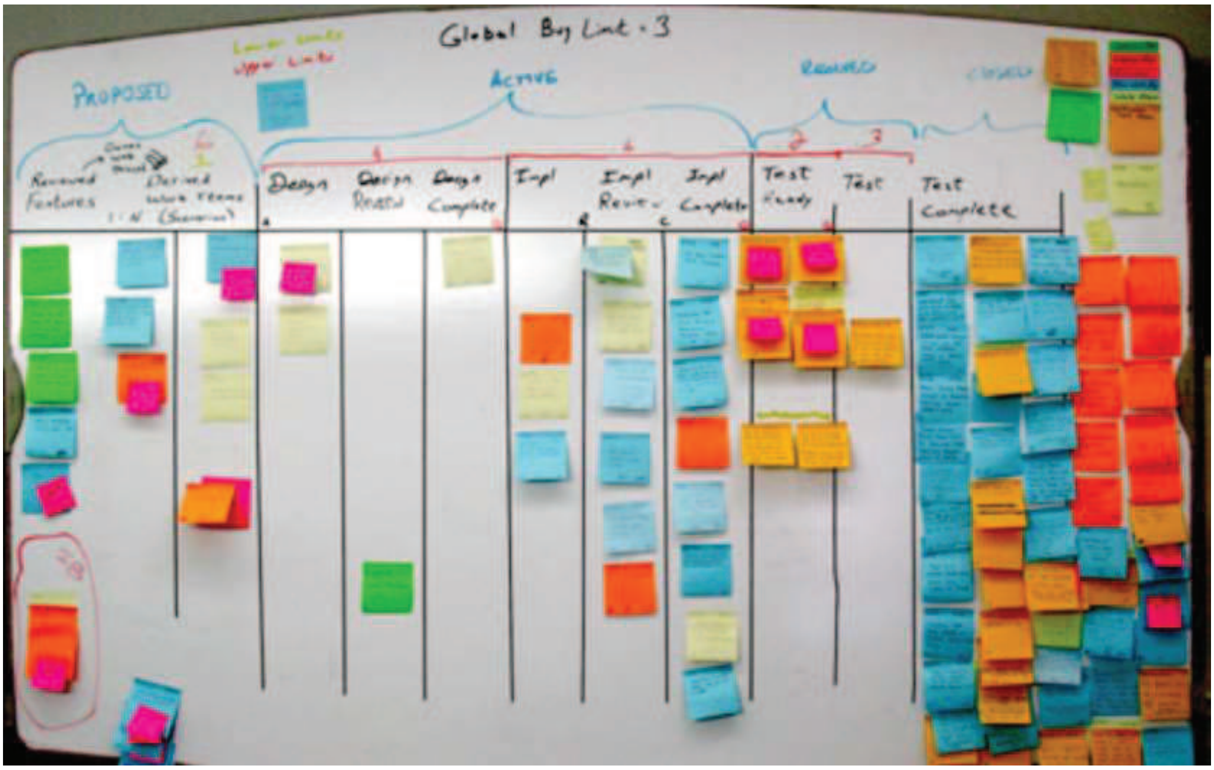
Fonte: (BOEG, 2011, p. 22)

Ao representar o fluxo de trabalho, o processo objeto da avaliação fica explícito, sendo conhecidas as fronteiras de cada etapa, seus *inputs* e *outputs*, é possível questionar o porquê de cada atividade existir, quem é responsável por executá-las e qual tempo médio de execução e espera em toda a cadeia de produção.

Uma maneira comum de visualizar o fluxo de trabalho no dia-a-dia é utilizando-se de cartões *post-it* afixados num quadro branco. Em geral, os quadros podem ser virtuais ou físicos, como pode ser visto na Figura 12, e as colunas representam os diferentes estados ou etapas do fluxo e os cartões as atividades a serem executadas. No decorrer do processo, os cartões vão sendo movimentados (geralmente da esquerda para direita), representando o fluir das atividades da concepção à entrega.

Porfim, não existe um padrão rígido para o desenho de um quadro Kanban uma vez que a implantação de um processo de desenvolvimento de software é complexa, composta de N variáveis, o que torna o fluxo de trabalho e de sua visualização (quadro) muito dinâmicos para responder a uma regra fixa pré-determinada.

Figura 12– Visualização do fluxo de trabalho por meio de um quadro físico



Fonte: (ANDERSON, 2010, p. 14)

O segundo valor do *Kaban* que é considerado a marca registrada do método, estabelece que se deve limitar a quantidade de trabalho executado ao mesmo tempo (WIP – *work in progress*). Segundo Jesper Boeg (2011), esse conceito foi derivado da Lei de Little que diz que o tempo médio para um item percorrer o sistema (*lead time*) é igual a quantidade média de itens em progresso (WIP) dividido pela quantidade média de itens produzidos no período (*throughput*). Isso pode ser visto Equação 2.

$$Lead\ Time = \frac{WIP}{Throughput} \qquad (2)$$

Isso significa que, num sistema que possui em média 10 atividades em progresso (WIP) e um *throughput* de 2 atividades por dia, afere-se um *lead time* médio de 5 dias. Dado o objetivo de diminuir o tempo gasto para finalizar uma tarefa, ou seja, ser mais eficiente, pode-se trabalhar de duas formas: aumentando o *throughput* ou dimuindo o WIP. Na maioria dos casos, a última opção é mais fácil de implementar (BOEG, 2011).

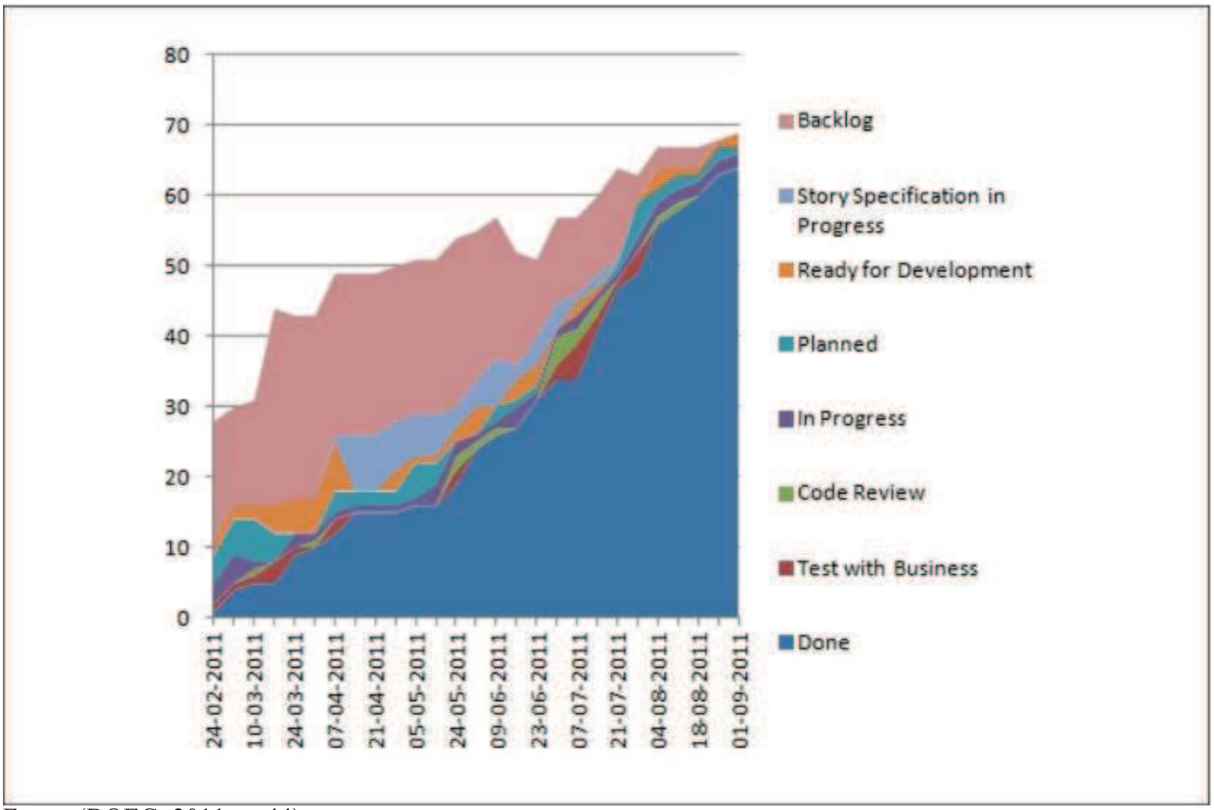
Os limites no processo produtivo se assemelham a *slots* de trabalho para cada uma de suas etapas e, quando aplicados em conjunto, implementam um sistema puxado em partes ou na totalidade do processo. A imposição de um limitador por meio de um sistema puxado atua como um dos principais estímulos para melhoria contínua, incremental e evolutiva do processo. Os elementos a serem observados são o número de atividades em progresso em cada estado do fluxo, fazendo que as novas atividades sejam puxadas somente quando há capacidade disponível dentro do limite estabelecido.

Existe uma extensa associação do limite de WIP com as práticas e princípios visto no TPS e no LSD. Ao limitar o trabalho em progresso é possível controlar o estoque de produtos não acabados e evitar a alternância de tarefas, o que é visto com bons olhos pelos praticantes do *Lean*. Impor tal restrição também é essencial quando se tem o objetivo de implantar o JIT (*one-piece flow*) por meio de um sistema puxado.

O terceiro valor diz que o fluxo de trabalho deve ser monitorado, medido e reportado para que os gestores possam coordená-lo ativamente, implantando mudanças contínuas, graduais e evolutivas no sistema. Não se pode simplesmente implementar uma mudança sem validar sua eficácia e avaliar seus efeitos positivos ou negativos sobre o método.

Uma ferramenta muito difundida pelos praticantes do Método Kanban é o CFD (Cumulative Flow Diagram ou Diagrama de Fluxo Cumulativo) e pode ser visto na Figura 13. Esse diagrama auxilia os gestores e o próprio time a monitorar e avaliar como as decisões e políticas de processo estão sendo refletidas no fluxo de trabalho. Por meio dele, é possível enumerar a quantidade de tarefas em cada etapa do fluxo de valor, o tempo médio de execução, o comportamento das filas e a fluidez do trabalho.

Figura 13 – Exemplo de Diagrama de Fluxo Cumulativo (CFD)



Fonte: (BOEG, 2011, p. 44)

Segundo Anderson (2010), até que o resultado de um processo seja explicitado por meio de métricas, é difícil ou até mesmo impossível discutir sobre como melhorá-lo. Sem um entendimento detalhado de como as coisas funcionam e como o trabalho é feito, qualquer debate sobre problemas tende a ser anedótico, subjetivo e emocional. Com dados expressos por meio de métricas é possível implementar uma discussão mais racional, empírica e objetiva na solução de problemas e evolução do processo, fazendo com que seja mais provável o consenso em torno de sugestões de melhoria (ANDERSON, 2010).

O quarto valor diz que é essencial o estabelecimento claro das políticas do processo para garantir consistência e qualidade. Em termos práticos, isso significa especificar regras e *checklists* para serem conferidos ao passar pelas colunas do quadro, de modo a garantir a qualidade em cada etapa do processo.

O quinto valor é também um princípio muito utilizado no universo dos Métodos Ágeis. Uma vez organizadas as políticas e coletadas as métricas, devem ser estabelecidos *loops* de *feedback* curtos que possam fomentar uma cultura de mudanças evolutivas proporcionadas pela ação e análise da reação. Os *loops* de *feedback* estabelecem um ciclo que alimenta o sistema com dados auxiliando o processo de melhoria.

Quando os times têm um entendimento das teorias sobre o trabalho, do fluxo, do processo e dos riscos, eles são mais capazes de construir uma compreensão compartilhada de um problema e sugerir ações de melhoria que possam ser aprovadas por consenso. O Método Kanban estimula pequenas mudanças: contínuas, incrementais e evolutivas por meio do empirismo e da ação colaborativa e experimental. Ou seja, os membros do time são as pessoas mais indicadas a propor ações de melhoria do processo, e nesse caso é preconizado o método científico de validação de hipótese para colocá-la em prática (ANDERSON, 2010).

Quanto às ferramentas de tralhado utilizadas na implantação do método, o quadro Kanban é a principal, utilizado para visualizar e coordenar o trabalho, da concepção à entrega ao cliente. Suas colunas mostram a sequência das atividades desempenhadas pelo time e os cards representam as tarefas em progresso e a serem executadas, o que promove transparência e comunicação.

Para cada atividade (coluna) existe um número limite de cartões que podem ser executados em paralelo. Isso é chamado de WIP (*work in progress*) ou atividades em progresso e determinam a quantidade de tarefas que foram iniciadas e não terminadas, estando em execução efetiva ou em filas. Tal visualização é importante para favorecer a sinergia entre os envolvidos, estabelecendo uma conduta aderente ao princípio *Lean* de fluxo contínuo e do JIT, ou seja, uma pessoa deve completar o trabalho existente antes de iniciar uma nova atividade. Dessa forma, um valor chave é atingido: o número reduzido de software inacabado e o controle sobre o estoque.

Outras variantes do quadro podem incluir raias horizontais que representam classes de serviço e avatares de mostram onde cada membro do time está trabalhando. As tarefas podem ser apresentadas em cartões de cores diferentes (determinando tipo de atividade) e a disposição no quadro estabelece a prioridade e ordem de execução (quanto mais acima, maior a prioridade).

O uso de estatísticas e diagramas para monitorar o processo é parte integral da abordagem Kanban. As métricas mais comuns são *lead time, cycle time, touch time,* *throughput*, número de defeitos, taxa de retrabalho e taxa de eficiência do processo. Esses dados são geralmente compilados em gráficos e tabelas, e são constantemente apresentados ao time para fomentar a discussão, auto-análise e as ações de melhoria (kaizen). Os diagramas mais comuns são o CFD, que traz visibilidade à quantidade de WIP, a velocidade do time e os gargalos do processo; acompanhado do Control Chart (Gráfico de Controle), representação assimilada da Engenharia da Produção que informa a data de finalização, o tempo gasto na execução e as médias históricas das atividades (BOEG, 2011).

3 METODOLOGIA

Este capítulo apresenta a metodologia utilizada na produção do trabalho. Metodologia significa o estudo do método, que por sua vez é o conjunto das atividades sistemáticas e racionais que permite alcançar um objetivo, traçando o caminho a ser seguido, detectando erros e auxiliando as decisões. A metodologia pretende responder às seguintes questões: Como? Com quê? Onde? Quanto? (LAKATOS e MARCONI, 2003)

3.1 Classificação da Metodologia

O presente trabalho pode ser classificado sob diversas óticas. Quanto a sua natureza, é uma pesquisa aplicada pois “objetiva gerar conhecimentos para aplicação prática e dirigidos à solução de problemas específicos” (SILVA e MENEZES, 2005).

Já do ponto de vista dos objetivos, é classificado como pesquisa exploratória pois tem como finalidade propiciar maiores informações sobre um tema e explicitar melhor determinado assunto. Pode-se dizer que estas pesquisas têm como objetivo principal o aprimoramento de ideias ou a descoberta de intuições. Seu planejamento é, portanto, bastante flexível, de modo que possibilita a consideração dos mais variados aspectos relativos ao fato estudado. Na maioria dos casos, essas pesquisas envolvem: (a) levantamento bibliográfico; (b) entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; e (c) análise de exemplos que estimulem a compreensão (SELLTIZ, WRIGHTSMAN e COOK, 1987). Segundo Gil (1999), pesquisas desse tipo, em geral, assumem a forma de pesquisas bibliográficas e estudos de caso.

A presente pesquisa também é uma pesquisa-ação, que segundo Thiollent (1988)

> ... é um tipo de investigação social com base empírica que é concebida e realizada em estreita associação com uma ação ou com a resolução de um problema coletivo no qual os pesquisadores e os participantes representativos da situação ou do problema estão envolvidos de modo cooperativo ou participativo (THIOLLENT, 1988)

Pode-se cotegorizar a pesquisa dessa forma pois o pesquisador participa como membro e líder do time em estudo, sendo suas ações:

- Treinar e capacitar os demais membros do time nos princípios e práticas da Filosofia *Lean*
- Sugerir ações para o aprimorando do fluxo de trabalho
- Garantir que as práticas diárias do time obedecessem ao fluxo estabelecido
- Coletar e informar o time sobre as métricas para avaliação do processo
- Ser o facilitador das reuniões informativas, avaliativas e de tomada de decisão

A maioria dos processos de desenvolvimento seguem um mesmo ciclo segmentado em 4 fases, inclusive o PDCA de Deming (1998) que influenciou o *Lean* e pode ser visto em práticas recorrentes como o *Kaizen*. Este ciclo se inicia com a identificação do problema e planejamento de uma solução, seguido de sua implementação, monitoramento e a avaliação de sua eficácia (TRIPP, 2005). Dessa forma, o próprio método de pesquisa pode cooperar com o desenvolvimento do método de trabalho objeto da pesquisa.

A ação do pesquisador se deu na sugestão das práticas adotadas pelo time em estudo e foi orientada pela aplicação dos princípios *Lean* ao desenvolvimento de software. Isso foi feito observando diversas pesquisas, dentre elas Morgan (1998), Poppendieck e Cusumano (2012), Mehta, Anderson e Raffo (2008), Subramanya (2012) e os estudos de caso de Middleton (2001; 2005; 2012), Suton (2008) e Widman, Hua e Ross (2010).

O universo de pesquisa da pesquisa-ação foi composto por membros de uma mesma empresa, sendo a amostra não-probabilística intencional, ou seja, escolhida por conveniência e com participação do pesquisador (SILVA e MENEZES, 2005).

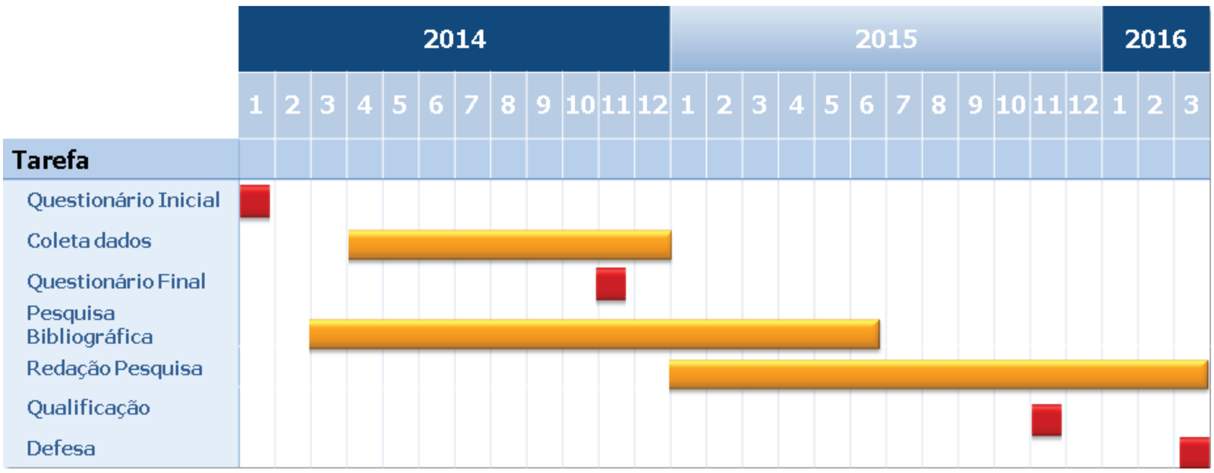
A coleta de dados foi feita por questionário de múltipla escolha aplicado aos membros do estudo de caso e também por observação direta assistemática (quando não há planejamento e controle previamente elaborados), feita a partir de um software que coletou as métricas de performance dos indivíduos em estudo (SILVA e MENEZES, 2005).

Por fim, quanto ao tratamento dos dados, essa é uma pesquisa qualitativa pois não segue um procedimento rígido para avaliação dos mesmos. Entre as principais características desta abordagem 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 por meio 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.

3.2 Aplicação da Metodologia

Quanto a aplicação, a pesquisa foi estruturada de acordo com as etapas apresentadas na Figura 14.

Figura 14 – Cronograma da Pesquisa



Fonte: elaborado pelo autor

a) Janeiro 2014 – Aplicação inicial do questionário de pesquisa

Como parte do processo de avaliação e desenvolvimento interno da empresa UOL S/A, as equipes de software responderam uma auto avaliação em janeiro de 2014.

O questionário requisitava que o respondente avaliasse o time que estava inserido observando as seguintes temáticas:

- *Delivery* contínuo e processo de *build*

- Boas práticas de desenvolvimento/codificação
- Utilização de tecnologia de ponta
- Aprimoramento dos processos
- Práticas para garantia da qualidade

As alternativas de resposta seguiam uma escala Likert para classificar individualmente o item avaliado segundo seu grau de maturidade de acordo com o Quadro 4.

Quadro 4 – Escala Likert para avaliação de grau de maturidade

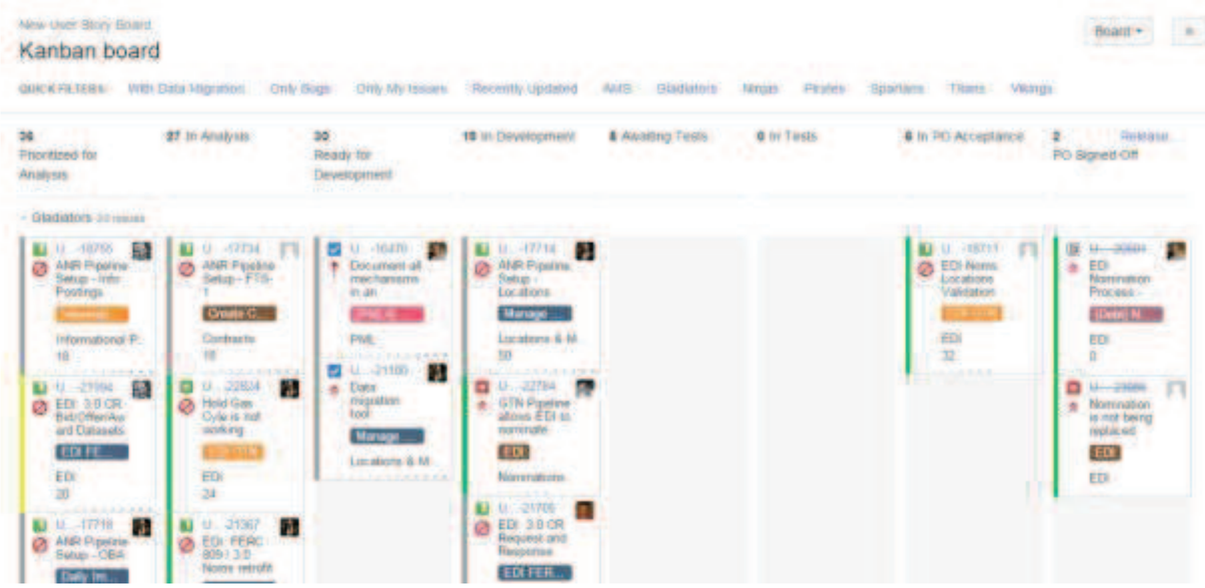
Nível	**Definição**
0 – Incompleto	Não fazemos, não existe
1– Inicial (Ad-hoc)	Fazemos às vezes e sem uma razão bem definida
2 – Gerenciado	Fazemos de forma recorrente como parte do fluxo de trabalho para alcançar um objetivo conhecido
3 – Definido	Fazemos sempre, como regra para mitigar riscos e garantir que mesmo em situações de stress o objetivo seja alcançado
4 - Quantitativamente gerenciado	Fazemos sempre, constantemente avaliando os resultados com base em objetivos quantitativos de qualidade e desempenho
5 – Em otimização	Fazemos sempre, constante aprimorando através de inovação e experimentos, avaliando os indicadores quantitativos de qualidade e desempenho, compreendendo as causas e efeitos de suas modificações

Fonte: elaborado pelo autor

b) Abril a Dezembro 2014 – Coleta dos dados de desempenho dos indivíduos pesquisados

A fase de desenvolvimento do projeto de software aconteceu no período de março a dezembro de 2014. Foi selecionado por conveniência do pesquisador um time já estabelecido, composto de 7 engenheiros de software. Nessa etapa, todas as atividades de análise, programação, testes, configuração e manutenção executadas pelos membros do time foram monitoradas pela ferramenta de gestão Jira da empresa Atlassian, que pode ser vista na Figura 15. Tal ferramenta é da categoria dos chamados *issue trackers,* utilizada como gerenciador de demandas do projeto, disponibilizando quadros virtuais, *dashboards* e relatórios customizados. Dessa forma, pôde-se compilar métricas de produtividade como quantidade de tarefas executadas, tempo total médio de execução, tempo médio em desenvolvimento, tempo médio em filas, quantidade de defeitos, etc.

Figura 15 – Exemplo de Quadro Kanban da ferramente Jira



Fonte: elaborado pelo autor

c) Novembro 2014 – Aplicação final do questionário de pesquisa

Ao término do projeto, foi aplicado novamente o mesmo questionário de auto-avaliação da maturidade do time, para qualificar as mudanças implementadas pelo processo *Lean* de melhoria contínua, executadas durante o ano de 2014.

d) Março 2014 a Junho 2015 – Realização de pesquisa bibliográfica

A realização da pesquisa bibliográfica foi efetuada tendo ênfase nos principais autores dos tópicos *Lean* e Lean Software Development. A exemplo, pode-se citar Womack, Ohno, Liker, Poppendieck, Sutton, Middleton e Anderson. O objetivo da pesquisa era, além do embasamento científico do estudo, conhecer o que já havia sido publicado no gênero, o método empírico utilizado e as conclusões alcançadas pelos autores.

e) Janeiro 2015 a Fevereiro 2016 – Redação da pesquisa, relatório e considerações finais

A redação da pesquisa foi feita durante o ano 2015 e início de 2016.

4 PESQUISA EMPÍRICA

Os dados publicados neste estudo foram coletados nos anos de 2012 e 2014 da empresa UOL S/A, sediada na cidade de São Paulo – SP. A UOL é a maior empresa brasileira de conteúdo, produtos e serviços de internet desde sua fundação em abril de 1996. Segundo a Omniture, possui o maior portal de conteúdo em língua portuguesa do mundo em quantidade de páginas e acessos. São 7,4 bilhões de páginas vistas e mais de 50 milhões de visitantes únicos por mês. Isso é equivalente a dizer que 8 em cada 10 internautas brasileiros acessam a home page do UOL (UOL SA, 2015).

Além do extenso conteúdo jornalístico disposto em mais de 1000 canais, a empresa é prestadora de serviços e comercializa produtos como e-mail, hospedagem de sites, armazenagem de dados, venda de publicidade, operação de meios de pagamento, segurança de dados, educação à distância, e-commerce, afiliados, dentre outros (UOL SA, 2015).

A publicação do nome da empresa UOL e dos dados apresentados nessa pesquisa, foi concedida por meio de declaração anexa ao trabalho.

4.1 Cenário Anterior

A presente pesquisa coletou métricas em duas frentes para analisar a efetividade das ações tomadas nesse período. A primeira frente teve por finalidade observar de forma geral a Gestão da Mudança, ou seja, da instalação de atualizações de software feitas no ambiente de produção e avaliar sua eficácia por meio da taxa de sucesso e do tempo médio dispendido. Nesse período, foram compilados os dados dos seis times de desenvolvimento que compunham o projeto.

A segunda frente de coleta foi mais específica e se concentrou nos dados de desempenho diário de 1 time escolhido dentre os 6 anteriores (escolha feita por conveniência do pesquisador). Os dados coletados foram a quantidade de tarefas finalizadas em cada período, o tempo médio gasto em cada etapa do fluxo de trabalho e a quantidade de itens trabalhados ao mesmo tempo. Desses dados pode-se derivar todas as métricas apresentadas no estudo.

Todos os times participantes foram oriundos do departamento de Pesquisa e Desenvolvimento (P&D) do UOL, e tinham a responsabilidade da análise, codificação, teste e entrega do software em produção de forma ágil e com baixo consumo de recursos.

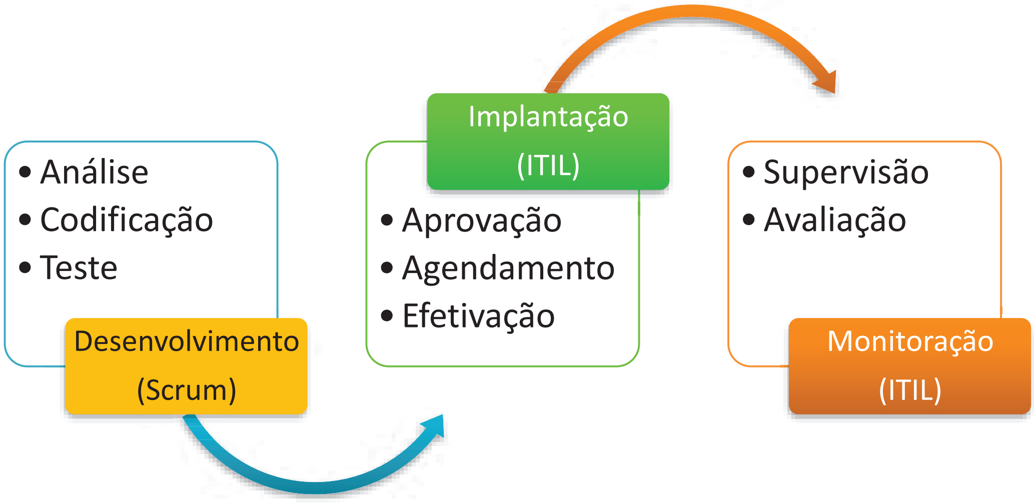
As atividades foram registradas de março a dezembro de 2014. O efetivo dos times era multidisciplinar, variando de 4 a 8 profissionais por time (programadores, web masters e analistas de qualidade), que não possuíam experiência prévia com a filosofia *Lean*, mas que aos poucos foram sendo introduzidos às práticas de mapeamento de fluxo de valor, eliminação de desperdícios, *one-piece flow*, sistema *kanban,* dentre outras.

Um outro time denominado Time de Operação era responsável pelo gerenciamento das aplicações e servidores de produção, e utilizava a rigor os preceitos do ITIL (Information Technology Infrastructure Library) para a manutenção e segurança dessa infraestrutura.

Os times mencionados trabalhavam em conjunto na criação, manutenção e sustentação de um produto de *e-commerce* chamado Shopping UOL (vitrine virtual de comparação de preços do varejo), assim como a criação do UOL Marketplace que era uma plataforma de comercialização de produtos integrando dois sistemas do UOL – o Shopping UOL e o serviço de pagamento on-line Pagseguro.

As atividades diárias se dividiam em etapas de desenvolvimento (análise de requisitos, codificação de serviços e interfaces; e testes de software), implantação (preparação e efetivação da mudança em produção) e a monitoração (supervisão do produto). Como pode ser visto na Figura 16, utilizava-se a metodologia ágil Scrum nas atividades de desenvolvimento, enquanto o ITIL regia as demais etapas do processo, desde a gestão da mudança, a efetivação do incremento de software e a monitoração dos servidores em produção.

Figura 16 – Etapas do processo de desenvolvimento de software no UOL



Fonte: elaborado pelo autor

O desenvolvimento dos incrementos ao software eram feitos em ciclos que o Scrum denominava Sprint, os quais tinham a duração de 1 mês (4 semanas). As 3 primeiras semanas eram compostas basicamente por atividades de levantamento de requisitos, codificação e testes preliminares. Na 4ª e última semana executava-se a grande parte dos testes, fechava-se o pacote e submetia-se o incremento para aprovação do Comitê de Mudança, uma agremiação típica do ITIL, que se reunia semanalmente e tinha por finalidade avaliar os detalhes das implantações para assim mitigar os riscos de insucesso. Se por qualquer motivo o pacote não pudesse ser finalizado a tempo, o mesmo teria de esperar uma semana, até a próxima reunião do comitê.

Porfim, após a avaliação (executada por meio de um documento detalhando todas as minúcias da mudança), era agendada a implantação, que era efetuada manualmente por membros dos times de P&D e Operação, geralmente ocorrendo às quintas-feiras com horário de início 04:00 e término 08:00 horas da manhã.

4.2 Transformação *Lean*

Em sua obra, Yasuhiro Monden (1983) instituiu um modelo para a implantação *Lean* em uma empresa seguindo 4 passos:

1. Envolver o topo da gestão para orientar e fornecer os recursos necessários
2. Formar equipes de projeto com gestores de sessão, departamento e operação
3. Operar um time-piloto
4. Estabelecer círculos de controle de qualidade e envolver os operadores

A transição *Lean* deve compreender toda a cadeia hierárquica, da base à gerência. Shigeo Shingo (1996), considerado um dos criadores do TPS, dizia ser necessário preparar o ambiente, sensibilizando e conscientizando todos os envolvidos no intuito de promover um engajamento coletivo. Colaboradores devem estar comprometidos com a mudança pois parte principalmente deles os insumos e ações visando a melhoria. Da mesma forma, o processo de mudança se torna insustentável se a gerência não financiar e apoiar sua realização (SHINGO, 1996).

Mesmo não tendo ciência do modelo de Monden, a implantação no departamento de P&D E-commerce do UOL observou precisamente cada um dos passos. A começar pela gerência que promoveu a transformação em todo o departamento por meio de treinamentos, instituição de comitês e estabelecimento de objetivos e metas de curto e médio prazo.

Os colaboradores já estavam dispostos em times (herança dos Métodos Ágeis), organizados em torno de produtos e respeitando uma hierarquia de comando enxuta (líderes de time, coordenadores e gerente).

A ação, coordenada a partir de 2013, visou abranger todo o departamento composto por 15 times. Entretanto, 2 times-piloto já demonstravam melhorias de performance e qualidade em iniciativas isoladas implantadas desde 2011. Os resultados obtidos pelo piloto validaram o conceito *Lean* e auxiliaram a estabelecer o *modus operandi* para as demais implantações.

Tendo isso em vista, foi instituído um grupo de aprimoramento dos métodos de trabalho chamado Comitê de Processos. Esse grupo possuía pelo menos um representante de cada time de projeto, sendo a liderença composta por pessoas experientes em transformações *Lean* e Métodos Ágeis de desenvolvimento de software. O comitê se reunia mensalmente com o objetivo de mentorar os membros dos times na implantação da filosofia enxuta e auxiliar a resolução dos problemas oriundos dessa transição.

4.2.1 Frente de análise: Gestão da Mudança

Uma das primeiras práticas encorajadas foi a instauração de reuniões *Kaizen*. Essa prática, além de trazer à luz ineficiências a serem trabalhadas, estimulava indiretamente o princípio de que cada membro do time é responsável pela melhoria contínua de seu processo de trabalho. Os *Kaizens* eram essenciais na identificação dos ofensores de qualidade e produtividade, e também o momento em que os times refletiam e sugeriam mudanças no método de trabalho (característica típica do ciclo PDCA).

As ações suscitadas pelos *Kaizens* eram avaliadas por meio da evolução das métricas do time. Um exemplo disso foi visto na reestruturação da gestão da mudança dos sistemas. Por meio da análise das métricas, do estudo do fluxo de trabalho e dos *Kaizens*, foi descoberto que o período de implantação do software onerava substancialmente o *lead time* das tarefas e que eram necessárias ações de otimização para dimuição de tempo entre as entregas em produção.

Para isso, foi efetuada uma parceria entre os times de P&D e Operação com o propósito de automatizar o processo de gestão e efetivação da mudança, a diminuição do lote e o aumento na freqüência de entregas, de forma a atingir o que na engenharia de software é conhecido como *delivery* contínuo, e, no *Lean,* fluxo contínuo. Esse período de ajustes internos durou por volta de 12 meses (ano de 2013) e promoveu uma série de aprimoramentos.

Após sequência de análises e negocioções, foram concedidos aos Times de P&D a efetivação das entregas sem a necessidade de uma inspeção do comitê de Operação. Essas mudanças de natureza controlada foram classificadas como pré-aprovadas, não sendo mais necessário aguardar a revisão e aval do Comitê de Mudança para sua execução.

A segunda mudança veio da constituição de um extensivo conjunto de testes automatizados que validassem as versões de software a serem implantadas de forma a garantir a qualidade do entregável. Por se tratar de uma ação automatizada, colheram-se os benefícios de uma cobertura abrangente, a eliminação de falhas humanas e, o mais importante, um tempo de execução ínfimo se comparado à execução manual.

A terceira e mais significativa mudança foi a automação do *deploy*. De todo o processo, essa foi a atividade de maior complexidade e envolveu massivamente o Time de Operações por se tratar de configurações e automações de "baixo nível" em servidores de aplicação e interfaces de comunicação.

4.2.2 Frente de análise: Desempenho do Time

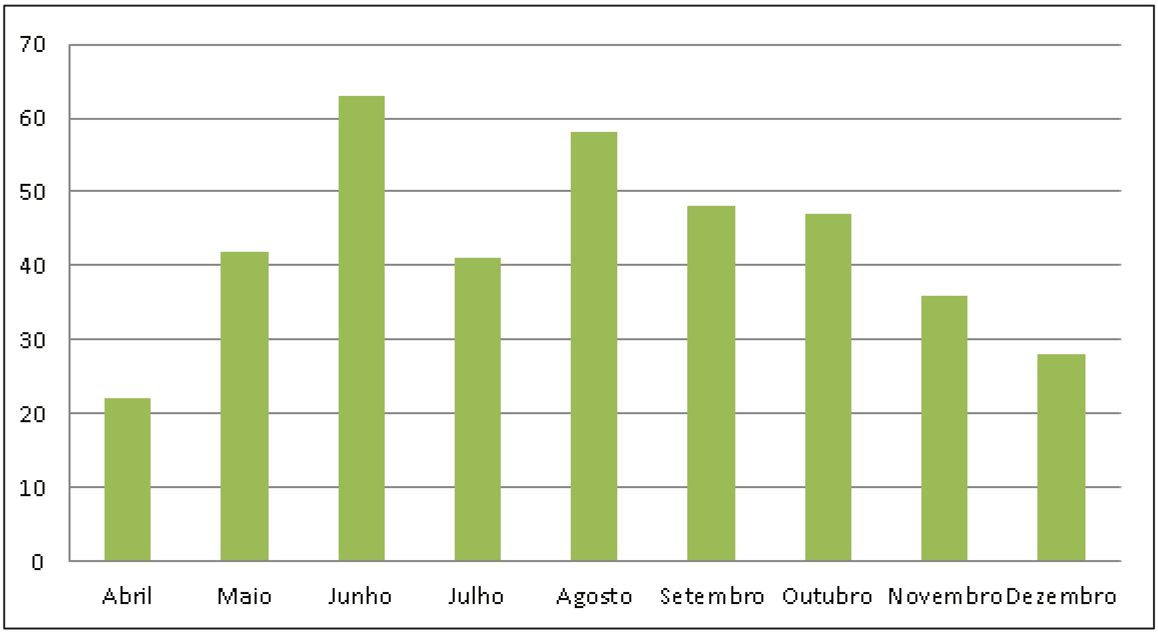
De março a dezembro de 2014 foram coletados dados de desempenho diário do principal time de P&D, responsável pelo desenvolvimento das interfaces e serviços do UOL Marketplace. O time era composto por 7 integrantes sendo 2 webmasters, 2 testadores e 3 desenvolvedores. Desde o início dessa operação foi utilizado o Método Kanban para o gerenciamento das atividades e do fluxo de trabalho.

Uma pequena parte da coleta das métricas foi feita diariamente por meio de registro manual. Porém, a grande maioria foi colhida por meio da ferramenta de gerenciamento de projetos Jira, a qual também abarcava o Quadro Kanban virtual utilizado pelo time. Dessa forma, todas as ações nas tarefas, mudança de estado, executores e tempo gasto ficavam automaticamente registrados.

Ao fim de cada mês era realizado o *Kaizen* do time, no qual realizava-se uma retrospectiva dos principais acontecimentos e expunham-se os resultados na forma de métricas. Desta forma, todas as ações de melhoria propostas a seguir passavam a estar embasadas em fatos e respaldados por números, não somente apoiadas em sentimentos e "achismos". Por meio das métricas podia-se também avaliar com mais consistência se as ações tomadas no passado estavam ou não surtindo o efeito desejado.

A métrica mais simples de ser compreendida é o *throughput*, que nada mais é que a quantidade de tarefas executadas ao fim de um período. Na Figura 17 pode ser visto o *throughput* mensal do time estudado no ano de 2014. Nesse gráfico percebe-se o ganho de velocidade do time ao início do projeto, a queda de produtividade em julho devido aos feriados em jogos da Copa do Mundo, e uma desaceleração no momento de pré-lançamento e lançamento em produção (mêses de novembro e dezembro).

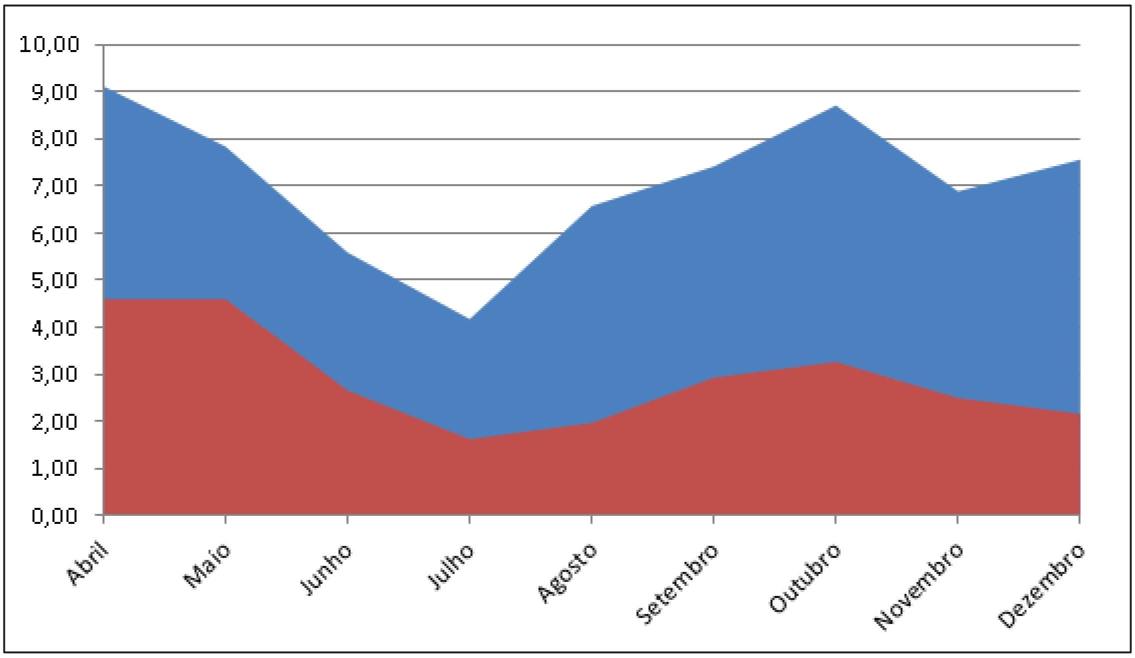
Figura 17 –Throughput – quantidade de tarefas finalizadas por mês em 2014



Fonte: elaborado pelo autor

Outras métricas muito utilizadas são o *Lead Time* (área azul do gráfico) que identifica o tempo médio total gasto da execução das tarefas e o *Work Time* (área vermelha do gráfico) que denota o tempo de trabalho gasto, eliminando o tempo em filas ou em etapas não geradoras de valor. Na Figura 18 é apresentada a quantidade de dias para *Lead Time* e *Work Time* em periodicidade mensal no ano de 2014.

Figura 18 – Lead Time e Work Time – média de dias gastos para finalizar uma tarefa em 2014



Fonte: elaborado pelo autor

Seria coerente dizer que um time deve sempre observar um *Lead Time* descrescente pois isso significa uma execução mais ágil e um time mais produtivo. Porém, a realidade nem sempre é assim. É importante entender que a quantidade média de dias cada vez menor não pode ser uma meta isolada quando é observado o *Lead Time*. Isso é devido à oscilação natural desse montante de tempo, que pode ocorrer por diversos fatores, como por exemplo:

- natureza das tarefas: no período inicial do projeto (mês de abril na Figura 18), as tarefas executadas eram em sua maioria estruturais ou de configuração (por exemplo, criação de serviços, banco de dados e configuração de servidores). Isso acarretou numa diferença de *Lead Time* com o mês de julho, no qual predominaram tarefas de criação de interface gráfica, que são de granularidade mais fina, e culminaram num menor tempo de execução. Assim, entende-se a variação no *Lead Time* como natural, dada a desigualdade entre os tipos de atividades.
- mudanças no time ou no ambiente de trabalho: o *Lead Time* não pode ser estudado dissociado do time e ambiente em que está sendo executado. Como exemplo, é fácil compreender que, se houver uma mudança nos integrantes de um time no meio do projeto, isso provavelmente causará um aumento no *Lead Time.* Isso também acontece quando é estudado o ambiente, ou as condições de

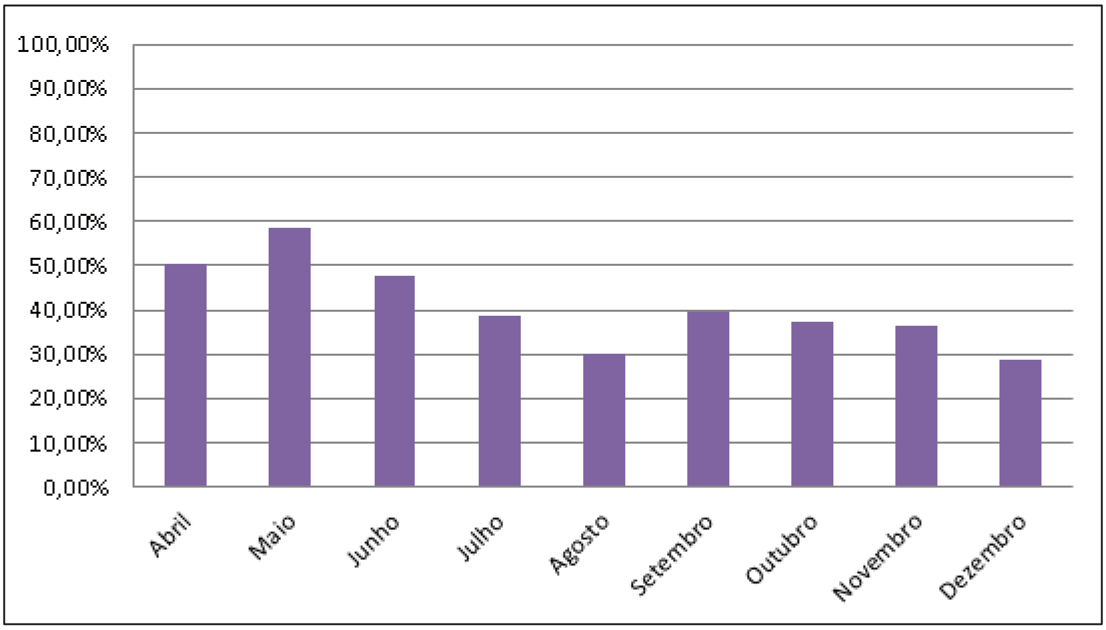
trabalho. No exemplo visto na Figura 18, a partir do mês de agosto foi implementada uma nova etapa no processo, adicionando restrições ao ambiente de testes e qualidade. Essa alteração promoveu um aumento compreensível no *Lead Time.*

- variabilidade externa: a interferência de fatores externos ao projeto é um grande vilão quando se deseja estabilizar o processo visando produtividade e previsibilidade. Contabiliza-se em fatores externos tudo o que não é responsabilidade do time em questão, como tarefas executadas parcialmente por outros times ou a necessidade de autorizações e aprovações de outrem. Como essas demandas são dependentes de atores fora do fluxo de trabalho (não gerenciados pelo sistema puxado), é inserida no processo uma variável imponderável causadora da oscilação do *Lead Time*.

Devido a variância do *Lead Time,* uma outra métrica deve ser examinada: a PE (Process Efficiency), ou Eficiência do Processo. Essa métrica é uma derivação das anteriores, compreendendo a razão entre *Work Time* e *Lead Time* e pode ser vista na Equação 3. Isso significa que quanto maior a PE, mais eficiente é o time em otimizar o tempo, diminuir filas, *buffers* e promover o *one-piece flow*. Na Figura 19 pode-se observar a Efiência do Processo para o time em estudo no ano de 2014.

$$PE = \frac{Work\ Time}{Lead\ Time} \qquad (3)$$

Figura 19 – Process Efficiency – eficiência média mensal do processo em 2014



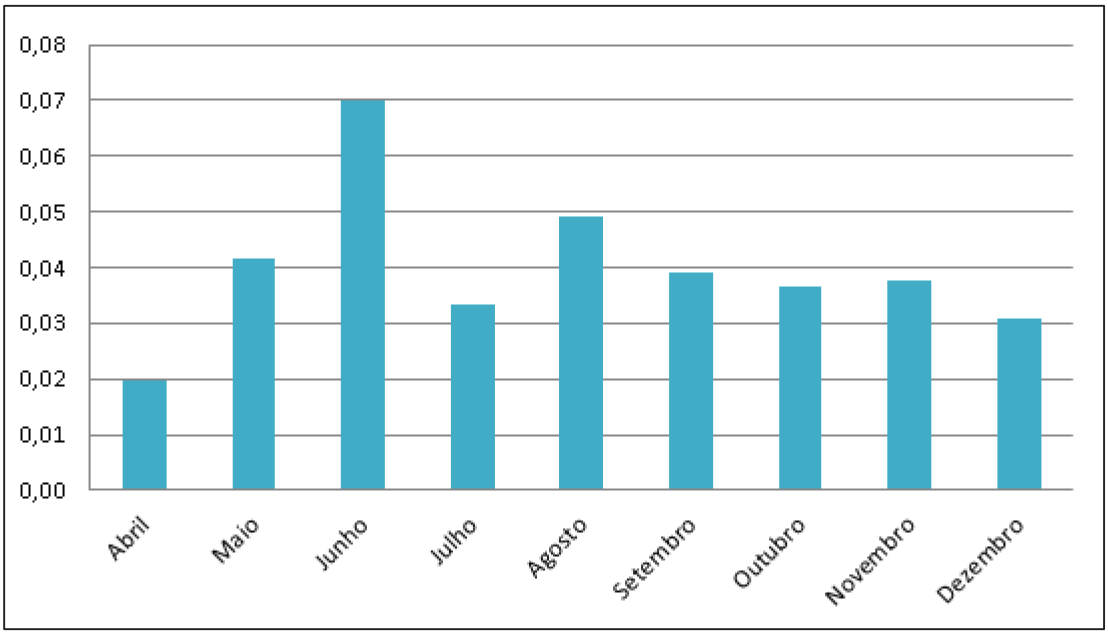
Fonte: elaborado pelo autor

Outra métrica derivada que segue a mesma semântica da PE (que mede a qualidade do *Lead Time*), é a TE (Throughput Efficiency), ou Eficiência do Throughput. A TE mede a qualidade do *throughput* pois leva em conta a quantidade geral de horas úteis no mês e pode ser vista na Equação 4. Isso faz com que a avaliação do período seja mais consistente por avaliar as variações: número de integrantes do time, feriados, férias, faltas e horas extra.

Quando o *throughput* passa a ser considerado tendo-se em vista as horas úteis no mês, percebe-se uma atenuação na diferença de tarefas entregues entre os períodos. A evolução da TE pode ser vista na Figura 20.

$$TE = \frac{Throughput}{Total\ Horas\ Úteis} \qquad (4)$$

Figura 20 – Throughput Efficiency – eficiência média mensal do *throughput* em 2014



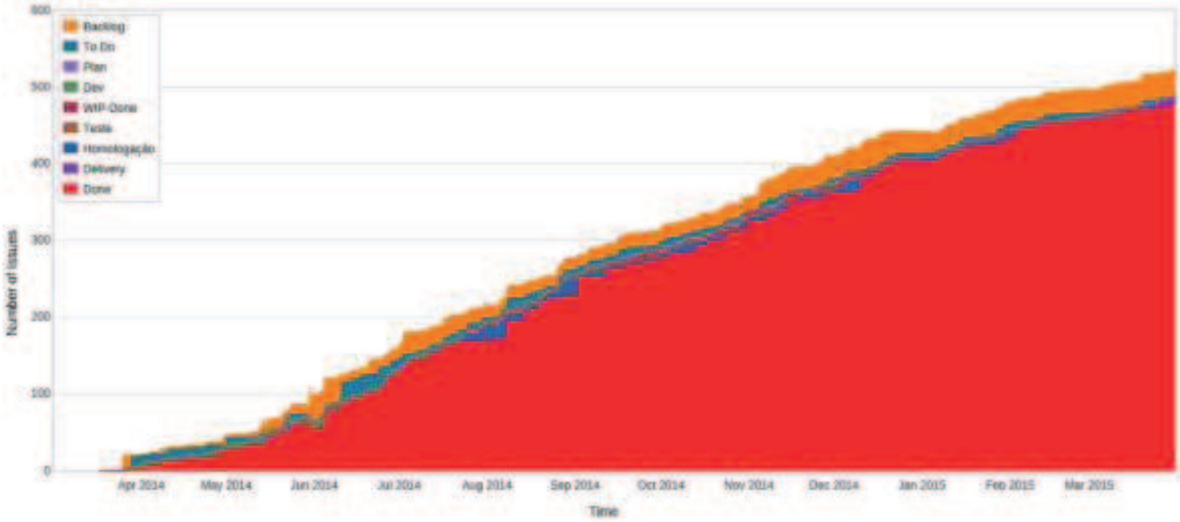
Fonte: elaborado pelo autor

Outra ferramenta utilizada de forma recorrente em reuniões de avaliação de desempenho é o CFD (Cummulative Flow Diagram ou Diagrama de Fluxo Cumulativo). Nesse gráfico é possível visualizar a evolução do estado das tarefas desenvolvidas pelo time tendo em vista:

- O volume vertical das áreas coloridas representa o número de atividades sendo desenvolvidas ao mesmo tempo;
- O volume horizontal das áreas coloridas representa a quantidade de itens num mesmo estado, ou seja, em execução ou filas;
- As espessuras das áreas coloridas do gráfico determinam quão aderente o processo está ao princípio *one-piece flow*. Quanto mais finas as áreas, menor o estoque intermediário nas etapas do fluxo;
- A inclinação das áreas determina a velocidade das entregas. Quanto mais verticais as retas, maior o *throughput*.

Por fim, o CFD não serve apenas para representar dados passados. Com ele também é possível traçar linhas de tendência, auxiliando gestores nas estimativas de prazo e escopo para entregas futuras. O CFD do time em estudo pode ser visto na Figura 21.

Figura 21 – Cummulative Flow Diagram (CFD) do período total do projeto

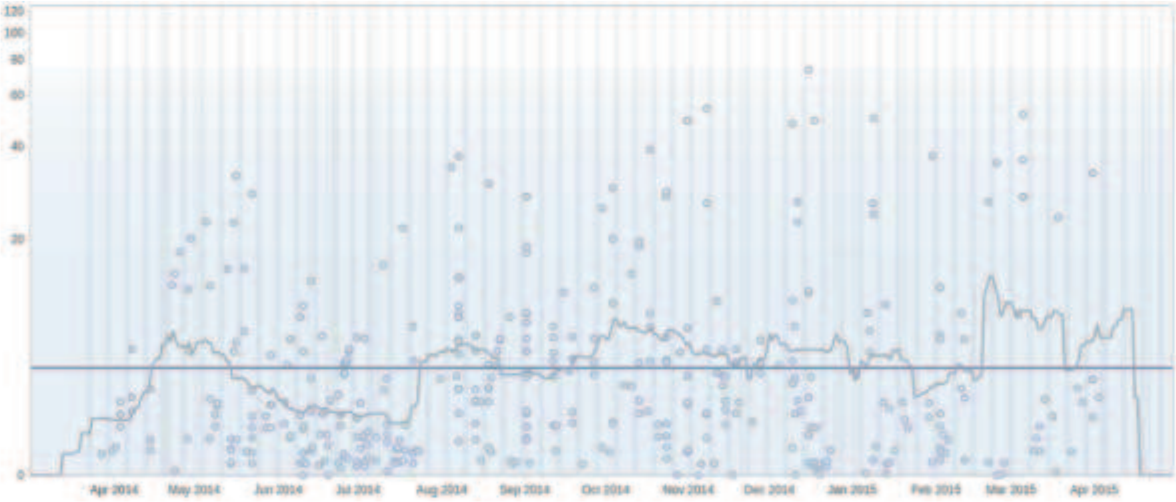


Fonte: elaborado pelo autor

O Control Chart ou Gráfico de Controle é um gráfico complementar muito usado na Engenharia da Produção. Cada ponto nesse gráfico representa as tarefas executadas, sendo o eixo X o seu dia de término e o eixo Y, a quantidade de dias despendida (*throughput*). A reta azul-marinho que cruza o gráfico é a média simples histórica do *throughput* e a linha cinza, a média ponderada do *throughput* nos últimos 5 dias.

Com esses dados é possível avaliar a consistência do processo quanto à variância das entregas, dado que, quanto maior a dispersão dos pontos no quadro, mais instável se apresentava o fluxo de trabalho. Essa reflexão tinha o objetivo de auxiliar a tomada de decisão para um fluxo com menor variabilidade, maior consistência e previsibilidade. O Gráfico de Controle do projeto em estudo pode ser visualizado na Figura 22.

Figura 22 – Gráfico de Controle do período total do projeto



Fonte: elaborado pelo autor

4.2.3 Princípios e Práticas Lean aplicados

O ano de 2013 representou o ano das mais significativas mudanças para os times envolvidos. O plano estabelecido para efetivação das mudanças estava pautado no empirismo, no compartilhamento de experiências entre os times. A sugestão de novas práticas geralmente era balizada por princípios *Lean*, Ágeis ou simplesmente hipóteses que passavam pela validação por intermédio da experimentação. Das inúmeras práticas aplicadas durante os anos do estudo, pode-se perceber uma influência marcante dos princípios do *Lean Thinking* e práticas aplicadas provenientes do *Lean Manufacturing*.

- *Kaizen*

A execução das reuinões de *kaizen* no UOL observou o mesmo princípio visto na Toyota. Seu objetivo principal era a aplicação de ciclos de melhorias contínuas e incrementais (PDCA), visando eliminação do desperdício, aperfeiçoamento das atividades geradoras de valor, promoção do auto-conhecimento por meio da análise de dados e também aspectos como o amadurecimento no âmbito de relações interpessoais.

Elas aconteciam com periodicidade mensal e tinham duração de 60 a 90 minutos. Na parte introdutória, era feita a exposição das métricas do time aferidas no mês em análise conforme Figura 23 e Figura 24. Na segunda parte, os colaboradores tinham a liberdade de expor fatos positivos e negativos ocorridos no período, concluindo a sessão com a proposição de ações de melhoria a serem implantadas no próximo mês.

Figura 23 – Exemplo de métrica em *Kaizen*: tempo gasto em cada etapa do workflow por tarefa

Fonte: elaborado pelo autor

Figura 24 – Exemplo de métrica em *Kaizen*: tempo total gasto em cada etapa do workflow

Fonte: elaborado pelo autor

O objetivo dessas reuniões estava também em promover uma análise pessoal do colaborador sobre o que estava fazendo para entregar valor aos clientes e sua real contribuição com o resultado do time, ao invés de culpar a empresa ou seus colegas pelo não cumprimento de metas. Além do mais, uma vez que a reunião era direcionada pelos próprios colaboradores, ela conferia o poder da proposta aos membros do time, da decisão pelo consenso, promovia o trabalho em equipe, a autonomia e o engajamento. Todas essas características extraídas do princípio *Lean* de respeito às pessoas.

- *Jidoka*

Essencialmente, *jidoka* refere-se a automação e garantia da qualidade. Na engenharia de software, esse princípio está presente, por exemplo, numa prática muito difundida entre os praticantes dos Métodos Ágeis, os testes automatizados (BECK, 2003). Os testes automatizados aplicam perfeitamente o princípio *jidoka* pois são mecanismos automáticos que garantem a qualidade do produto e eliminam a necessidade de desenvolvedores e testadores executarem as mesmas operações retidas vezes.

No UOL, um outro exemplo implementado foi a preparação das estações de trabalho via *scripts* automáticos. Sempre que um desenvolvedor novo chegava ou um hardware tinha de ser substituído, uma série de instalações e configurações deveriam ser feitas para habilitar a estação de trabalho. Isso despendia tempo (por volta de 12 horas) e não garantia a padronização das configurações. Ao implementar um *script* que realizava todo o trabalho de *download*, instalação e configuração, pôde-se garantir um padrão nas estações (o que facilitou muito os *troubleshootings* no processso de desenvolvimento) e diminuiu o tempo total de instalação para 1 hora.

Outra implementação do princípio *jidoka* foi observada na automação dos processos de *build* e *deploy* dos componentes de software. Isso, além de agilizar um processo manual custoso, auxiliou a prevenção de entregas malsucedidas por erros na criação de pacotes.

- *Poka-yoke*

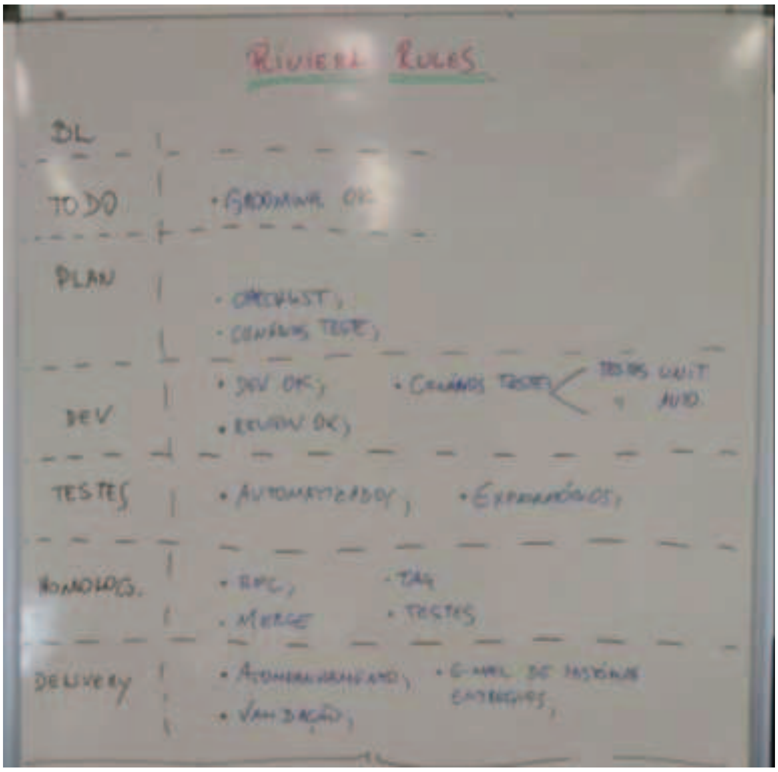
Poka-yoke é um termo japonês que significa “a prova de erros” e, no *lean manufacturing*, pode ser visto em qualquer mecanismo que auxilia um operador previni-los. No projeto em estudo, um exemplo de *poka-yoke* ocorria toda vez que um *bug* de software era encontrado. Como parte da solução do defeito, era obrigatório ao desenvolvedor implementar um teste unitário para cobri-lo. Dessa maneria, um mesmo defeito tinha a tendência a ocorrer uma única vez em todo o ciclo de vida de desenvolvimento.

- *Gestão Visual*

A Gestão Visual é uma prática muito comum em times que seguem os Métodos Ágeis. Como parte dessa herança, era muito comum no UOL estarem afixados na parede, quadros de tarefas, gráficos, metas e avisos. Isso auxiliava a disseminação da informação e o alinhamento entre os membros do time, além de ser um elemento para promover motivação e foco.

Como pode ser visto na Figura 25, as políticas a serem observadas em cada etapa do *workflow* estavam afixadas na parede, à vista de todo o time. Isso também servia como lembrete ao cumprimento do processo para garantia da qualidade.

Figura 25 – Gestão Visual: políticas do workflow expostas em quadro branco, no ambiente de trabalho



Fonte: o autor

- *Andon*

Andon refere-se a um sistema de notificação sobre problemas de processo e qualidade da produção. Um exemplo deste princípio aplicado na engenharia de software é a prática da integração contínua (IC). A IC resume-se em garantir a consistência do software em meio a múltiplos desenvolvedores, os quais criam e alteram conteúdos muitas vezes interdependentes.

Na Figura 26 pode ser visto um Painel de Integração Contínua, ferramenta semelhante ao Painel Andon da manufatura e que foi utilizado no UOL, via monitor afixado na parede. Desta forma, ficava evidente a qualquer membro do time a ocorrência de uma alteração mal-sucedida no código fonte, promovendo ações rápidas para recuperação da consistência do software.

Figura 26 – Painel de Integração Contínua: exemplo de Painel Andon

Fonte: (ATLASSIAN, 2011)

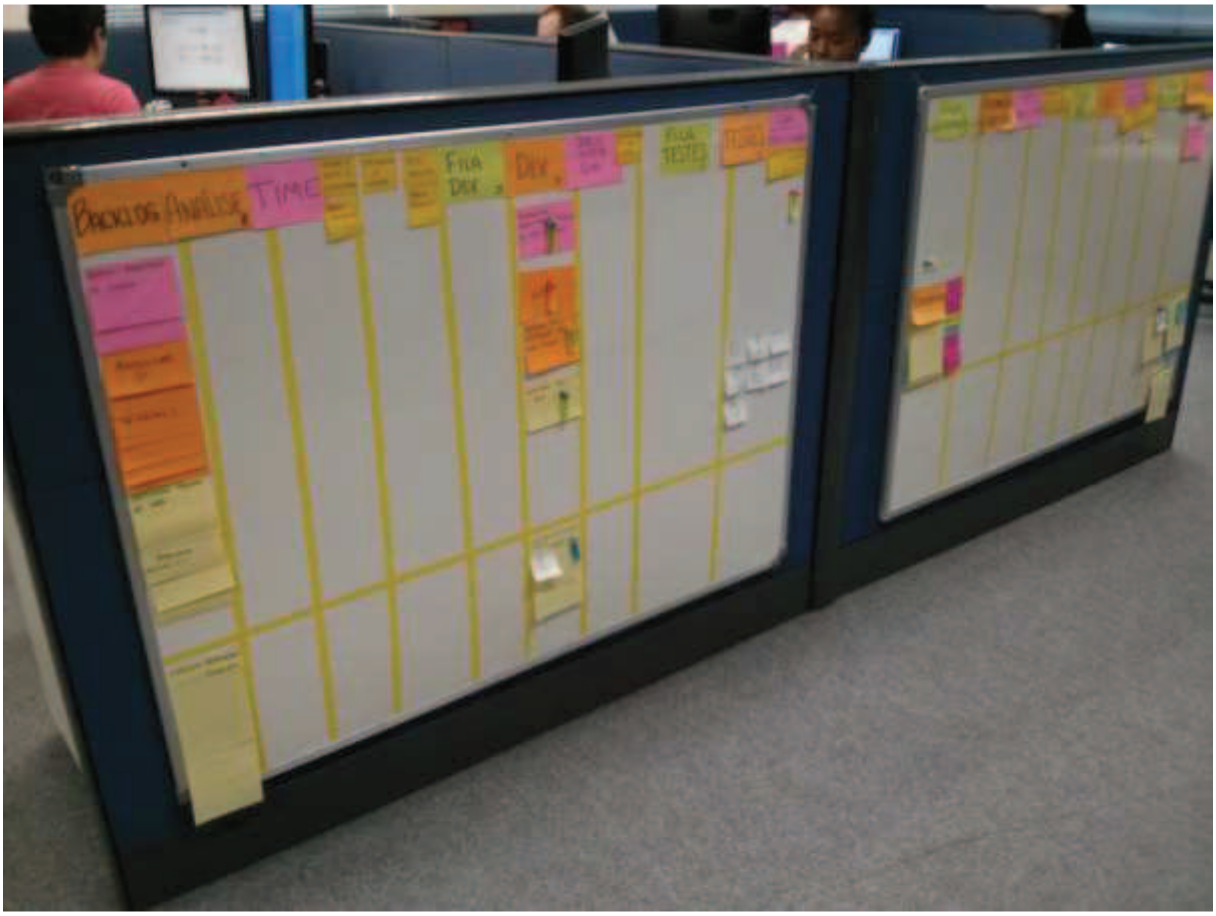
- *Kanban*

A ferramenta *kanban* na manufatura é um sistema de gerenciamento da produção que se utiliza de *kanbans* (cartões) para sinalizar a capacidade produtiva do sistema e assim favorecer o controle do estoque por meio do limite do trabalho em progresso (WIP).

Na engenharia de software isso se faz da mesma forma, porém os *kanbans* não são afixados em produtos ou peças, mas em quadros que podem ser físicos ou virtuais. Essa técnica foi bastante disseminada pelo Scrum, o método ágil mais conhecido e praticado no mercado (VERSION ONE, 2014).

No UOL, o time estudado se utilizava do Método Kanban de desenvolvimento de sistemas. Como percebido pelo próprio nome, o método implementava um sistema puxado para o gerenciamento da produção de software, porém o mesmo ia além do mero uso de *kanbans* num quadro, tendo uma abrangência estendida pela aplicação dos demais princípios da Filosofia *Lean.* O quadro Kanban físico utilizado para gerenciar as demandas do time pode ser visto na Figura 27. Posteriormente o mesmo foi subistituído por um quadro virtual.

Figura 27 – Quadro Kanban afixado no ambiente de trabalho



Fonte: o autor

- *Nemawashi*

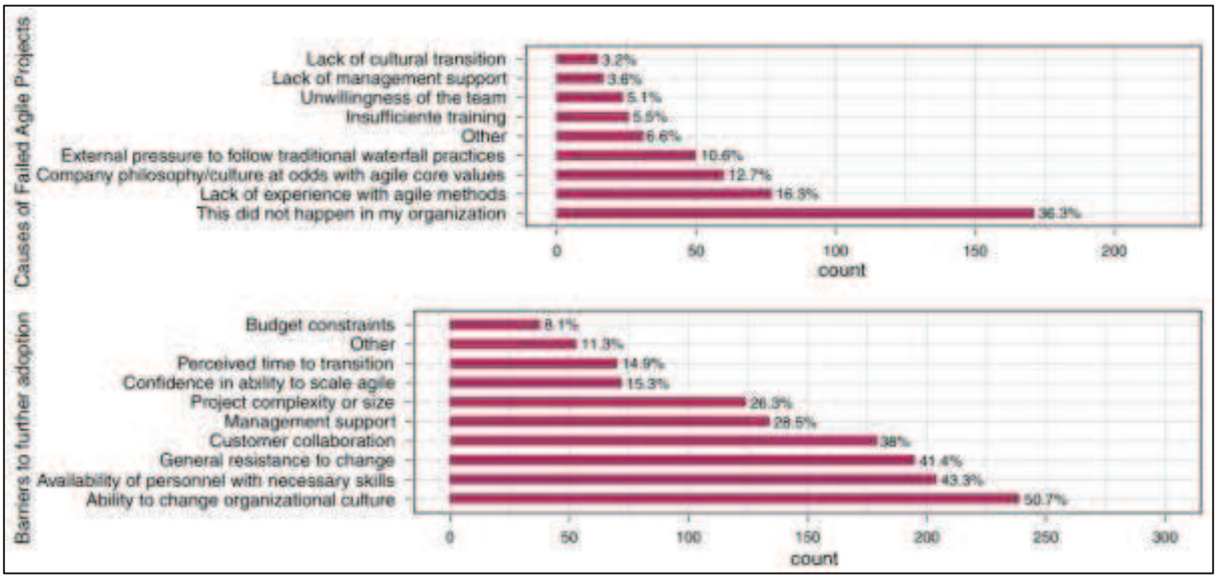
Por mais que se fale em processos, técnicas e ferramentas, ao fim do dia, todo o resultado e tudo o que é construído está fundamentado em pessoas. Há momentos em que as empresas passam por reestruturações e um fator muito importante não pode ser negligenciado: a receptividade ou a resistência das pessoas às mudanças.

Um exemplo disso pode ser visto na engenharia de software, quando analisadas pesquisas acadêmicas e de mercado. A consultoria especializada em Métodos Ágeis Version One publica uma pesquisa de alcance global desde 2005 e que foi reproduzida em 2013 por pesquisadores da USP, porém com um universo de respondentes restrito a instituições brasileiras.

Ambas as pesquisas reuniam novos projetos, ou seja, em transição de métodos tradicionais para os Métodos Ágeis e obtiveram resultados semelhantes quanto a nomeação de causas de falha e barreiras para futuras implementações. Algumas das causas mais indicadas foram: filosofia/cultura da empresa conflitante, pressão externa para retornar aos métodos anteriores, falta de interesse do time na mudança, falta de transição da cultura organizacional. Quanto a barreiras para próximas implementações, as respostas mais indicadas foram: falta de habilidade em mudar a cultura organizacional, disponibilidade de pessoal e resistência à mudança.

Como pode ser visto na Figura 28, as principais causas de falhas e barreiras para projetos em transição não estão no objeto da mudança, mas nos fatores humanos que circundam a transição.

Figura 28 – Fatores humanos: causas de falha e barreiras à futuras implementações



Fonte: (MELO, *et al.*, 2013, p. 538)

Dessa forma, entende-se que, a despeito dos benefícios que a mudança poderia proporcionar, o fator crítico para as empresas pesquisadas estava no correto gerenciamento da transição.

No contexto estudado, o *Nemawashi* representou um procedimento formal iniciado pela alta gerência, e que estabeleceu as bases para todo o processo de transição. Isso compreendeu a organização de comitês de mudança com a participação massiva dos envolvidos, o que foi imprescindível para reunir apoio, colher *feedbacks* e ministrar treinamentos, sendo um elemento fundamental por possibilitar a solução de questionamentos, o consenso dos envolvidos e a diminuição dos efeitos da resistência à mudança.

- *Gemba / Genchi genbutsu*

O *genchi genbutsu* é a tomada de decisão por parte do gestor, embasado no conhecimento adquirido *in loco* e não por meio de informação transmitida por outrem. No caso em estudo, as lideranças diretas sobre o time estavam em dois papéis: o Agile Coach e o Product Owner. Ambos os termos foram emprestados de outros métodos de desenvolvimento de software.

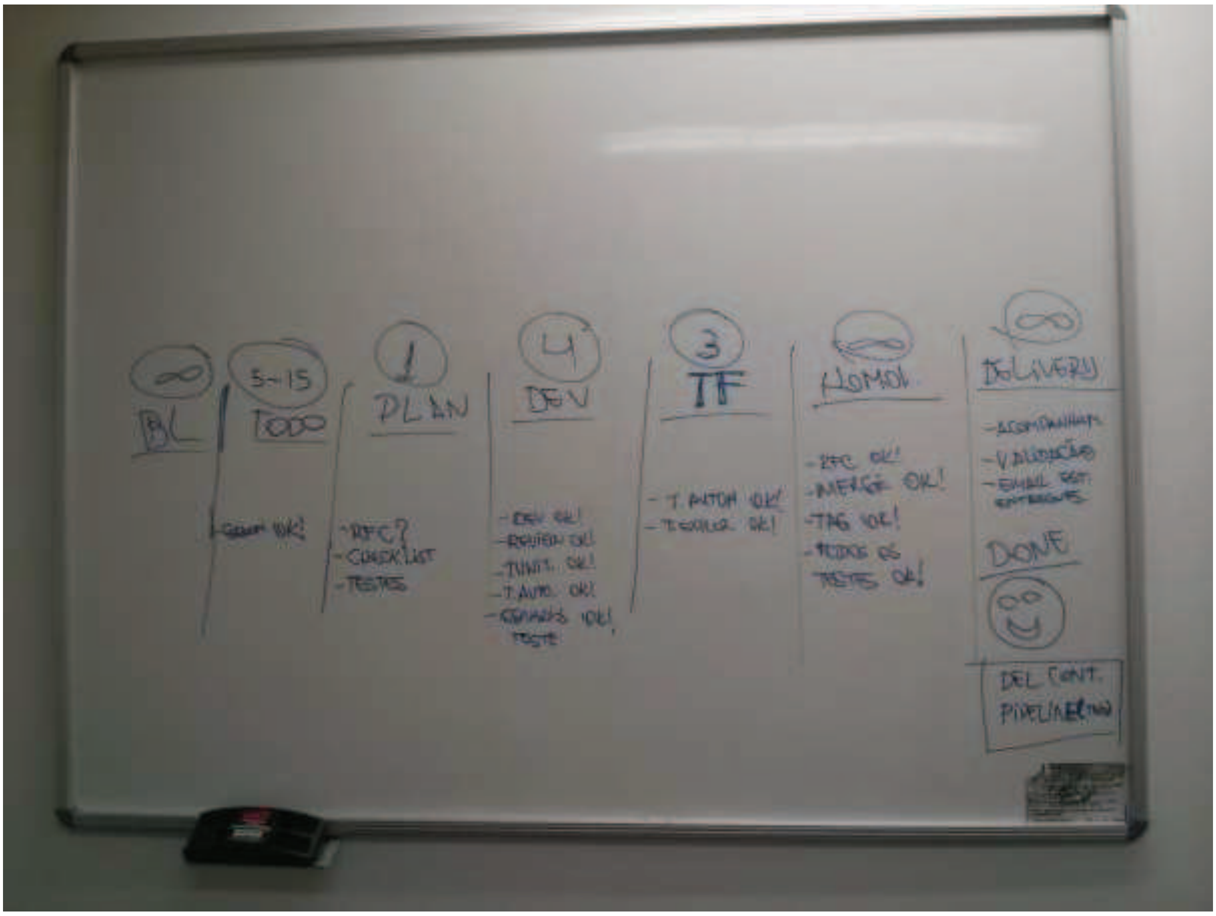
O Agile Coach era responsável por coordenar o time com o foco no processo, na produtividade e na qualidade da entrega do software. No time em estudo, ele também era um desenvolvedor e executava os dois papéis de forma concomitante. Isso auxiliava muito esse líder a entender as dificuldades encontradas no dia-a-dia. Já o Product Owner era responsável por gerenciar todos os *stakeholders,* organizando e priorizando suas demandas para porteriormente serem produzidas pelo time.

Nesse caso, o *genchi genbutsu* acontecia na medida que ambos os líderes se assentavam num raio inferior a 6 metros de distância dos demais membros do time, facilitando a comunicação, a resolução de dúvidas e a co-criação.

- *Value Stream Map* (VSM)

O *Value Stream Map* (VSM), ou mapa de fluxo de valor, foi implementado assim que se iniciou a reestruturação dos times no projeto UOL Marketplace. Num processo que durou por volta de 4 horas, organizado em duas reuniões, foi produzido um modelo que representava todas as etapas do processo e seus envolvidos. Isso serviu para analisar o estado atual do fluxo, pontos de otimização, desperdícios, o tamanho das filas e também projetar o estado futuro ideal para a cadeia produtiva, que pode ser vista na Figura 29. Dessa forma, o fluxo de trabalho se tornou conhecido à todos, foi gerado um alinhamento sobre o objetivo da transição e também foram estabelecidas as metas e *milestones* de produtividade e qualidade para o projeto.

Figura 29 – Value Stream Map do fluxo de trabalho no UOL Marketplace



Fonte: o autor

- *Heijunka*

O objetivo da *heijunka* na manufatura é converter a instabilidade da demanda em um processo previsível, visando estabilizar o fluxo de valor e atenuar a variabilidade. Na engenharia de software, essa observância deve ser muito maior, dado se tratar da produção de um bem intangível, fruto de trabalho do conhecimento, que é mais sensível a fatores como criatividade e incertezas; o que eleva consideravelmente os níveis de variabilidade.

David Anderson (2010) aprofundou os estudos sobre variabilidade do processo dentro do Método Kanban, criando mecanismos que pudessem explorar o conceito da *heijunka*. Para as fontes internas de variabilidade, ou seja, aquelas que são geradas e condicionadas pelo time e processo, podem-se enumerar algumas ações:

- Diminuir o tamanho do lote de trabalho. Por exemplo, cada tarefa deve conter um único objetivo.

- Estabelecer classes de serviço de forma a classificar o tipo de tarefa que está sendo executada. Por exemplo, criação de banco de dados, interface gráfica, serviços, configurações de servidores.
- Diminuir na medida do possível o WIP. Trabalhar concomitantemente em todas as etapas do processo (*one-piece flow*).
- Possuir um time multi-funcional e cross-funcional para suprir picos de demanda em qualquer etapa do processo e, assim, suavizar e garantir o fluxo.
- Gerenciar e evitar o retrabalho gerado por insumos inconsistentes (requisitos incompletos ou ambíguos), defeitos (*bugs*) ou refatorações não planejadas.

Já as fontes externas de variabilidade são as que acontecem por ações ou interferências externas ao time, como bloqueios por falta de informações, espera por trabalho executado por outro time e quando demandas inesperadas são priorizadas. Essas devem ser atenuadas ao máximo.

- Troca Rápida (SMED e QCO)

As siglas SMED (*Single Minute Exchange of Dies*) e QCO (*Quick Change Over*) são utilizadas para representar as ferramentas *Lean* que promovem a troca rápida de *setups*, fator crucial para o fluxo produtivo com pouco variabilidade (*heijunka*).

No UOL, esse conceito foi implementado tendo exatamente o mesmo objetivo. Em teoria, os times que participavam do projeto estavam divididos pelo tipo de componente que estavam preparados à trabalhar. Isso feria o princípio da *heijunka* toda vez que havia um pico de demanda de um determinado componente, gerando uma variabilidade devida ao gargalo.

Para resolver esse problema, além de times capazes de trabalhar com qualquer tipo de demanda, era necessário agilizar o processo de troca de configuração da máquina do desenvolvedor, uma operação custosa e propensa a erros.

A solução encontrada foi a criação de ambientes de desenvolvimento a partir de máquinas virtuais pré-configuradas, o que permitia ao desenvolvedor trabalhar com múltiplos componentes sem alterar suas configurações locais.

4.3 Resultados

Em 2012, antes do período de transformação *Lean*, o número de entregas feitas em produção era considerado padrão para times Scrum de desenvolvimento de software. Havia o planejamento inicial de uma entrega por mês, totalizando 12 no período de um ano. Caso houvesse falha em alguma execução, era agendada uma entrega corretiva para a mesma semana. Isso fez com que em 2012 ocorressem 15 grandes entregas. Esses números não eram ruins, porém o *mindset Lean* havia indicado pontos de desperdício e encorajado nos times a busca pela “perfeição”.

Dada a sequência de eventos *Kaizen*, 2013 foi um ano de transição que promoveu o aperfeiçoamento do processo, principalmente a integração entre as etapas de construção e implantação. Todavia, os dados coletados nesse período estavam incompletos ou inconsistentes e tiveram de ser descartados.

Finalmente, em 2014 foi iniciado o processo integrado entre os times de P&D e Operação. A diminuição do lote (pacote de software) proporcionou um crescimento expressivo na quantidade de entregas e junto a todas as demais melhorias, culminou no aumento da taxa de sucesso dos *deploys* (de 80% para 96,1%), que pode ser visto na Tabela 5.

Tabela 5 – Números de entregas e respectivas taxas de sucesso e falha (2012-2014)

	2012	**2013**	**2014**
Nº Entregas	15	Fase de transição	428
Nº Sucesso	12		411
Nº Falhas	3		17
Taxa Sucesso	80,0%		96,1%
Taxa Falha	20,0%		3,9%

Fonte: elaborado pelo autor

Outro resultado mensurado foi o nível de maturidade do processo de acordo com os membros do time de P&D. Essa auto-avaliação dos membros do time em estudo foi realizada nos períodos pré e pós implementação da transição *Lean* (jan e nov/2014) e respondida pelos 7 integrantes do time. A Tabela 6 apresenta as competências avaliadas, a média simples das respostas e a diferença entre as médias.

Tabela 6 – Média simples da auto-avaliação do time de P&D

Competência	Jan/2014	Nov/2014	Diferença
Delivery contínuo e processo de build	1,00	2,85	+1,85
Boas práticas de desenvolvimento/codificação	0,71	3,57	+2,86
Utilização de tecnologia de ponta	0,85	1,00	+0,15
Aprimoramento dos processos	1,28	3,28	+2,00
Práticas para garantia da qualidade	1,42	3,14	+1,72

Fonte: elaborado pelo autor

Como pode ser visto, na avaliação dos membros do time de P&D, houve o avanço no nível de maturidade de todas as competências. Há um destaque para o item “Boas práticas de desenvolvimento/codificação” que, além de possuir a maior gradação (3,57) na avaliação de novembro, foi a competência com maior crescimento (+2,86) entre as respostas pré e pós transição *Lean*.

O fato de haver um crescimento na prática de *delivery* contínuo, corrobora com o aumento das taxas de sucesso dos *deploys* mensurados na Tabela 5, e com o avanço em eficiência nesse processo visto o expressivo aumento na quantidade praticada.

Outro item a ser observado é a “Utilização de tecnologia de ponta”, que apresentou um pequeno crescimento (+0,15). Isso significa que não houve mudanças estruturais tecnológicas como aquisição de hardware e nem mesmo *upgrades* em licenças de software. Os ganhos de produtividade, qualidade e o aumento nos níveis de avaliação foram fruto de mudanças de gestão, sem aquisições de tecnologia ou recursos financeiros.

O produto desenvolvido no período abordado foi lançado em novembro de 2014. Os integrantes do time de desenvolvimento observaram o período de manutenção e operação assistida até março de 2015, quando foram realocados para outros projetos.

5 CONSIDERAÇÕES FINAIS

Os princípios da Filosofia *Lean* difundidos pelo Sistema de Produção Toyota demonstraram sua eficácia com os resultados da montadora japonesa, que se tornou líder num segmento de grande concorrência, por meio dessa mundança no paradigma de gestão.

No universo do desenvolvimento de software, o *Lean*, em parceria com os Métodos Ágeis, está tendo cada vez mais espaço nas empresas, apresentando uma abordagem pragmática de redução de desperdícios e custos, aumento de lucros e do valor entregue aos clientes. Como foi apresentado, o *Lean* é um grande beneficiador da prática de desenvolvimento de software pois possibilita maior eficiência, eficácia, integração e qualidade.

Porém, como alertaram Poppendieck e Cusumano (2012), se esse estudo for visto sob a ótica da implementação de um conjunto de práticas, não se faz uma boa adaptação entre ambientes da manufatura e do software. Por isso, o *Lean* deve ser pensado como um conjunto de princípios ao invés de práticas; e sua aplicação na engenharia de software um processo conceitual derivado e empírico.

Sendo assim, entende-se que as aplicações apresentadas nesse estudo respondem a questão de pesquisa "Como a Filosofia *Lean* pode ser aplicada em um projeto de desenvolvimento de software?", porém, fica evidente que o fator crucial não está em copiar ou seguir um conjunto de regras estáticas. Toda tentativa de implantação deve ser vista como uma instância única e exclusiva, que segue um padrão embasado em conceitos empíricos desenvolvidos por décadas na indústria de manufatura japonesa e que pode ser aplicado na engenharia de software, de forma a colher os benefícios do aumento da eficiência e eficácia.

À empresa UOL S/A, seria proveitosa a contínua mensuração do impacto das transformações *Lean* em seus times de desenvolvimento de software, o que serviria de justificativa para o aumento nos investimentos, e também estímulo à ampliação das implantações para o nível corporativo.

Como sugestão de trabalhos futuros, fica o estudo de transições *Lean* em diferentes contextos. No âmbito empresarial, pode-se avaliar a implantação em companhias de porte menor como as *startups*, ou de setores mais letárgicos em termos de inovação tecnológica e processo de software, como setores industrial, agrícola e da administração pública. Um outro enfoque seria estabelecer um estudo comparativo com os métodos tradicionais de desenvolvimento, por meio de um tratamento quantitativo dos dados e expressando de forma objetiva os ganhos ou perdas de produtividade e qualidade com a utilização de métodos *Lean* de desenvolvimento de software.

REFERÊNCIAS

ABES. **Mercado Brasileiro de Software: panorama e tendências**. ABES - Associação Brasileria das Empresas de Software. São Paulo, p. 24. 2015.

ANDERSON, D. **Kanban:** Successful Evolutionary Change for Your Technology Business. Sequim: MA: Blue Hole Press, 2010.

ANDERSON, D. J. Lean Software Development. **Modern Management Methods**, 2011. Disponivel em: <http://lkna.leankanban.com/wp-content/uploads/2013/03/LeanSoftwareDevelopment.pdf>. Acesso em: 5 jun. 2014.

APPLEGATE, L. M.; MCFARLAN, F. W.; MCKENNEY, J. L. **Corporate information systems mangement:** The issues facing senior executives. 4. ed. Chicago: Irwin, 1996.

ATLASSIAN. Wallboards of Shame. **Atlassian Blogs**, 2011. Disponivel em: <http://blogs.atlassian.com/2011/09/wallboards_of_shame>. Acesso em: 27 janeiro 2016.

BECK, K. **Test-driven development:** by example. [S.l.]: Addison-Wesley Professional, 2003.

BECK, K. et al. Manifesto Ágil. **Manifesto para Desenvolvimento Ágil de Software**, 2001. Disponivel em: <http://agilemanifesto.org/iso/ptbr/>. Acesso em: 5 jun. 2014.

BOEG, J. **Priming Kanban:** A 10 step guide to optimizing flow in your software delivery system. [S.l.]: Trifork, 2011.

BOEHM, B. W. A spiral model of software development and enhancement. **Computer**, v. 21, n. 5, p. 61-72, maio 1988.

BRITANNICA. Enciclopédia Britânica Online, 2015. Disponivel em: <http://www.britannica.com/topic/Fordism>. Acesso em: 18 novembro 2015.

CAWLEY, O.; WANG, X.; RICHARDSON, I. **Lean/Agile Software Development Methodologies in Regulated Environments - State of the Art**. Fist Internation Conference, LESS 2010. Helsinki: Springer. 2010. p. 31-35.

CHARETTE, R. N. **Foundations of Lean Development:** The Lean Development manager's guide. Spotsylvania: ITABHI Corporation, v. 2, 2002.

COPLIEN, J.; BJORNVIG, G. **Lean architecture:** for Agile Software Development. [S.l.]: John Wiley & Sons Ltd, 2010.

DEMARCO, T.; LISTER, T. **Peopleware:** productive projects and teams. 2. ed. [S.l.]: Dorset House Publishing, 1999.

DEMING, W. E. A System of Profound Knowledge. In: NEEF, D.; SIESFELD, G. A.; CEFOLA, J. **The Economic Impact of Knowledge**. [S.l.]: Butterworth-Heinemann, 1998. p. 161.

DIJKSTRA, E. W. The humble programmer. **Communications of the ACM**, v. 15, n. 10, p. 859-866, 1972.

DORUK OTOMASYON. Doruk Otomasyon, 2010. Disponivel em: <http://www.dorukotomasyon.com/tr/andon.htm>. Acesso em: 21 janeiro 2016.

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

FUJIMOTO, T. **The evolution of a manufacturing system at Toyota**. [S.l.]: Oxford University Press Oxford, v. 104, 1999.

GALGANO, A. **Las tres revoluciones. Caza del desperdicio:** Doblar la productividad con la "LEAN Production". Madrid: Ediciones Díaz de Santos, 2004.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 4. ed. São Paulo: Atlas, 1999.

GOFORTH, K. A. **Adapting lean manufacturing principles to the textile industry**. Dissertação (Mestrado em Ciências). Universiidade da Carolina do Norte. [S.l.]. 2008.

GOLDRATT, E. M. **Theory of constraints**. Croton-on-Hudson, NY: North River Press, 1990.

GOLDRATT, E. M. **Critical chain**. Great Barrington, MA: North River Press, 1997.

HEIJUNKA - flexibilizar e nivelar a produção. **Citisystems**, 2013. Disponivel em: <http://www.citisystems.com.br/heijunka/>. Acesso em: 27 janeiro 2016.

HIGHSMITH, J. A. **Adaptive Software Development:** An Evolutionary Approach to Controlling Chaotic Systems. [S.l.]: Dorset House Publishing, 2000.

HIGHSMITH, J. A. **Agile software development ecosystems**. [S.l.]: Addison-Wesley Professional, v. 13, 2002.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The Unified Software Development Process**. Readding, MA: Addison-Wesley, 1999.

JONSSON, H. **Lean Software Development:** A Systematic Review. IDT Mini-conference on Interesting Results in Computer Science and Engineering (IRCSE). [S.l.]: [s.n.]. 2012.

KATAYAMA, E. T. **A contribuição da indústria da manufatura no desenvolvimento de software**. Dissertação (Mestrado) - Universidade de São Paulo. [S.l.]. 2010.

KEYTE, B.; LOCHER, D. A. **The complete lean enterprise:** Value stream mapping for administrative and office processes. [S.l.]: CRC Press, 2004.

KILPATRICK, J. Lean Principles. **Utah Manufacturing Extension Partnership**, 2003. Disponivel em: <http://www.inmatech.nl/res/pdfs/leanprinciples.pdf>. Acesso em: 7 janeiro 2016.

KRAFCIK, J. F. Triumph of the lean production system. **Sloan Management Review**, v. 30, n. 1, p. 41-51, 1988.

LAKATOS, E. M.; MARCONI, D. A. E. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas, 2003.

LARMAN, C.; VODDE, B. **Lean Primer**, 2009. Disponivel em: <http://www.leanprimer.com/downloads/lean_primer.pdf>. Acesso em: 19 janeiro 2016.

LARMAN, C.; VODDE, B. **Practices for scaling lean & agile development:** large, multisite, and offshore product development with large-scale Scrum. [S.l.]: Pearson Education, 2010.

LEAN TI. O que é Mapeamento do fluxo de valor (MFV)? **Lean TI**, 2013. Disponivel em: <http://www.leanti.com.br/conceitos/6/Mapeamento-do-fluxo-de-valor.aspx>. Acesso em: 7 janeiro 2016.

LEHMAN, M. M. Programs, life cycles and the laws of software evolution. **IEEE**, v. 68, n. 9, p. 1060-1076, 1980.

LIKER, J. K. **The Toyota way**. [S.l.]: Esensi, 2004.

LITTLE, J. A proof for the queuing formula: $L= \lambda W$. **Operations research**, v. 9, n. 3, p. 383-387, 1961.

LO GIUDICE, D. et al. **The 2015 State Of Agile Development: Learn From Agile Expert Firms**. Forrester Research. [S.l.]. 2015.

LO GIUDICE, D.; KISKER, H.; ANGEL, N. **How Can You Scale Your Agile Adoption?** Forrester Research. [S.l.]. 2014.

MAFFEO, B.; SILVA, S. B. D. **Engenharia de software e especificação de sistemas:** soluções para quem necessita da informação para agir. Rio de Janeiro: Campus, 1992.

MAGELA, R. **Engenharia de Software Aplicada:** Princípios. 4. ed. [S.l.]: Alta Books, v. 1, 2006.

MEHTA, M.; ANDERSON, D.; RAFFO, D. Providing value to customers in software development through lean principles. **Software Process: Improvement and Practice**, v. 13, n. 1, p. 101-109, 2008.

MELO, C. O. et al. The evolution of agile software development in Brazil. **Journal of the Brazilian Computer Society**, v. 19, n. 4, p. 523-552, 2013.

MIDDLETON, P. Lean software development: two case studies. **Software Quality Journal**, v. 9, n. 4, p. 241-252, 2001.

MIDDLETON, P.; FLAXEL, A.; COOKSON, A. **Lean software management case study:** Timberline inc. Extreme Programming and Agile Processes in Software Engineering. [S.l.]: Springer Berlin Heidelberg. 2005. p. 1-9.

MIDDLETON, P.; JOYCE, D. Lean software management: BBC Worldwide case study. **IEEE Transactions on Engineering Management**, v. 59, n. 1, p. 20-32, 2012.

MONDEN, Y. **Toyota production system:** practical approach to production management.

[S.l.]: Engineering & Management Press, 1983.

MORGAN, T. **Lean Manufacturing Techniques Applied to Software Development**. Tese (Mestrado em Engenharia e Gestão). Massachusetts Institute of Technology. Cambridge. 1998.

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

OHNO, T. **Toyota production system:** beyond large-scale production. Portland, OR: Productivity Press, 1988.

OSONO, E.; SHIMIZU, N.; TAKEUCHI, H. **Extreme Toyota:** Radical contradictions that drive success at the world's best manufacturer. [S.l.]: Wiley, 2008.

PERNSTÅL, J.; FELDT, R.; GORSCHEK, T. The Lean Gap: A Review of Lean Approaches to Large-Scale Software Systems Development. **The Journal of Systems and Software**, n. 86, p. 2797-2821, jul. 2013.

POPPENDIECK, M.; CUSUMANO, M. Lean Software Development: A Tutorial. **IEEE Software**, 2012. 26-32.

POPPENDIECK, M.; POPPENDIECK, T. **Lean software development:** An agile toolkit. [S.l.]: Addison-Wesley Professional, 2003.

POPPENDIECK, M.; POPPENDIECK, T. **Implementing Lean Software Development:** From Concept to Cash. [S.l.]: Addison-Wesley, 2006.

POPPENDIECK, M.; POPPENDIECK, T. **Leading Lean Software Development**. [S.l.]: Addison-Wesley, 2010.

PRESSMAN, R. **Engenharia de Software**. 6. ed. [S.l.]: McGraw-Hill, 2006.

PRESSMAN, R. **Software Engineering:** A Practitioner’s Approach. 7. ed. [S.l.]: McGraw-Hill, 2009.

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

REINGOLD, E. **Toyota:** People, Ideas, and the Challenge of the New. Londres: Penguin Books, 1999.

RIES, E. **The lean startup:** How todays entrepreneurs use continuous innovation to create radically successful businesses. [S.l.]: Random House LLC, 2011.

RODRÍGUEZ, P. **Combining lean thinking and agile software development**. 143 f. Tese (Doutorado em Ciências) - Departamento de Ciências de Processamento da Informação. Universidade de Oulu. Oulu. 2013.

ROYCE, W. W. **Managing the development of large software systems**. IEEE WESCON. [S.l.]: [s.n.]. 1970.

SELLTIZ, C.; WRIGHTSMAN, L. S.; COOK, S. W. **Métodos de pesquisa nas relações sociais**. 2. ed. São Paulo: EPU, 1987.

SENGE, P. **A Quinta Disciplina:** A Arte e A Prática da Organização Que Aprende. São Paulo: Best Seller, 1990.

SHINGO, S. **O Sistema Toyota de Produção - o ponto de vista da engenharia de produção**. Porto Alegre: Bookman, 1996.

SILVA, E. L. D.; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianópolis: UFSC, 2005.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. [S.l.]: Pearson Addison-Wesley, 2007.

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

STANDISH GROUP. **CHAOS Report**. Standish Group International Inc. [S.l.]. 1995.

STANDISH GROUP. **CHAOS Report**. Standish Group International Inc. [S.l.]. 2014.

STOECKLEIN, M. Quality Improvement Systems, Theories and Tools. In: STOECKLEIN, M. **The Healthcare Quality Handbook**. [S.l.]: Health Administration Press, 2005. Cap. 4.

SUBRAMANYA, S. R. Adapting some Rules and Principles of TPS (Toyota Production System) to Software Development. **International Journal of Computer Applications**, p. 36-41, abril 2012.

SUTTON, J. Welcoming Software Into the Industrial Fold. **Crosstalk: The Journal of Defense Software Engineering**, v. 21, n. 5, maio 2008.

THIOLLENT, M. **Metodologia da pesquisa-ação**. São Paulo: Cortez & Autores Associados, 1988.

THOMSON REUTERS. Web of Science, 2016. Disponivel em: <http://apps.webofknowledge.com>. Acesso em: 7 janeiro 2016.

TOYODA, E. **Toyota:** Fifty Years in Motion. [S.l.]: Kodansha International, 1987.

TRIPP, D. Pesquisa-ação: uma introdução metodológica. **Educação e pesquisa**, v. 31, n. 3, p. 443-466, 2005.

UOL SA. Sobre o UOL. **UOL**, 2015. Disponivel em: <http://sobreuol.noticias.uol.com.br>. Acesso em: 7 janeiro 2016.

VERSION ONE. **State of Agile Survey**. [S.l.]. 2014.

VILKKI, K.; ERDOGMUS, H. Point/Counterpoint. **IEEE Software**, v. 29, n. 5, p. 60-63, 2012.

WEST, D. et al. **Agile development: Mainstream adoption has changed agility**. Forrester Research. [S.l.]. 2010.

WIDMAN, J.; HUA, S. Y.; ROSS, S. C. Applying lean principles in software development process–a case study. **Issues in Information Systems**, v. 1, n. 11, p. 635-639, 2010.

WOMACK, J. P.; JONES, D. T. **Lean Thinking:** banish waste and create wealth in your corporation. [S.l.]: Simon and Schuster, 2010.

WOMACK, J. P.; JONES, D. T.; ROOS, D. **The machine that changed the world:** The story of lean production - Toyota's secret weapon in the global car wars that is now revolutionizing world industry. [S.l.]: Simon and Schuster, 1991.

APÊNDICE A

Estão dispostas abaixo as respostas da Pesquisa de Maturidade dos processos internos do UOL feita com o time de P&D responsável pelo desenvolvimento do UOL Marketplace. As respostas são anônimas e foram coletadas no período antes (jan/2014) e depois (dez/2014) da implementação da transformação *Lean* nos processos de desenvolvimento de software. A escala Likert usada para gradação das respostas pode ser vista no Quadro 4.

Tabela 7 – Respostas da Pesquisa de Maturidade em Jan/2014

Respondente	Delivery Contínuo	Densenvolvimento	Tecnologia	Processos	Qualidade
1	2	1	1	1	2
2	1	1	1	2	2
3	2	0	1	2	2
4	0	1	0	1	1
5	1	1	1	1	2
6	0	1	1	1	0
7	1	0	1	1	1

Fonte: elaborado pelo autor

Tabela 8 – Respostas da Pesquisa de Maturidade em Nov/2014

Respondente	Delivery Contínuo	Densenvolvimento	Tecnologia	Processos	Qualidade
1	4	4	N/A	4	4
2	3	4	1	3	4
3	4	5	N/A	4	4
4	3	3	N/A	3	3
5	2	3	N/A	3	3
6	2	3	N/A	4	2
7	2	3	N/A	2	2

Fonte: elaborado pelo autor

ANEXO A