Fatec Cruzeiro Prof. Waldomiro May
Revista Científica H-TEC Humanidades & Tecnologia

ESTUDO DA APLICAÇÃO DAS METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE TRADICIONAIS E ÁGEIS UNIFICADAS AO DESENVOLVIMENTO LEAN DE SOFTWARE

Autores
André Philipe Santos Pinto Ribeiro[1]
Ligia Maria T. de F. Brezolin[2]
Bruno Bustamante Ferreira Leonor[3]

EIC ENCONTRO DE INICIAÇÃO CIENTÍFICA

Resumo
A procura por excelência de produtos muitas vezes se apresenta em melhorias em seu processo de desenvolvimento. O objetivo desta pesquisa é mostrar vantagens da aplicação das metodologias de desenvolvimento de software e desenvolvimento lean, bem como propor estudos de caso com conteúdos comparativos que demonstrem os valores de antes e depois da aplicação dos conceitos aqui apresentados. O estudo da aplicação dessas práticas no dia-a-dia das empresas produtoras de software é necessário para o sucesso dos projetos com vistas às vantagens dessas práticas. Espera-se contribuir com a comunidade de desenvolvimento de software com o material desta pesquisa, alcançando a excelência nos processos de desenvolvimento.
Palavras-chave: Lean. Desenvolvimento. Software. Metodologias. Ágil.

Abstract
The demand for excellence products show in improvements in its development process. The objective of this research is to demonstrate to the reader the advantages of applying the Lean software development methodologies to a project as well as to propose case studies with comparative contents of before and after the application of the software development. This study of the application of the software development methodologies in the day-to-day of the software producing companies is increasingly necessary for the success of the projects and it also shows the advantages of using them. Is hoped to contribute whit the development community of software development with this research, achieving the excellence in its the development process.
Keywords: *Lean. Development. Software. Methodologies. Agile.*

Introdução

Com o crescimento da tecnologia em geral, a demanda por desenvolvimento de software se torna maior, já que, vinculado à tecnologia, transformar as necessidades do usuário em funcionalidades de software é premente.

Tendo em vista o desenvolvimento de novos projetos, é visível instabilidades no processo de software, visto que o fracasso de muitos projetos não é incomum neste ambiente, produto de equipes sobrecarregadas que, muitas vezes, são até superdimensionadas tornando objetivos inalcançados e prazos descumpridos. Assim, torna-se fator negativo para aqueles que esperam produto proveniente da comunidade de desenvolvimento de sistemas em geral.

Existem metodologias para desenvolvimento de software que visam sanar os problemas

[1] Graduando no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas na Fatec
[2] Docente na Fatec Prof. Waldomiro May. Email: ligia.brezolin@gmail.com
[3] Docente na Fatec Prof. Waldomiro May

que a comunidade de desenvolvimento apresenta com relação ao processo adotado para criar seus produtos. Estas metodologias podem ser denominadas tradicionais como o modelo cascata ou métodos ágeis como o Scrum.

Além das conhecidas, também há a possibilidade de aplicar junto as metodologias de desenvolvimento o desenvolvimento Lean de software, ideologia que pode ser desconhecida pela comunidade de desenvolvimento de software.

Embora não exista promessa de cem por cento de sucesso ao seguir um ou outro método para desenvolvimento de projetos, é sempre claro que existe oportunidades de se melhorar processos para buscar a excelência dos produtos.

Como objetivo geral tem-se: contribuir com a comunidade de desenvolvimento de software demonstrando melhorias que visam aumentar a taxa de sucesso dos projetos de desenvolvimento de software em geral, utilizando as metodologias existentes como cascata e Scrum, trazer ao conhecimento do leitor o desenvolvimento Lean de software e propor o uso dos conhecimentos e de boas práticas na aplicação correta das metodologias e ferramentas, para atingir melhores resultados em projetos de desenvolvimento de software.

Em busca da melhoria contínua de qualquer processo, ao identificar indicadores baixos de sucesso dos projetos de software, faz justificar a escolha deste tema que deve impactar diretamente na taxa de sucesso dos produtos de software.

É preciso seguir um roteiro de atividades para desenvolver software. A apresentação de metodologias para construí-lo é exposta por Sommerville (2004, p. 7), que demonstra a metodologia cascata como a tradicional para a comunidade de software. Já Sutherland (2008, apud PINTO, 2011, p. 45) diz que Scrum foi formalizado em 1995, trazendo novas práticas que visam executar tarefas, em que o prazo é mais curto, dividindo tarefas entre as partes interessadas.

Womack e Jones (1994, apud PINTO, 2011, p. 36) já enfatizam a criação da produção enxuta no meio automotivo, mas não deixam como obrigação a aplicação única destas práticas nesse setor, que tem como fundamento eliminar todo o desperdício, que, segundo Steffen (2011, p. 2), é tudo que não agrega valor ao produto final.

A metodologia principal para o desenvolvimento deste trabalho é efetuar pesquisas bibliográficas em livros, artigos, sites, entre outros, apresentar os conceitos que contribuem com a comunidade de desenvolvimento de software no seu processo de desenvolvimento e executar estudo desses conceitos demonstrando as vantagens de sua utilização.

Espera-se contribuir com a comunidade de desenvolvimento de software em seu

processo, trazer ao conhecimento do leitor o estudo da aplicação de boas práticas demonstrar suas vantagens, para haver impacto positivo nos indicadores de sucesso dos projetos de desenvolvimento de software da comunidade de TI.

1. Fundamentação Teórica

Nesta seção serão apresentados os estudos já levantados a respeito das metodologias, suas características, em especial, Cascata e Scrum, e o desenvolvimento Lean de software, que pode ser agregado no processo de desenvolvimento de software unificados às metodologias tradicionais e ágeis.

1.1. Metodologias para desenvolvimento de Software

Segundo Leandro (2010), desenvolver softwares é a elaboração e implantação de sistemas computacionais e transformação da necessidade do usuário ou de um segmento do mercado em produto de software. Medeiros (2013) diz que para desenvolver software é preciso seguir um roteiro de atividades e tarefas visando alta qualidade. Um processo de software torna-se, então, uma sequência de passos a ser seguida com o objetivo de desenvolver o produto. Ignorar um processo organizacional para o desenvolvimento de um projeto pode resultar em esforços desnecessários, perda da possibilidade de atingir o objetivo e até mesmo o caos. Conforme estudo realizado por Johnson (1999, apud FRANCO, 2007, p. 15), identificou-se já, em 1999, 74% de projetos mal sucedidos, cujas as causas foram relacionadas a prazos, orçamentos ou atendimento totais de requisitos até completo caos e cancelamento. Apenas 26% de projetos obtiveram êxito, aponta a pesquisa.

Existem diversas metodologias utilizadas pelos gestores de projetos hoje em dia para desenvolver os projetos de software. Fica a cargo do profissional definir qual será a prática metodológica que ele irá utilizar, podendo ainda utilizar-se apenas de seu instinto no desenvolver das atividades, o que se torna arriscado para o êxito do projeto.

1.1.1. Metodologias Tradicionais

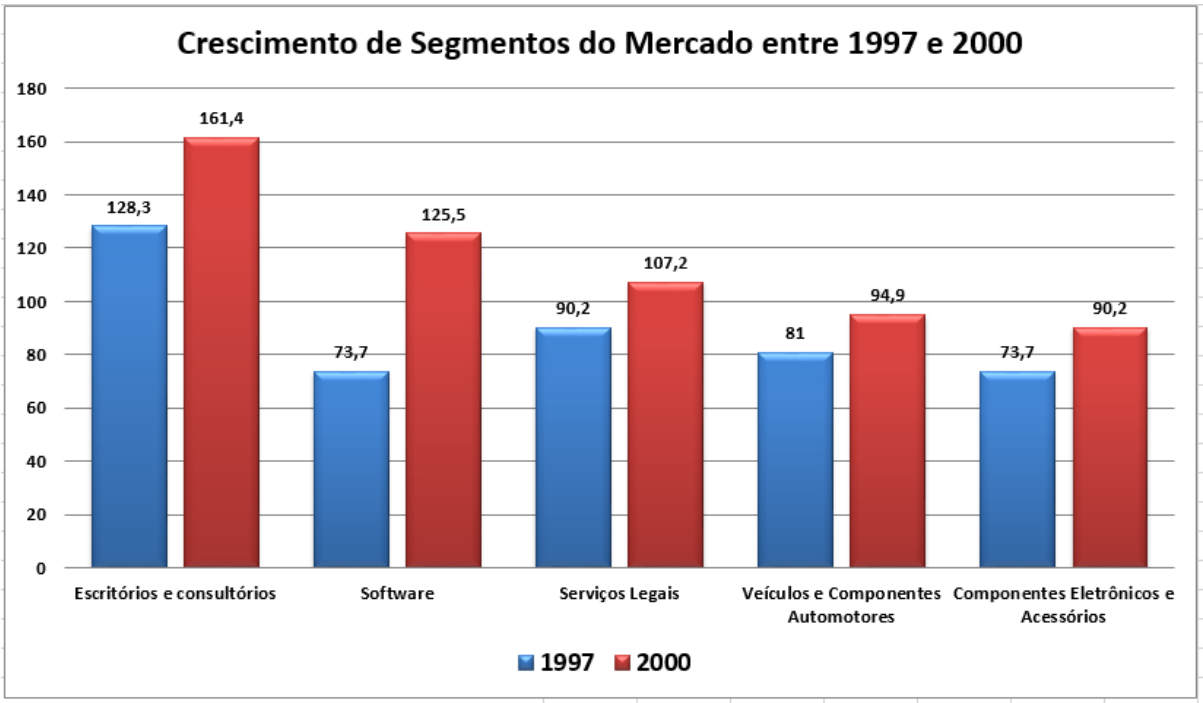
A comunidade de TI (Tecnologia da Informação) tem como métodos tradicionais, aqueles que não fazem parte das metodologias ágeis criadas mais recentemente.

Segundo Sommerville (2004, p. 7), o processo de software deve seguir quatro etapas básicas: especificação do software, desenvolvimento do software, validação do software e evolução do software.

Como método tradicional mais comumente usado e descrito pela literatura, tem-se adotado o Modelo Cascata, proposto por Winton Royce, em 1970, com a finalidade de atender as necessidades de controlar projetos personalizados de grande escala (FRANCO, 2007).

Extremamente estruturado, vinha na contramão de tudo que já era proposto. Medeiros (2013) expõe em seu estudo sobre modelos prescritivos que, como a revolução da produção de software era grande, cresceu, em contrapartida, a necessidade de se organizar os processos de produção de software, em busca da obtenção de maior taxa de sucesso. Na Figura 1 observa-se o crescimento de alguns segmentos do mercado entre 1997 e 2000, com destaque para software.

Figura 1: Crescimento de seguimentos do mercado entre 1997 e 2000



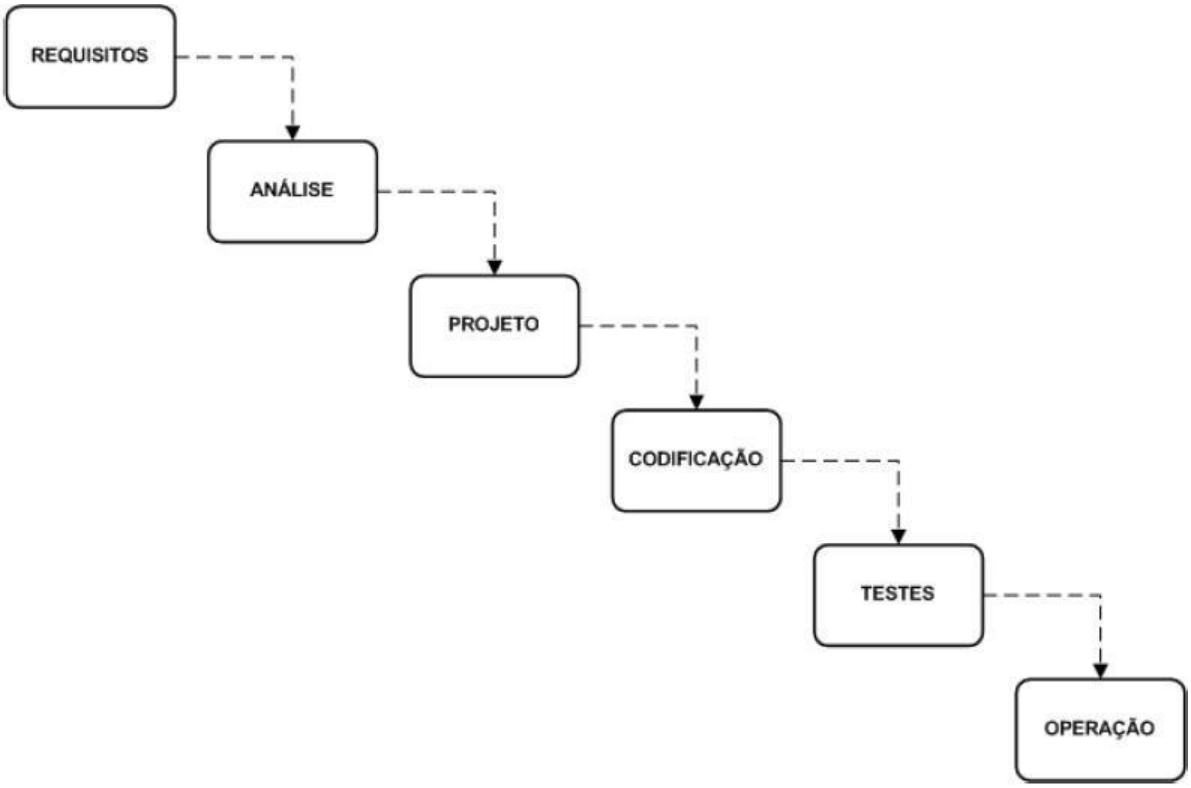
Fonte: Franco (2007, p. 2)

Como pode-se observar na Figura 1, Franco mostra o crescimento de cinco segmentos do mercado, cujo o foco se dá em Software e seu maior crescimento em relação a todos os outros setores. Ao analisar essa partícula da história, é possível mensurar o quão necessária foi a organização dos processos de software que Winton Royce trouxe em 1970.

A aplicação do modelo cascata é mais indicada quando se compreende bem o problema e tem o escopo bem definido. Uma sugestão de aplicação é para modificação em sistemas existentes, em que a ideia se torna mais coesa e há capacidade de seguir um fluxo para desempenhar as tarefas.

Medeiros (2013) descreve o modelo cascata como ciclo de vida clássico ou tradicional, oriundo como uma abordagem sequencial sistemática para o desenvolvimento de software , conforme Figura 2.

Figura 2: Ilustração do ciclo cascata



Fonte: Franco (2007 p. 12)

Conforme a Figura 2, o modelo cascata apresenta em sequência conjuntos de processos que devem ser levados em consideração ao utilizar esta metodologia.

Descreve Franco (2007 p.12) que a fase de Requisitos é considerada rigidamente de maneira prévia o levantamento de todas as necessidades que o sistema irá atender, todos enumerados e validados com as partes envolvidas.

Deve-se fundamentar o detalhamento de todos os requisitos gerando os requisitos de sistema, a partir do que já foi levantado e acordado como escopo do projeto. Nesta etapa deve haver coesão no que será definido para que o projeto possa seguir as etapas sem retornos.

Na fase de análise, é feito todo o levantamento obtido a partir dos requisitos. Diagramas de fluxo de dados, de estado, de classes, além do cronograma do projeto que prevê a finalização e outras aplicações de engenharia de software. Usualmente, nessa etapa obtêm- se dos desenvolvedores um conjunto de orçamento detalhado.

Franco (2007 p.13) deixa claro que essas duas fases são de extrema importância para definição do projeto, uma vez que dão o embasamento para suportar toda sua continuidade e garantia de sucesso.

Projeto é a fase que contém quatro atributos distintos, segundo descreve Franco (2007 p.13): estrutura de dados, arquitetura de software, detalhes procedimentais e caracterização da interface. Esta etapa gera uma significativa parte do documento e descreve em nível mais profundo o que é levantado nas etapas anteriores, deixando mais real aos olhos dos desenvolvedores, antes da codificação.

A etapa de codificação traz a tradução de tudo que foi executado até o momento em linhas de código. Todas as funcionalidades levantadas devem ser configuradas e construídas pelos programadores nesta etapa, que deve ser composta minuciosamente comprometida com a documentação construída até o momento.

Finalizando a etapa anterior, obtém-se finalmente o software em execução nas mãos do cliente. Em grande parte dos projetos, Franco (2007 p.13) aponta que existe a necessidade de adicionar ou modificar funcionalidades, atividade esta que pode ou não ser incluída em contrato previamente.

Cada etapa do modelo cascata só pode ser iniciada a partir do momento que a anterior está formalmente encerrada, isto deve-se à distinção que cada fase tem como característica.

Raramente esse método é seguido por programadores já que sua complexidade e rigidez torna o desenrolar de um projeto de software bastante complicado, ainda mais quando o cliente ou o analista não conhece por completo as necessidades a serem atendidas, tornando-se mais adequado o modelo cascata aos projetos quando se conhece completamente o escopo.

1.2. Metodologias ágeis

Franco (2007 p. 17) ressalta o uso desenfreado do termo ágil que muitas vezes gera desconfiança em relação às metodologias ágeis. É importante definir com critérios o termo ágil.

O conceito de agilidade, segundo Highsmith (2002, apud FRANCO, 2007, p.17), é considerado como a capacidade de criar e responder a mudanças que um ambiente turbulento como o de negócios pode criar como forma de manter lucros neste ambiente.

O movimento ágil foi formalizado na indústria no ano de 2001 como “Manifesto Para o desenvolvimento Ágil de Software”

Fadel e Silveira (2010 p.6) citam a ideologia levantada no conselho a respeito do Manifesto Ágil que traz consigo quatro pilares básicos:

- Indivíduos e interações com mais importância que processos e ferramentas;
- Software funcional é mais importante que documentações extensas;
- Colaborar com o cliente tem mais importância que negócios e contratos;
- Adaptar-se a mudanças necessárias é mais importante que seguir o plano inicial.

1.3. Scrum

Segundo Sutherland (2008, apud PINTO, 2011, p. 45), o Scrum foi uma prática que surgiu em meados dos anos 1990, mais exatamente formalizado em 1995. Destaque para o foco desta metodologia em ter sua aplicação em situações em que planejamento em longo prazo se mostra difícil.

Sutherland (2008, apud PINTO, 2011, p. 46) deixa claro que Scrum veio como uma maneira de controle para processos de complexibilidade maior, com comportamentos imprevisíveis em seus segmentos como prazos, recursos, tecnologias, entre outros.

Não é objetivo do Scrum definir processos, mas sim trazer definições de como as equipes devem trabalhar para atingir o objetivo de seus projetos. Tem como papéis Scrum Master, Product Owner e o Time.

O Scrum Master é colocado a nível de Gestor de Projetos. Ele é o responsável por facilitar o caminho do Time, gerenciar o ritmo do trabalho e o funcionamento do Scrum. Ele deve garantir o fluxo de trabalho, bem como o compromisso de garantir o comprometimento de todas as boas práticas do Scrum.

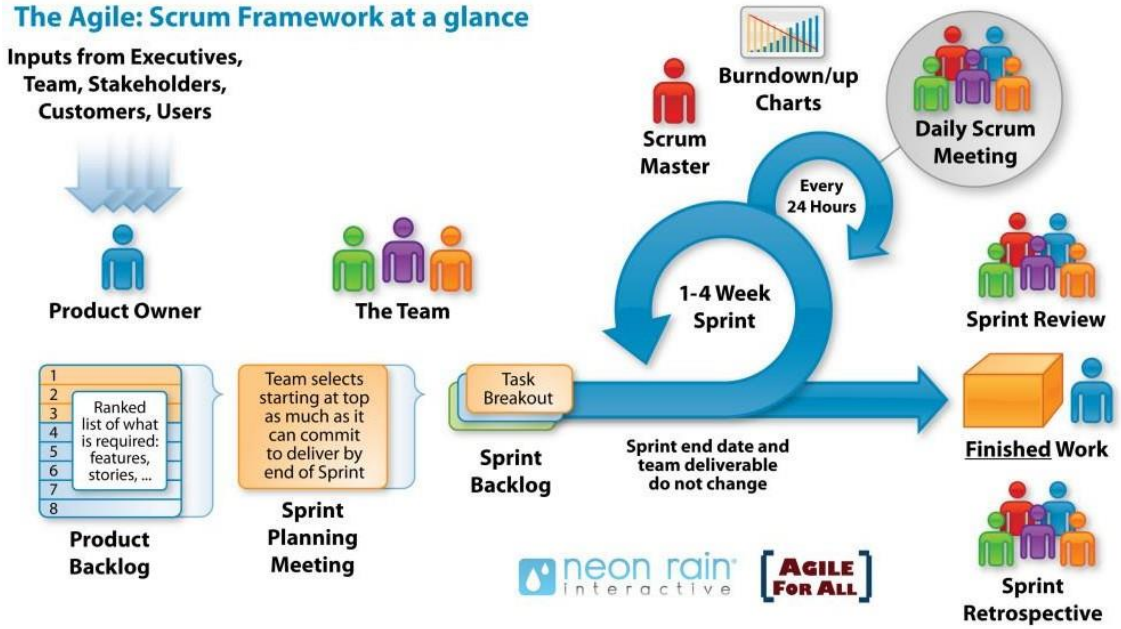
O Product Owner é o validador do produto. Ele é a palavra final na aceitação do que é produto do projeto. Deve adaptar o produto caso haja novas necessidades bem como garantir a criação dos requisitos funcionais, não funcionais e funcionalidades que agregam valor ao negócio implementadas o mais possível.

O Time também é parte integrante do Scrum. Torna-se responsável pelo seu próprio gerenciamento, deixando o Scrum Master no papel de facilitador no cotidiano da equipe. De forma auto gerenciável, devem decidir entre si a divisão das tarefas, buscando a finalidade de transformar requisitos em funcionalidades.

Spinola (2012) demonstra a prática das reuniões, conforme a Figura 3, que expõe um Sprint, ou seja, um ciclo de tempo para execução de tarefas organizadas determinado pela

equipe, geralmente entre 2 a 4 semanas.

Figura 3: Ciclo de entregas Scrum.



Fonte: http://www.agileforall.com.

Na Figura 3 são expostas fases do ciclo de vida Scrum, com entregas contínuas. Fadel e Silveira (2010, p. 13) expõem o significado de cartões no ciclo de entregas Scrum:

Product Backlog são todas as características listadas que o produto deve possuir e que ainda não foram desenvolvidas;

Sprint Backlog é um subconjunto da Backlog do produto que foi definido pelo cliente para ser implementado durante o Sprint atual. Não deve haver modificação deste subconjunto dentro do Sprint vigente;

Backlog do Sprint é a lista priorizada a partir das tarefas obtidas do Backlog selecionado;

Backlog de Impedimentos são os obstáculos listados e identificados pela equipe que não são pertencentes ao contexto de desenvolvimento;

Gráficos de Acompanhamento são elementos que devem medir a quantidade de trabalho que ainda resta para atingir a finalização de um produto ou Sprint. É recomendado fazê-lo para várias etapas do projeto.

As reuniões devem ser diárias, deve-se garantir que durem entre 10 e 15 minutos, apenas para alinhar entre a equipe a situação dos backlogs definidos. Ao final de cada Sprint, uma reunião para revisão e planejamento do próximo Sprint.

1.4. Lean Manufacturing

Segundo Ghinato (2008, apud PACHECO, 2014, p. 3), o sistema de produção enxuta mais recente conhecido é o sistema Toyota de Produção (STP - Toyota Production System).

O sistema Toyota de Produção, de acordo com Shingo (1989, apud LIMA, 2012), tem como pedra fundamental a eliminação de desperdícios. Qualquer atividade que não agregue valor ao produto final é um foco de desperdício a ser eliminado. O objetivo do sistema de produção Lean é otimizar a organização de forma que atenda necessidades no menor prazo, com maior qualidade aumentando a segurança e a moral dos colaboradores da organização.

1.5. Desenvolvimento Lean de Software

Womack e Jones (1994, apud PINTO, 2011, p. 36) comentam que o surgimento da produção enxuta no setor automotivo não dita a obrigatoriedade da aplicação exclusiva dela a este setor, e sim, deixa liberdade para a aplicação a diversos setores.

Respeitando as restrições, percebe-se a possibilidade então de aumentar a eficácia do processo de desenvolvimento de software.

Lima (2012) enfatiza que nem todas as práticas são aplicáveis, já que software é um produto intelectual e de maior customização que automóveis. O sistema de produção Lean oferece a capacidade de se buscar por desperdícios modificando de maneira radical a maneira com que se desenvolve softwares hoje em dia.

1.5.1. Sete Princípios Lean

Segundo Pinto (2011,p. 36), a aplicação da produção enxuta no desenvolvimento de software se dá compreendendo sete princípios: eliminar desperdícios, incluir qualidade ao processo, amplificar o aprendizado, adiar decisões ao máximo, entregar o mais rapidamente possível, delegar o poder à equipe, incorporar integridade e otimizar ao todo.

1.5.1.1. Desperdícios

Steffen (2011) define desperdício como qualquer esforço imputado no processo de desenvolvimento que não agrega valor ao produto final que é percebido pelo cliente. No caso de software pode ser considerado superficialmente: burocracias, retrabalhos, processos extras

e funcionalidades desnecessárias podem ser considerados desperdícios já observáveis em primeira instância. Identificados existem 7 tipos de desperdícios:

1.5.1.1.1. Requisitos

Se especificados cedo demais, acabam perdendo a credibilidade, tornando-se as vezes desnecessários ou obsoletos, resultando em funcionalidades não eficazes que travam ou que não atendem às reais necessidades do cliente, necessitando de retrabalhos posteriores.

1.5.1.1.2. Processos

Burocracias ou passos a mais que tornam o processo de desenvolvimento cada vez mais amarrado, havendo documentações extensas e desnecessárias que acumularão apenas poeira em gavetas, que, por fim, não irão tornar o produto mais eficiente.

1.5.1.1.3. Funcionalidades a mais

Steffem (2011) também expõe que 80% das funcionalidades são desnecessárias para o cliente, assim, mais uma vez mostra que é possível retirar o que não agrega valor ao produto final. Código que dificulta manutenções e exigem dimensionamento maior de infraestrutura para funcionalidades que muitas vezes nunca serão realmente utilizadas.

1.5.1.1.4. Troca de tarefas

Revezar entre tarefas também causa desperdício de tempo, já que no revezamento de atividades é necessário relembrar de onde parou, retomar o ritmo pode custar tempo que também é o vilão para o desperdício de recursos.

1.5.1.1.5. Atrasos

Em geral aumentam os custos do projeto, geralmente com multas contratuais, portanto, trazem desconforto e descrédito diante do cliente, o que pode levar a empresa à falência. Um bom produto entregue atrasado pode não ser mais a necessidade do cliente.

1.5.1.1.6. Defeitos

Prevenir defeitos por meio de inspeção impedindo que chegue ao cliente é bom, porém não é o que uma equipe eficaz faz. O defeito desperta o retrabalho, vilão do desperdício. Fazer certo da primeira vez é o correto, bugs não agregam valor além de insatisfazer qualquer cliente.

1.5.1.1.7. Movimentação

A conversa muitas vezes pode resolver inúmeras dúvidas e esclarecer várias questões. Trazer o cliente para perto é mais interessante do que ler páginas e páginas de documentos que ocasionam perca de tempo e muitas vezes ofuscam a percepção de ideias chave para o desenvolvimento.

1.5.1.2. Qualidade

Atributo que aos olhos do cliente é inegociável. Poppendieck e Poppendieck (2001, apud STEFFEN, 2011) revelam que qualidade percebida pelo cliente alcança a totalidade do produto sendo equilibrado entre suas funções, usabilidade, economia e confiabilidade, causando um encanto no cliente. Traz também a qualidade conceitual a nível de sistema, com relação a sua arquitetura, facilidade de uso a nível do cliente e manutenção preparada e facilitada.

1.5.1.3. Amplificar o aprendizado

Segundo Steffen (2011), é um princípio que auxilia equipes na criação do seu conhecimento. No desenvolvimento de software há implementação de ideias o tempo todo, o conhecimento auxilia na criação de boas ideias e acelera esse processo. Praticar ciclos de feedback, desenvolvimento interativo, meios de compartilhar informações são úteis para atender este princípio.

1.5.1.4. Tomar decisões com mais prioridade sobre o assunto

Geralmente de maneira mais tardia, é uma medida que visa diminuir a incerteza. Quando mais informações se obtiver para a tomada de decisão, mais certeira será a consequência desta decisão em relação ao objetivo final do produto.

1.5.1.5. Entregas rápidas

Princípio que garante a colheita de feedback e oferece a oportunidade de aprender com erros. Garantia de que o cliente receberá o que importa hoje e não o que precisava ontem.

1.5.1.6. Respeitar as pessoas

Princípio fundamental. O produto de software muitas vezes é reflexo da equipe que o desenvolve e criar um ambiente que favoreça também o desenvolvimento de pessoas, constrói um relacionamento de respeito entre empresa e pessoas também.

1.5.1.7. Otimizar o todo

Do início ao fim da menor tarefa para a maior. Muitas vezes várias pequenas otimizações significam um grande resultado no produto final. Valorizar o desempenho da equipe, medir tempos de ciclo, satisfação do cliente entre outras práticas que auxiliam na visualização de possibilidades de otimização.

1.6. DMAIC

Conforme Andrade (2017), a Metodologia DMAIC é uma das ferramentas da gestão da qualidade que são mais utilizadas de grande a pequenas empresas.

A gestão da qualidade é grande responsável por, atualmente, dentro das organizações, garantir controle dos processos, atividades e equipes, além de apresentar visão ampla e

detalhada do negócio e garantir que as melhorias sejam aplicadas e que custos sejam efetivamente reduzidos. O Ciclo DMAIC traz controle e organização das ações, busca efetiva pela solução de problemas e torna quantitativo e organizado o planejamento estratégico, além de fornecer mais eficiência à ele.

1.6.1. Definição de DMAIC

A sigla DMAIC tem como tradução no português: Definir, Medir, Analisar, Melhorar (improve) e Controlar. É determinada pela nomenclatura que cada etapa do ciclo propõe.

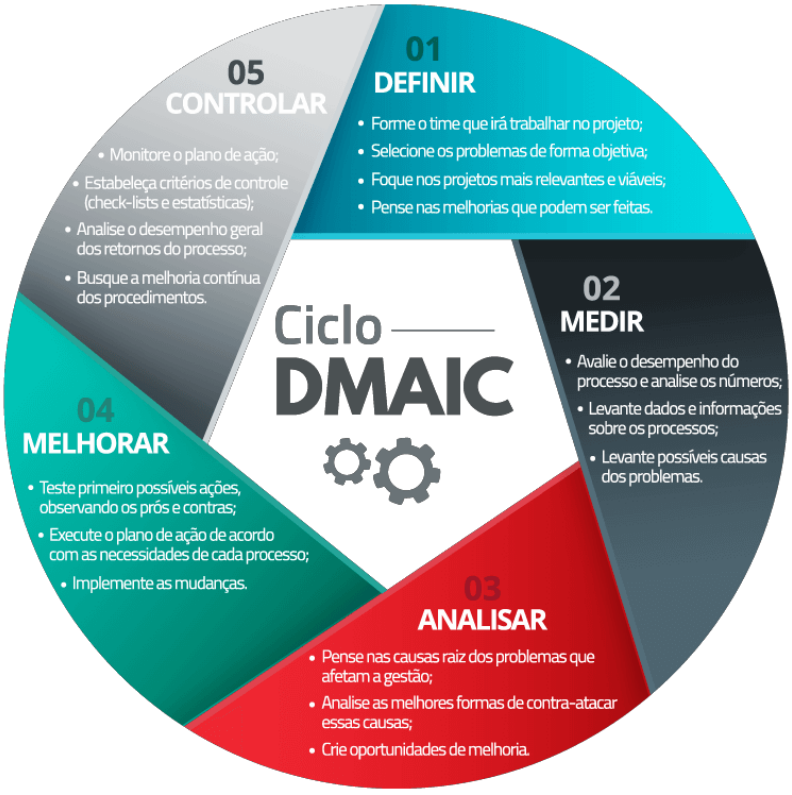
Ele se apresenta no formato de um roteiro e possui 5 etapas. É construído a partir da Metodologia Lean Six Sigma e ajuda empresas a utilizar a análise dos dados obtidos por meio da gestão e produzir tomadas de decisões com base nestes dados.

A missão empregada ao Ciclo DMAIC é fornecer base para que as empresas melhorem seus processos, foquem no aprimoramento de atividades, entreguem o melhor ao seu cliente e utilizem ao máximo seus recursos, produzindo mais com menos.

1.6.2. Etapas ciclo DMAIC

Serão abordadas as 5 etapas do ciclo DMAIC

Figura 4: Ciclo DMAIC



Fonte: www.siteware.com.br

Na Figura 4, Andrade (2017) expõe as 5 etapas do ciclo DMAIC.

Definir é a etapa em que se estabelece o planejamento do projeto, reconhecimento do ambiente de desenvolvimento, forma-se o time e objetivos.

Em Medir avalia-se o estado atual do processo, levanta-se informações sobre e as possíveis causas dos problemas observáveis.

Analisar é a etapa de estudos mais aprofundados, transformação das possíveis causas em causas raízes, criação de ações propostas para as causas raízes e criação das oportunidades de melhoria.

Melhorar é a etapa de ação, implementar o projeto ou melhorias, avaliar os prós e contras e execução do plano de ação.

Controlar é o monitoramento do projeto ou melhoria aplicada. Estabelece critérios para se controlar e analisa o desempenho do processo melhorado e avalia possíveis melhoras em relação ao estado inicial, levantado na etapa Medir. Inclui a busca por melhoria contínua dos processos.

2. O caminho da pesquisa

Estudo de caso: Grupo de desenvolvimento Turma: 2º semestre 2017-2

Este estudo de caso foi desenvolvido para observar o comportamento da aplicação das técnicas e práticas do Lean unificados ao SRUM.

A seleção foi um grupo de desenvolvimento de trabalho da disciplina de Engenharia de Software na Fatec Cruzeiro Professor Waldomiro May.

2.1. Equipe

Para este estudo de caso foi selecionada a equipe: Alunos do segundo semestre de 2017-2 da disciplina de Engenharia de software I, sem nenhuma experiência na área além dos desenvolvimentos de trabalhos acadêmicos. Para metodologia de controle de seus processos, foi sugerida a utilização do SCRUM, metodologia aceita pelos integrantes do grupo para o desenvolvimento de suas atividades.

2.2. Produto

O produto desenvolvido neste estudo de caso foi o Sistema para FastFood 2.0 e sua documentação, atendendo os requisitos determinados pelo cliente do grupo, que resumidamente seria: agilizar o atendimento, dispor uma interface agradável e intuitiva.

O cliente do grupo tratava-se de outro grupo de desenvolvimento e eles forneciam as informações na posição de clientes para que o grupo de desenvolvimento, a partir delas, buscasse atendê-las gerando um produto com funcionalidades esperadas pelo cliente.

2.3. Contrato

No início de suas atividades, foi proposto o desenvolvimento de um contrato que é o acordo do start de projetos Lean 6 Sigma adaptado para desenvolvimento de sistemas. Este contrato possui premissas para o produto a ser desenvolvido e seus processos de desenvolvimento. É firmado entre a equipe e o cliente, compromisso total das partes envolvidas.

2.4. Desenvolvimento

No desenvolvimento do sistema, foi implantado o desenvolvimento utilizando a Metodologia DMAIC: Definir, medir, analisar, melhorar e controlar, metodologia utilizada em Lean 6 Sigma para buscar causa(s) raiz(es) e desenvolver soluções para tais, controlar projetos e controlar e observar andamento de projetos.

Na etapa definir, a equipe de desenvolvimento deve firmar o contrato entre as partes envolvidas, inserindo nele o objetivo que o produto deve contemplar, os problemas antes da intervenção da equipe, o cenário esperado após a intervenção da equipe, o papel de cadaintegrante da equipe, e a descrição dos processos do desenvolvimento dentro de cada campo da metodologia DMAIC.

2.5. Scrum no Projeto

O SCRUM foi utilizado pela equipe para controle da evolução dos processos relacionados ao desenvolvimento do projeto, como um todo, desde levantamento de requisitos, passando pela documentação até o software propriamente dito.

3. Materiais e Métodos

A metodologia da presente pesquisa se dá por meio de leitura de trabalhos acadêmicos, artigos, livros e sites. A busca está relacionada por materiais que tragam informações a respeito de engenharia de software, metodologias para desenvolvimento de software e pensamento Lean unificado ao desenvolvimento Lean de software.

O levantamento destes estudos e cruzamento de informações e pensamentos de autores trarão como resultado contribuição para a comunidade de desenvolvimento de software demonstrando por meio deste estudo as vantagens da aplicação correta dos conceitos aqui apresentados no processo de desenvolvimento de software seja em empresas ou desenvolvedores autônomos.

4 Análise e discussão dos resultados

No fim do estudo de caso, a equipe gerou o produto atendendo as expectativas do cliente, porém, no decorrer das atividades, foi possível observar que não houve aplicação comprometida de toda a metodologia LEAN e SCRUM. As atividades de controle não foram executadas com comprometimento e a não aplicação do SCRUM de maneira correta acarretou em sobrecarga de atividades na equipe de desenvolvimento.

Havia inexperiência da equipe quanto a competências técnicas e isto gerou insegurança e sobrecarga de atividades de outras disciplinas. Estes eventos atrapalharam o controle de suas atividades e, na visão do gestor, o processo de controle das atividades propostas para desenvolver o produto traria mais carga ainda de atividades.

É um equívoco observar o controle dos cartões e reuniões como uma atividade adicional, pois esta atividade é a matriz de controle do desenvolvimento do projeto. Nela devem estar o comprometimento da equipe e do gestor, que no controle das pendências observam os empecilhos e destacam atitudes para o cumprimento de cada tarefa, resultando no produto esperado pelas partes interessadas.

Considerações Finais

A Tecnologia da Informação sempre foi o setor de vanguarda dos avanços da indústria e é vital a sobrevivência de empresas que oferecem produtos neste ramo. Esta sobrevicência só existe a partir de recursos para a produção de seus produtos como Sistemas, por exemplo.

O pensamento Lean tem invadido toda a indústria de manufatura e é promissor em criar caminhos para que as empresas se tornem mais competitivas ao mercado, fornecendo norte para projetos de redução de custos e alcance de maior produtividade com menos recursos, que se mostra como caminho para a garantia de sobrevivência no mercado.

Não é possível mostrar indiferença em relação à competitividade do mercado no ramo das empresas de Tecnologia da Informação. É necessário procurar ser competitivo e reduzir os recursos necessários para a produção de seus produtos. Nada mais justo do que observar os outros setores e trazer o pensamento Lean e adaptá-lo a esta realidade e às boas práticas já existentes na literatura e na área de Engenharia de Software.

Este trabalho contribui com a comunidade de desenvolvimento de software demonstrando que existe sim a possiblidade de alavancar os projetos e produzir mais utilizando menos recursos, por meio da eliminação dos desperdícios e otimização dos processos. É

necessário adaptar a esta realidade as métricas do pensamento Lean e aplicá-las a equipes experientes e fica este desafio: comprovar com medições e indicadores a melhora da eficiência após a aplicação das metodologias aqui apresentadas.

Referências

ANDRADE, Luiza; **O que é ciclo DMAIC e como utilizar?** www.siteware.com.br Disponível em < https://www.siteware.com.br/metodologias/o-que-e-ciclo-dmaic/ > Acesso em 14 Abr. 2018.

FADEL, Aline Cristina; SILVEIRA, Henrique da Mota. **Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean.** Universidade Estadual de Campinas , Libeira 2010. Disponível em: <http://www.ceset.unicamp.br/liag/Gerenciamento/monografias/Lean%20Agil_v8.pdfAcesso > em: 05 abr. 2017.

FRANCO, Eduardo Ferreira. **Um Modelo De Gerenciamento de Projetos Baseado Nas Metodologias Ágeis de Desenvolvimento de software e nos Princípios da Produção Enxuta.** 2007 Escola Politécnica da Universidade de São Paulo Disponível em <http://www.teses.usp.br/teses/disponiveis/3/3141/tde-09012008-155823/pt-br.php> Acesso em 15 Fev. 2017.

LEANDRO, Lucas. **Desenvolvimento de software.** © 2011 desenvolvimentodesoftware.xpg.uol.com.br. Disponível em: <http://desenvolvimentodesoftware.xpg.uol.com.br/desenvolvimento-de-software.html>. Acesso em: 18 mai. 2017.

LIMA, Hélder Seixas. **Lean Software: Eliminação de desperdício no processo de desenvolvimento.** Universidade Estadual de Montes Claros, Montes Claros Ago 2012. Disponível em: <http://documento.ifnmg.edu.br/action.php?kt_path_info=ktcore.actions.document.view&fDocumentId=3636> Acesso em: 20 de mai. 2017.

MEDEIROS, Higor. **Introdução ao Modelo Cascata.**© 2013 devmedia.com.br Disponível em: <http://www.devmedia.com.br/introducao-ao-modelo-cascata/29843> Acesso em: 1 mai. 2017.

PACHECO, Diego Augusto de Jesus. **Teoria das Restrições, Lean Manufacturing e Seis Sigma: limites e possibilidades de integração.** Universidade Federal do Rio Grande do Sul, Porto Alegre jan./jun. 2013. Disponível em: <http://www.scielo.br/pdf/prod/2014nahead/aop_prod1171_ao.pdf> Acesso em: 03 abr. 2017.

PINTO, Pietro Pereira**. Uma abordagem para a construção de Retrospectivas Scrum baseada nos conceitos de melhoria continua e Lean Software Development.** Universidade Federal de Pernambuco, Recife Ago. 2011. Disponível em: <http://repositorio.ufpe.br/bitstream/handle/123456789/1675/arquivo6809_1.pdf?sequence=1 > Acesso em: 04 abr. 2017.

SPINOLA, Rodrigo Oliveira. **Lições aprendidas na implantação de metodologias ágeis** – Revista Engenharia de Software Magazine 50. © 2012 devmedia.com.br Disponível em <http://www.devmedia.com.br/licoes-aprendidas-na-implantacao-de-metodologias-ageis-revista-engenharia-de-software-magazine-50/25373#comentariosArtigo> Acesso em: 25 fev. 2017.

SOMMERVILLE, Ian. **Livro Engenharia de Software** 2 ed. Rio de Janeiro: Editora Person Education do Brail 2004.

STEFFEN, Juliana Berossa, **Lean para desenvolvimento de Software, Afinal o que é isto?**.© 2011 ibm.com. Disponível em: <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/lean_para_desenvolvimento_de_sw_o_que_c3_a9_isso_afinal12?lang=en>. Acesso em: 02 abr. 2017.