

INSTITUTO TECNOLÓGICO DE AERONÁUTICA



Suelen Goularte Carvalho

Padrões Para Implantar Métodos Ágeis

*Trabalho de Conclusão de Curso em Tecnologia da
Informação – Área de Gestão Estratégica de Projetos
2012*

***Pró-Reitoria de Extensão e
Cooperação***

Número da CDU (tamanho 10)

Suelen Goularte Carvalho

Padrões Para Implantar Métodos Ágeis

Orientador

Prof. Dr. Eduardo Martins Guerra (ITA)

Curso de Especialização em Tecnologia da Informação
Gestão Estratégica de Projetos

SÃO JOSÉ DOS CAMPOS

INSTITUTO TECNOLÓGICO DE AERONÁUTICA

2012

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Carvalho, Suelen Goularte
Padrões Para Implantar Métodos Ágeis / Suelen Goularte Carvalho.
São José dos Campos, 2012.
135 f.

Trabalho de Conclusão de Curso – Especialização em Tecnologia da Informação. Gestão Estratégica de Projetos.- Instituto Tecnológico de Aeronáutica, 2012. Orientador: Prof. Dr. Eduardo Martins Guerra.

1. Gestão de Projetos. 2. Métodos Ágeis. 3. Padrões. I. Carvalho, Suelen Goularte. II. Instituto Tecnológico de Aeronáutica. III. Padrões Para Implantar Métodos Ágeis.

REFERÊNCIA BIBLIOGRÁFICA

CARVALHO, Suelen Goularte. **Padrões Para Implantar Métodos Ágeis**. 2012. 135 f. Trabalho de Conclusão de Curso. (Lato Sensu) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Suelen Goularte Carvalho

TÍTULO DO TRABALHO: Padrões Para Implantar Métodos Ágeis

TIPO DO TRABALHO / ANO: TCC / 2012

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias deste trabalho de conclusão de curso e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste trabalho de conclusão de curso pode ser reproduzida sem a autorização do autor.

Suelen Goularte Carvalho
Avenida dos Ourives, 458
04194-260 – São Paulo – SP

PADRÕES PARA IMPLANTAR MÉTODOS ÁGEIS

Essa publicação foi aceita como Relatório Final de Trabalho de Conclusão de Curso do Programa de Especialização em Tecnologia da Informação na Área de Gestão Estratégica de Projetos.

Eduardo Martins Guerra
Orientador

Paulo André Castro
Coordenador do Programa de Especialização em Tecnologia da Informação

Carmen Lucia Ruybal dos Santos
Pró-Reitora de Extensão e Cooperação

ITA

São José dos Campos, 17 de Fevereiro de 2012

*Á pessoa mais importante na minha vida e minha maior
inspiração, minha mãe Sonia Maria Goularte.*

Agradecimentos

Agradeço primeiramente e especialmente a pessoa mais importante da minha vida, minha mãe Sônia Maria Goularte, que me proporcionou o mais importante na vida: muito amor, carácter e educação possibilitando que eu tivesse ferramentas suficientes para trilhar um caminho e chegasse até este curso. Muito, muito obrigada!

A minha irmã, simplesmente por existir, e ser sempre minha irmã. Obrigada pelo carinho e cuidados.

Ao meu companheiro, confidente e amigo, David Robert Camargo de Campos, ao qual me apoiou, inspirou e motivou nos momentos mais difíceis e que sempre esteve ao meu lado.

Aos meus colegas de classe Pedro Bandeira, Mario Bandeira, Amanda Bennicasa, Fernanda Pellegrino, Saulo Carvalho, Victor Ralio, Daniel Ulbricht, Jairo “Master” e Thiago Luciano pelas brincadeiras, churrascos, amizade e todo o carinho durante todo o curso e depois. Pelo apoio, ideias, inspiração e amizade para continuar e concluir.

Aos professores do curso Prof. Guilherme Lopes e Prof. André Pereira que acreditaram em mim e foram mais que professores. Aos demais professores pelo empenho em nos ensinar e compartilhar experiências. Ao prof. Pellegrino, que me inspirou com sua integridade e objetividade.

Ao meu orientador Prof. Dr. Eduardo M. Guerra, que aceitou meu pedido de trilhar este caminho junto comigo sempre muito atencioso e exigente, que me fez trabalhar de fi a dar o melhor de mim e propondo ideias desafiadoras.

Aos meus entrevistados Daniel Cuckier, Guilherme Chapiewski, Maurício Matsuda e Fábio Akita pelo tempo, atenção e prestatividade em me receber de braços abertos.

Agradeço a todos que contribuíram de alguma forma, indireta ou diretamente, para a conclusão desta pesquisa. Aos autores das bibliografias. Aos que me acompanharam e estiveram ao meu lado neste tempo de curso.

Muito Obrigada!

Resumo

A pressão por entregas cada vez mais rápidas no desenvolvimento de software tem feito empresas buscarem conhecimento em métodos ágeis para executarem seus projetos. Porém, existe uma dificuldade das empresas em migrarem de suas metodologias atuais para os métodos ágeis. Esta dificuldade muitas vezes, faz com que as empresas a optem por continuarem com seus processos já existentes mesmo que não sejam mais eficientes para a execução dos projetos.

Este estudo trata de uma pesquisa exploratória com objetivo principal de documentar, ao molde de padrões, ações recorrentes na implantação de ágil, utilizadas com sucesso em diversos contextos. Os dados foram coletados através de entrevistas e estudos de caso. Esses padrões podem servir como guia para empresas que desejam implantar métodos ágeis.

Após a implantação, os projetos de software tendem a ser entregues em menos tempo, com mais qualidade e de acordo com as expectativas dos clientes.

Abstract

The pressure to delivery ever faster in software development has made companies seeking expertise in agile methodologies to develop their projects. However, it is hard for companies to migrate from their current methodologies for agile methodologies. This difficulty often means that companies to choose to continue with their existing processes even if they are not more efficient for the execution of projects.

This study is an exploratory research with the objective of documenting as patterns recurring actions in deploying agile, successfully used in many contexts. Data were collected through interviews and case studies. These patterns can serve as a guide for companies wishing to deploy agile.

After the implementation of agile methods, software projects tend to be delivered in less time, with more quality and in accordance with customer expectations.

Lista de Ilustrações

Figura 1 - Uso de funcionalidades de software.	18
Figura 2 - Tradução do gráfico de Joshua Kerievsky.	26
Figura 3 - Degraus para o sucesso.	28
Figura 4 - Excerto da linguagem dos Padrões Para Implantar Métodos Ágeis.	42
Figura 5 - Mapa Mental dos Padrões Para Implantar Métodos Ágeis	45
Figura 6 – Navegação de Padrões Para Implantar Métodos Ágeis.....	46
Figura 7 – Queda do Muro de Berlim. A comunicação sem barreiras.	48
Figura 8 - Barreiras físicas em forma de divisórias frontais.	50
Figura 9 – Mesmo as divisórias menores interferem na comunicação e integração do time...	50
Figura 10 - Escritório da empresa Jera sem divisórias e integrantes próximos.....	52
Figura 11 - Exemplo do Yahoo! ausência de divisórias laterais.	52
Figura 12 - Na Webgoal, ambiente sem divisórias e os integrantes próximos.	53
Figura 13 - Seja qual for a informação, coloque-as na parede.	55
Figura 14 - Ambiente não informativo e não interativo. Não há quadros nas paredes.....	57
Figura 15 - Ambiente não informativo e não interativo. Não há quadros nas paredes.....	58
Figura 16 - Quadro expondo arquitetura de sistema.....	59
Figura 17 - Ambiente do Yahoo! com monitor visível para o time que mostra os resultados dos build's.	60
Figura 18 - Ambiente Yahoo!. Informativo e interativo.....	60
Figura 24 - Neném dando seus primeiros passos. Fonte: http://bebegravidéz.com/tag/primeiros-passos/	63
Figura 19 - Canivete Suíço representa um time multifuncional.	67

Figura 22 – Testes são usados para a validação de conteúdo. Fonte: http://www.sempretops.com/wp-content/uploads/enem12.jpg	72
Figura 23 - Custo da Mudança vs. Tempo de Desenvolvimento [65].	77
Figura 20 – Velha dando bronca.....	78
Figura 21 – Cozinha antes e depois da refatoração.	83
Figura 25 - Mapa Mental dos Padrões Para Implantar Métodos Ágeis	88

Lista de Tabelas

Tabela 1 - Principais diferenças entre o modelo tradicional e o sobreposto [24]......	17
Tabela 2 - Resumo dos Padrões Para Implantar Métodos Ágeis.....	43
Tabela 3 - Lista de frameworks de criação de testes de código.	74

Listas de abreviaturas, siglas e símbolos

ROI	-	<i>Return on Investment</i> (Retorno sobre Investimento).
WIP	-	<i>Work in Progress</i> (Trabalho em Processo).
TDD	-	<i>Test-Driven Development</i> (Desenvolvimento Dirigido por Testes)

Sumário

1. Introdução.....	15
1.1. Motivação	16
1.2. Objetivo	19
1.3. Abordagem de Solução.....	19
1.4. Originalidade e Relevância.....	20
1.5. Organização do Trabalho.....	21
2. Fundamentação Teórica.....	22
2.1. Gestão de Projetos	22
2.2. Métodos Ágeis	23
2.3. Padrões.....	24
2.3.1. Formatos de Padrões.....	25
2.3.2. Formato Alexandrino	26
2.3.3. Formato Adotado	27
3. Pesquisa	33
3.1. Hipóteses.....	33
3.2. Processo de Pesquisa	33
3.2.1. Identificação de Padrões Existentes.....	35
3.2.2. Coleta de Dados.....	35
3.2.2.1. Entrevistas.....	35
3.2.2.2. Estudos de Caso	39
3.2.3. Análise dos Dados	39
3.2.4. Escrita dos Padrões	40
4. Linguagem de Padrões	41

4.1. Classificação dos Padrões.....	42
4.2. Navegação.....	46
4.3. Padrões Relacionados.....	47
5. Padrões Para Implantar Métodos Ágeis	47
5.1. DERRUBE AS BARREIRAS	48
5.2. ESCREVA NA PAREDE.....	55
5.3. COMECE ÁGIL NA GESTÃO.....	63
5.4. MONTE UM CANIVETE SUIÇO	67
5.5. FAÇA TESTES.....	72
5.6. AUTOMATIZE VALIDAÇÕES	78
5.7. FAÇA REFATORAÇÃO	83
6. Conclusão	88
7. Contribuições	89
8. Trabalhos Futuros.....	89
Referências	90
Glossário.....	97

1. Introdução

Com um mercado cada vez mais competitivo e exigente, o tempo passou a ser fator crucial para a sobrevivência das empresas [7]. Esta afirmação é enfatizada em [6] ao dizer que a demanda por software cresce e os prazos diminuem. Este fato leva as empresas a buscarem novas formas de executar seus projetos de software visando entregas mais rápidas e de maior valor para o cliente.

Os métodos ágeis de desenvolvimento de software vêm ganhando crescente popularidade desde o início da última década [3]. Eles são regidos pelo Manifesto Ágil de Desenvolvimento de Software [10][3].

O Manifesto Ágil para o Desenvolvimento de *Software* reúne quatro valores e doze princípios compartilhados por todos os métodos ágeis, será abordado mais detalhadamente na Seção 2.2. Dentre os métodos ágeis, podemos citar alguns que levam maior destaque, são eles: Scrum, XP, Lean e Kanban.

Segundo Dybå (2000) e Nerur et al. (2005) apud [3] os métodos ágeis podem ser vistos como uma reação aos métodos tradicionais, também conhecidos como dirigidos por planos, que enfatizam o planejamento e a predição para cada problema do desenvolvimento.

No entanto, esta área de pesquisa é muito carente. Dybå e Dingsoyr (2008) apud [3] citaram que apesar dos diversos artigos publicados sobre métodos ágeis, pouco se sabe sobre como esses métodos são realmente implantados nas empresas e que efeitos geram.

Ainda são muitas as dificuldades enfrentadas pelas empresas ao tentarem migrar de seus métodos atuais para métodos ágeis. Podemos citar, por exemplo, a dificuldade na mudança de valores [1], pessoas resistentes e impasses relacionados a infraestrutura como os principais.

Diante dessas dificuldades algumas empresas simplesmente continuam a executar os projetos das formas tradicionais utilizadas até então. Outras empresas tentam realizar a implantação e passam por frustrações por não obterem sucesso, o que normalmente leva a difamação do método, que recebe a culpa pelo fracasso. E outras, realizam a implantação com sucesso, mas não sabem exatamente como foi feito.


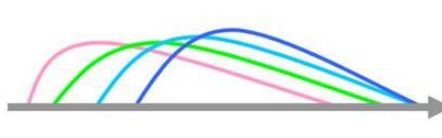
Este trabalho é composto pelo capítulo 2, que trata da fundamentação teórica. Capítulo 3 que trata da Pesquisa. Capítulo 4 que trata da Linguagem de Padrões. Capítulo 5 que trata dos padrões identificados. Capítulo 6 que trata de Resumo sobre os padrões identificados. Capítulo 7 que aborda o diálogo das entrevistas e eventos relacionados ao tema e capítulo 8 que trata da Conclusão.

1.1. Motivação

Em Fevereiro de 1986, os professores Hirotaka Takeuchi e Ikujiro Nonaka publicaram o artigo “*The New New Product Development Game*” (O Novo Novo Jogo de Desenvolvimento de Produto) na *Harvard Business Review*. Dentre as conclusões do estudo, os autores afirmam que a demora no lançamento de novos produtos, além de postergar o retorno sobre os investimentos, também poderiam ocasionar na perda de competitividade [24].

A Tabela 1, também fruto do estudo dos professores, apresenta as principais diferenças com relação ao modelo tradicional de desenvolvimento de produto (*waterfall*) e o modelo sobreposto (*overlap*) também chamado pelos autores de “Abordagem Rugby”.

Tabela 1 - Principais diferenças entre o modelo tradicional e o sobreposto [24].

Processo	Seqüenciado (Waterfall)	Sobreposto (Overlap)
		
Definição do Escopo	Grande detalhamento de requisitos. O escopo era definido no início do projeto e havia baixa tolerância à mudança.	O escopo era detalhado durante a execução do projeto. Existia grande flexibilidade para acomodar mudanças.
Condução e Prazos	Cada fase só começava após a conclusão da anterior. Os projetos tinham prazos de conclusão mais longos.	As fases podiam ser iniciadas mesmo sem a conclusão da anterior. Os prazos eram reduzidos significativamente.
Perfil da Equipe	Especializada, com atuação pontual. Cada um cuidava somente da sua parte. Baixo nível de colaboração e interação.	Multifuncional, com atuação permanente. Visão compartilhada do produto, com alto volume de colaboração e interação.
Modelo de Gestão	Centralizado (comando e controle). Baixo incentivo a autonomia e inovação.	Descentralizada. Equipes eram auto organizadas. Grande estímulo a autonomia e inovação.
Geração de Resultados	Prazos longos e baixa interação limitavam resultados. Baixo nível de transferência de conhecimento e evolução da equipe.	Agilidade e interação geravam resultados de impacto. Altos níveis de transferência de conhecimento e evolução da equipe.

O modelo sobreposto proporciona um escopo mais flexível, prazos de término menores, modelo de gestão inovador e resultados mais eficientes com uma equipe multifuncional. Enquanto no modelo tradicional, ou modelo sequenciado, o escopo e prazos são definidos no início do projeto, o que tornam difíceis mudanças futuras, o modelo de gestão é centralizado, os resultados gerados são menos eficientes e inovadores e a equipe é especializada.

O gráfico na Figura 1 a seguir, apresenta os percentuais de utilização de funcionalidades de *softwares* desenvolvidos com métodos tradicionais [44].

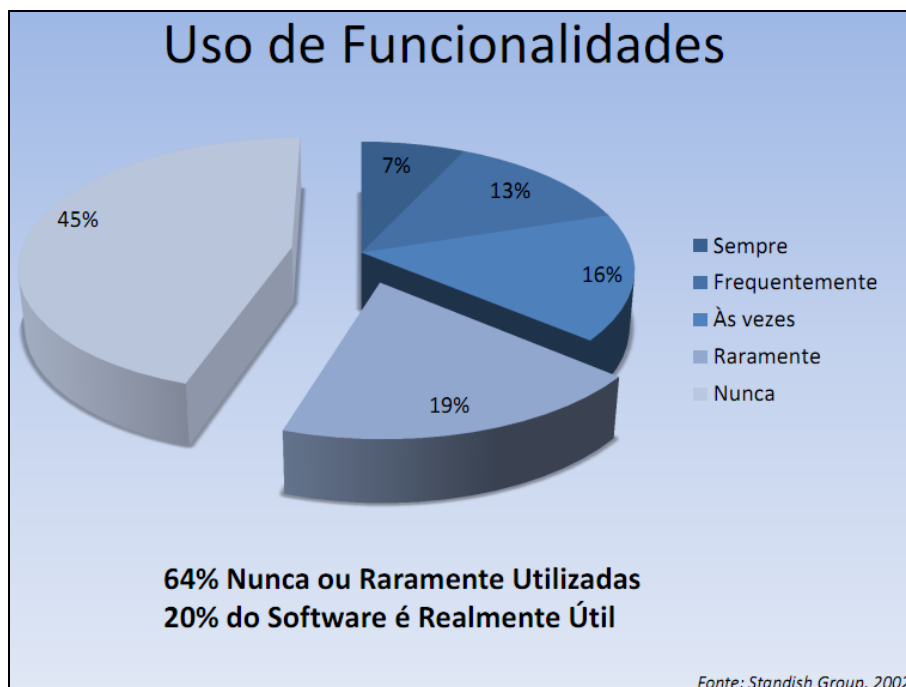


Figura 1 - Uso de funcionalidades de software.

Podemos observar na Figura 1 que quase 65% das funcionalidades são utilizadas nunca ou raramente.

Duas importantes conclusões dos professores, e muito relevantes ao contexto: “As empresas que adotavam o modelo sobreposto, conseguiam reduzir drasticamente seus ciclos de desenvolvimento, o que permitia a liberação de novos produtos para o mercado em períodos cada vez mais curtos.” e “... na maioria das vezes, esses produtos tornavam-se sucesso de vendas, uma vez superavam as expectativas dos clientes devido ao alto nível de atendimento das suas necessidades e também pela incorporação frequente de inovações.” [24]. Ou seja, naquela época já foi possível perceber alguns benefícios trazidos pelo modelo sobreposto frente ao modelo sequenciado.

O trabalho dos pesquisadores Hirotaka Takeushi e Ikujiro Nonaka, concluem que entregas mais rápidas e de maior valor ao cliente podem ser alcançadas utilizando o modelo sobreposto. Esta abordagem é muito similar ao que, a partir de 2001, passamos a chamar de métodos ágeis.

1.2. Objetivo

Este trabalho tem o objetivo de identificar e documentar, em forma de padrões, boas práticas para a implantação de métodos ágeis a fim de servir como guia facilitador a todos que desejam iniciar com métodos ágeis em suas empresas ou equipes, pois está aqui documentado o conhecimento tácito de muitos que fizeram com sucesso esta implantação.

1.3. Abordagem de Solução

Para extrair as informações para o estudo foram realizadas entrevistas abertas com o apoio de um questionário semiestruturado. Por questionário semiestruturado entenda que se trata de um conjunto de perguntas aplicado ao entrevistado, onde as respostas e dados colhidos não são unicamente relativos as perguntas. Ou seja, se o entrevistado tem algo a acrescentar em relação ao tema e que não esteja dentre as perguntas definidas e é uma informação relevante, este dado também é registrado e considerado. Desta maneira pode-se obter o máximo de informação referente a experiência do entrevistado.

Os requisitos de seleção dos entrevistados são: líderes que implantaram Métodos Ágeis em algum projeto de desenvolvimento de software e que tenham obtido sucesso nesta implantação.

No primeiro momento, foi necessário selecionar pessoas ao qual seriam enviados os convites para as entrevistas. Estes nomes foram obtidos através de indicações feitas pelo orientador, o autor e por alguns dos entrevistados e através de pesquisa na internet.

Após a identificação de alguns nomes, foi enviado o convite para a entrevista. Diante dos retornos positivos obtidos, a entrevista foi agendada considerando a disponibilidade de data, horário e local de cada entrevistado.

Na entrevista é apresentando o objetivo da pesquisa e explanado que as questões existentes são voltadas a orientação da mesma, porém que não devem ser consideradas como limitação para as respostas. A duração das entrevistas variou de 45 minutos até 1h30, de acordo com a experiência de cada entrevistado. Foi armazenado o áudio de todas as entrevistas e estes seguem como apêndice ao trabalho no CD-ROM. Algumas entrevistas foram armazenadas em mais de um áudio por questões de interrupções que foram necessárias durante as mesmas, nestes casos, foi identificado no nome do áudio como “Parte1”, “Parte2” e assim por diante. Após a entrevista, o áudio foi ouvido repetidas vezes a fim de identificar os padrões.

1.4. Originalidade e Relevância

O termo “Implantar” pode ser definido como inserir algo em algum lugar. Também A seguir podemos observar a definição dada pelo Dicionário da Língua Portuguesa, Michaelis para o termo “Implantar” [40].

Implantar

im.plan.tar

(im1+plantar) vtd 1 Plantar (uma coisa) em outra; arraigar, fixar: Implantava na alma dos discípulos sadios ensinamentos cristãos. vtd 2 Estabelecer, introduzir: "Filosofia materialista integral que a Rússia tenta implantar" (Tristão de Ataíde, ap Franc. Fernandes). vtd 3 Hastear, arvorar: Implantou a flâmula da vitória. vpr 4 Arraigar-se, estar implantado: Implantara-se uma

parasita no tronco carcomido. vpr 5 Estabelecer-se, fixar-se: Grande leva de imigrantes ali se implantara.

Desta maneira, podemos dizer que a implantação de métodos ágeis trata-se da introdução de uma nova ideia. O livro *Fearless Change: Patterns to Introducing New Ideas* [2], é base fundamental para esta pesquisa. Muitos dos padrões contidos neste livro fazem parte da linguagem de padrões resultante desta pesquisa.

O trabalho de Daniel Cukier [1], tal como este trabalho, também é um derivativo do livro *Fearless Change: Patterns to Introducing New Ideas* [2], porém atua também de forma complementar ao adicionar e sugerir novos padrões para introduzir ideias especificamente na Indústria de Software. Alguns dos padrões contidos neste trabalho também, fazem parte da linguagem de padrões resultante desta pesquisa.

O livro *Agile Adoption Patterns* [12] aborda padrões referente a maioria das práticas mais comuns de métodos ágeis. O mesmo tem o foco principal no valor de negócio e exige que o leitor possua um conhecimento avançado sobre métodos ágeis.

Contudo, este trabalho diferencia-se das referências supracitadas por tratar especificamente de padrões para a implantação de métodos ágeis, onde os contextos são focados nos problemas técnicos e humanos que por, muitas vezes, não receberem a devida atenção, podem ser um dos responsáveis pelo fracasso na implantação de ágil.

1.5. Organização do Trabalho

Este trabalho está organizado da seguinte maneira: na Seção 2 é apresentado o referencial teórico, que introduz os principais temas necessários para uma leitura agradável e inteligível do restante do trabalho; na Seção 3 é apresentado como foi o processo de

levantamento de dados para a pesquisa e quais critérios foram utilizados para se encontrar um padrão; na Seção 4 é apresentado de maneira resumida, a linguagem de padrões resultante desta pesquisa e uma breve descrição de cada padrão; na Seção 5 são apresentados os padrões de forma completa; na Seção 6 é apresentada a Conclusão; na Seção 7 é apresentado as contribuições e na Seção 8 são apresentadas as sugestões de Trabalhos Futuros.

2. Fundamentação Teórica

Para a leitura deste trabalho, é importante ter bem claro a definição de 3 itens, são eles: gestão de projetos, métodos ágeis e padrões, nesta ordem.

2.1. Gestão de Projetos

O PM BoK [25] definiu um **projeto** como sendo um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. O termo “temporário” é utilizado, pois neste contexto, o projeto possui um início e término previamente definidos e o termo exclusivo é usado para salientar que por mais que haja atividades similares em projetos distintos, este só será um projeto se não for totalmente igual a outro.

O guia ainda definiu que **gerenciamento de projetos** é a aplicação de conhecimentos, habilidades, ferramentas e técnicas adequadas às atividades do projeto, a fim de atender as necessidades e expectativas de seus diversos *stakeholders* [25].

Na visão de Harold Kerzner, um **projeto** trata-se de um empreendimento com objetivo bem definido, que consome recursos e opera sobre pressão de prazos, custos e qualidade e que projetos são, em geral, considerados atividades exclusivas em uma empresa. Além disso, **gestão de projetos** é definida como o planejamento, a programação e o controle de uma série

de tarefas integradas de forma a atingir seu objetivo com êxito para benefício dos participantes do projeto [26].

Este trabalho opta pela definição do PM BoK em vez da de Kerzner. Pois ao se trabalhar com agilidade existe como premissa que se aplica habilidades e conhecimentos para desempenhar as tarefas, e que os esforços são norteados para atender as expectativas dos clientes. Em contrapartida a definição de Kerzner, onde nem sempre um projeto de software tem um objetivo bem definido logo no início, e se o tiver, trabalhando com métodos ágeis, este item se torna bem maleável, podendo mudar com o tempo.

2.2. Métodos Ágeis

O termo “Métodos Ágeis” surgiu em meados do ano de 2001, como resultado de uma reunião que ocorreu em Utah contando com 17 participantes que conheciam diferentes boas maneiras de desenvolver *software* [10].

O objetivo desta reunião, que durou três dias, foi entender o que estas diferentes boas maneiras de desenvolver *software* tinham em comum. Nesta reunião, foi criado o Manifesto Ágil para o Desenvolvimento de *Software*, que reúne quatro valores e doze princípios que todo método ágil compartilha.

Os quatro 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.

Nos valores do Manifesto Ágil, apesar de ter valor nos itens à direita, dá-se mais valor para os itens à esquerda (em negrito).

Os métodos ágeis surgiram como uma alternativa aos métodos tradicionais de desenvolvimento de software. Visam eliminar o excesso de documentações e reuniões ficando apenas com aquelas cujo se considera realmente necessárias aos projetos.

Na implantação de métodos ágeis, é necessário compreender, ou seja, “alcançar com a alma” [40] os princípios ágeis de desenvolvimento de software e não apenas aprendê-los, “reter na memória” [40].

Os princípios ágeis não tratam de técnicas ou outros métodos. Os princípios ágeis tratam fortemente, apesar de não unicamente, de temas humanos tais como a comunicação, confiança entre os envolvidos, responsabilidade e comprometimento de cada integrante do time.

Os padrões identificados nesta pesquisa abordam alguns problemas que comumente existem em equipes tradicionais, que dificultam a adoção de métodos ágeis, justamente por serem fatores que se contrapõem a alguns dos princípios.

2.3. Padrões

Padrões não são invenções de algo novo [1], é uma forma de organizar o conhecimento de experiências. O arquiteto Christopher Alexander definiu o padrão como uma regra de três partes que expressa a relação entre um certo contexto, um problema e uma solução [50]. Martin Fowler apresenta uma definição mais simples que diz que: um padrão é uma ideia que foi útil em algum contexto prático e provavelmente será útil em outros [9].

É sabido por muitos o valor da experiência em projetos. Quantas vezes não temos a sensação do *deja vu* – aquele sentimento que já resolveu aquele problema antes, mas não sabe

exatamente quando ou como. Se pudéssemos lembrar-nos da experiência e utilizar a solução adotada anteriormente, não precisaríamos redescobri-la. Este é o objetivo intrínseco aos padrões.

No livro *Agile Adotion* [12] é dito que “Há algo sobre padrões e livros. Eles andam juntos.”. Esta afirmação é feita, pois uma das formas mais comuns de documentar padrões é através de livros [12]. Porém também pode ser feito através de artigos, teses, monografias e similares, desde que estejam devidamente documentados.

Ainda é importante lembrar que, o fato de ser aplicado um padrão sobre um problema recorrente, não é garantia de sucesso [1], ele simplesmente garante que em muitos casos, determinado problema foi solucionado de determinada maneira.

2.3.1. Formatos de Padrões

A documentação de um padrão é uma tarefa que necessita basicamente da observação por parte do escritor. Este deve seguir em uma estrutura definida para fazer a documentação, denominamos esta estrutura de “formato do padrão”.

Um padrão pode ser escrito de muitas formas. Muitas destas formas incluem os seguintes componentes: nome, contexto, forças, problema, solução, padrões relacionados e resultados esperados [2].

A Figura 2 apresenta a tradução de um gráfico criado por Joshua Kerievsky [27] contendo quatro formatos existentes de se escrever padrões. Os formatos são nomeados como Portland, Coplien, GoF (*Group of Four*) e Alexandrino, e foram plotadas no gráfico de acordo com seu nível de maturidade e clareza.

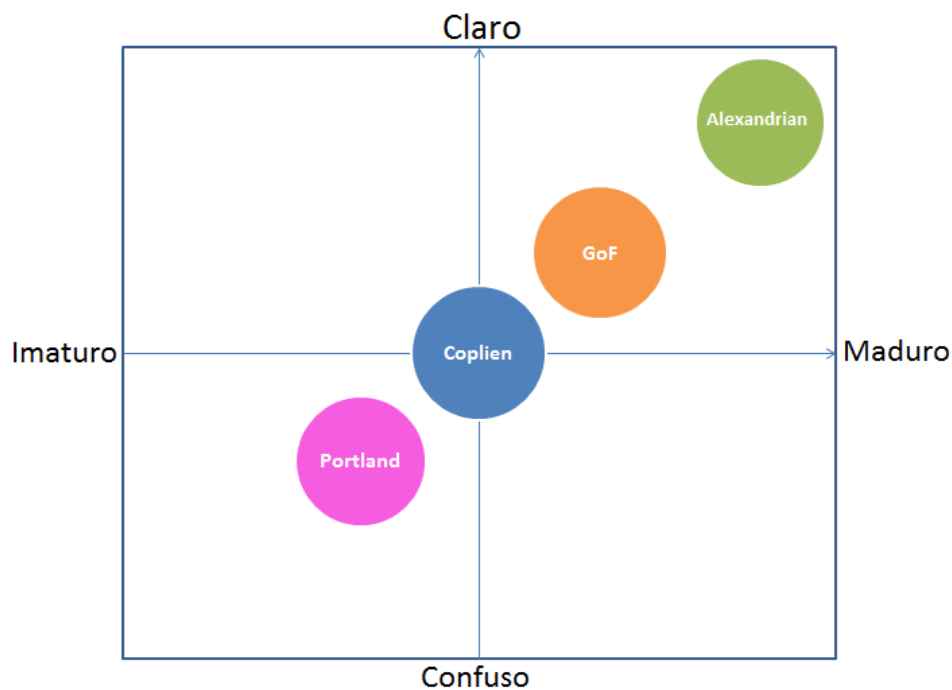


Figura 2 - Tradução do gráfico de Joshua Kerievsky.

É possível observar na Figura 2 que quanto mais para cima e mais a direita do gráfico, mais claro e maduro é o formato do padrão. Sendo assim, o mais claro e maduro é o formato Alexandrino seguido por Gof, Coplien e por último Portland.

2.3.2. Formato Alexandrino

O formato Alexandrino foi criado por Christopher Alexander, um arquiteto de construções que se utilizou de padrões para documentar práticas de sucessos na construção de edifícios e cidades.

Nesta pesquisa é utilizada uma variação do padrão Alexandrino. Por ser o formato mais claro e maduro de se escrever um padrão dentre os formatos citados na Figura 2.

A escolha pelo formato Alexandrino para a documentação dos padrões aqui documentados, também foi influenciada pelo modelo utilizado no livro *Fearless Change* [2], que também se utilizou de uma variação do mesmo, visto que ambos os trabalhos, este e o [2] tratam de implantar uma ideia.

2.3.3. Formato Adotado

Apesar de existir várias seções na documentação de um padrão, estas seções não precisam conter títulos que informem que elas iniciaram, pois a forma como o texto é apresentado já representa sobre qual parte do padrão se trata determinado excerto.

A seguir é apresentada a definição de como cada seção do padrão foi documentada. Cada quadrado possui: o nome da seção, a descrição do que ela propõe, a definição de como o texto será apresentada e um exemplo de acordo com a definição de apresentação informada.

Seção: Nome e Confiabilidade

Um nome para que o padrão seja rapidamente lembrado e sua confiabilidade de acordo com os dados da pesquisa.

Como irá se apresentar:

Em letras maiúsculas e em negrito. A confiabilidade será representada por asteriscos ao lado do nome, podemos ter de 1 a 3 asteriscos, sendo que quanto mais, maior a confiabilidade do padrão.

Exemplo:

SUBA AS ESCADAS **

Seção: Ilustração

Imagem para ilustrar a ideia do padrão.

Como irá se apresentar:

Uma única imagem após o nome com legenda que mostra a relação com o padrão.

Exemplo:

Figura 3 - Degraus para o sucesso.

Seção: História

Introduz a ideia do padrão.

Como irá se apresentar:

Em itálico.

Exemplo:

John acorda todos os dias às 7h. para iniciar sua rotina. Ao acordar, John toma banho, se trocar coloca terno e gravata, toma café da manhã e sai em direção ao escritório onde trabalha. Este trajeto leva em média 1h30 e John reclama todos os dias de quanto tempo ele perde até seu trabalho...

Seção: Resumo

Resume o problema, contexto e solução do padrão.

Como irá se apresentar:

Em negrito.

Exemplo:

Se você se sente desmotivado diariamente no seu trabalho, você deve buscar novos horizontes e oportunidades.

Seção: Contexto

Detalhes sobre o contexto com informações de experiências passadas, evidências, implantações, casos de uso, histórias e outros. Podem-se encontrar aqui relações com outros padrões.

Como irá se apresentar:

Em formato de texto de corpo.

Exemplo:

Você se sente mal e irritado ao acordar numa manhã de dia de trabalho e pensar que terá que ir trabalhar. Ao chegar ao escritório sente que não acontecerá nada de diferente e se lembra de vários itens que você não gosta no seu trabalho...

Seção: Problema

Detalhes do problema.

Como irá se apresentar:

Em negrito.

Exemplo:

Não é difícil perceber quando estamos insatisfeitos no trabalho atual, mas é difícil tomarmos uma atitude para mudar este sentimento.

Seção: Forças

São fatores que influenciam positiva ou negativamente o problema.

Como irá se apresentar:

Em formato de texto de corpo. Pode aparecer em tópicos.

Exemplo:

Mudar proporciona novas situações e experiências, que podem resultar em novos conhecimentos e aprendizados. As mudanças são naturalmente complexas, pois exige sair do conhecido e visitar novos campos.

Seção: Essência da solução (precedido de “Portanto”)

Em poucas palavras, descreve a solução.

Como irá se apresentar:

Em negrito.

Exemplo:

Portanto:

Aperfeiçoe seus conhecimentos e busque por novas oportunidades.

Seção “Mais sobre a solução”

Detalha a solução proposta.

Como irá se apresentar:

Em formato de texto de corpo.

Exemplo:

O primeiro passo é saber qual oportunidade vocês espera encontrar e definir um objetivo, após isso, defina onde serão pesquisados e realize esta busca...

Seção: Consequências

Após a aplicação da solução, quais os resultados, positivos ou negativos, esperados.

Como irá se apresentar:

Em formato de texto de corpo. Pode aparecer em tópicos.

Exemplo:

Simplemente por tomar a decisão de mudar a pessoa já ficará mais motivada e provavelmente passará a desempenhar um trabalho melhor mesmo no trabalho atual, ao qual não gosta. Isso ocorre, pois agora é possível pensar em um novo horizonte que mostra que não será sempre ruim como está.

Ao definir o perfil que se irá buscar a pessoa se torna mais segura, pois...

Seção: Usos conhecidos

Relaciona casos conhecidos onde foi aplicado o padrão.

Como irá se apresentar:

Em itálico.

Exemplo:

De acordo com pesquisas, o brasileiro tem cada vez mais trocado de emprego a se ver insatisfeito com o atual. Michele tem 28 anos e atualmente está na empresa XPTO...

Uma linha contendo a sequência de símbolos “◆ ◆ ◆” irá separar a seção **Resumo** da seção **Contexto**. Uma linha contendo a sequência de símbolos “= = = = =” irá separar

a seção **Mais sobre a solução** da seção **Consequências**. Padrões de outras fontes estarão com sublinhado e possuem suas referências na Seção **Referências**.

3. Pesquisa

Este trabalho trata-se de uma pesquisa descritiva e exploratória. Descritiva porque, este tipo de pesquisa visa observar, analisar, registrar e correlacionar os fatos ou fenômenos sem manipulá-los [23] e exploratória porque são abstraídas práticas recorrentes objetivando descrever sua natureza na forma de padrões [22].

3.1. Hipóteses

Existem diversas práticas comuns na implantação de métodos ágeis. Estas práticas podem ser reutilizadas por pessoas que também desejem realizar esta implantação em suas corporações ou equipes.

Algumas práticas estão presentes na maior parte das implantações. Essas práticas contém o maior nível de confiabilidade definido neste trabalho, que é o nível 3 identificado por 3 símbolos de “asterisco” (***). Essas práticas comumente presentes na maioria das implantações são DERRUBE AS BARREIRAS, ESCREVA NA PAREDE e COMECE ÁGIL NA GESTÃO.

3.2. Processo de Pesquisa

O processo desta pesquisa envolveu as fases de: identificação de padrões já existentes, coleta de dados em entrevistas e estudos de caso, análise dos dados e escrita dos padrões.

Durante a fase de identificação dos padrões existente, além de identifica-los, também foi possível observar as diversas formas de escrever um padrão. Nesta etapa foram encontrados os padrões utilizados nas referências [1], [2] e [12] além de outros citados em [27].

Na fase de coleta de dados tive contato com pessoas com abrangente experiência prática na implantação de métodos ágeis, possibilitando uma coleta ampla de dados que alimentaram a próxima fase. Além destas informações, foram pesquisadas também por estudos de caso onde a implantação de métodos ágeis foi observada, tentada ou efetivamente realizada.

As informações para a fase de análise de dados foram surgindo gradativamente, pois as entrevistas tinham datas diferentes. O primeiro contato foi com os estudos de caso, após foram coletadas as informações das entrevistas. Destas informações foi retiradas sete práticas. Durante a leitura dos estudos de caso e áudios das entrevistas, foi possível identificar temas que estavam presentes justamente nas implantações, estes temas foram selecionados para serem documentados como padrões.

Os temas identificados tratam basicamente de informações centralizadas, integrantes de equipes com dificuldades de comunicação devido a distâncias físicas, pessoas pouco colaborativas e débitos técnicos gerais.

A fase de escrita do padrão é iterativa e incremental. Iniciou-se após a leitura de alguns estudos de caso e foram sendo complementados conforme as informações das entrevistas eram coletadas e analisadas.

3.2.1. Identificação de Padrões Existentes

Para ser considerado um padrão, era necessário que o problema ocorresse em duas ou mais fontes da pesquisa, ou seja, nas entrevistas ou nos estudos de casos.

3.2.2. Coleta de Dados

Visto que objetivamos apresentar os padrões para implantar com sucesso os métodos ágeis, foram utilizadas duas formas de obter informações: através de entrevistas e análise de estudos de caso.

3.2.2.1. Entrevistas

Foram entrevistadas quatro pessoas, Fábio Akita (Locaweb e outras empresas), Guilherme Chapiewski (Globo.com e Yahoo!), Maurício Matsuda (Webgoal) e Daniel Cukier (Locaweb), ambos se enquadravam no perfil de já ter implantado métodos ágeis em algum projeto relacionado ao desenvolvimento de *software*. Vários outros profissionais foram contatados, no entanto não se mostraram dispostos a falar publicamente sobre a metodologia interna de suas empresas.

As entrevistas foram realizadas com um questionário previamente estruturado e com questões abertas, exigindo desta maneira respostas completas e não apenas sim ou não. Este tipo de questionário permitiu entender de forma ampla e contextualizada a experiência do entrevistado a cerca do tema.

As perguntas foram elaboradas e subdividas em grupos de acordo com seus objetivos de abordagem. As perguntas 1, 2 e 3 são questões sobre a Decisão de se Implantar Ágil. As perguntas 4, 5, 6, 7, 8 e 9 são sobre as ações executadas para a implantação. As perguntas 10,

11 e 12 são a respeito dos obstáculos encontrados na implantação e como eles foram superados. As perguntas 13, 14 e 15 são sobre o contexto antes da implantação de métodos ágeis. As questões 16 e 17 são sobre o contexto após a implantação de métodos ágeis.

A seguir são apresentadas as perguntas do questionário e seus propósitos para o trabalho.

Pergunta 01 - Em que momento foi decidido usar ágil?

Esta questão tem o objetivo de saber a empresa ou empresas onde foi aplicado métodos ágeis pelo entrevistado. Desta forma possibilitando a obtenção de informações sobre porte da companhia, cultura e demais informações similares que caracterizam o ambiente.

Pergunta 02 - Foi uma decisão *top/down* ou *bottom/up*?

Esta questão tem o objetivo de identificar se a decisão de implantar métodos ágeis veio por parte a alta gerência, meia gerência ou dos desenvolvedores.

Pergunta 03 - Foi utilizado ágil no desenvolvimento e na gestão?

Esta questão tem o objetivo de identificar quais métodos ágeis foram sendo adotados, e se eram métodos ágeis focados na gestão, técnica ou ambas.

Pergunta 04 - Quais ações/práticas foram adotadas para implantar ágil?

Esta questão tem o objetivo de entender quais as ações realizadas no decorrer da implantação dos métodos ágeis.

Pergunta 05 - Quais das práticas adotadas deram mais certo e por quê?

Esta questão tem o objetivo de entender quais as ações que o entrevistado considerou imprescindível para a implantação de métodos ágeis.

Pergunta 06 - Alguma deu errado/resultado não esperado?

Esta questão tem o objetivo de identificar alguma ação que não tenha dado certo e quais os contextos neste momento.

Pergunta 07 - Foram executadas numa ordem específica? Se sim, por que e qual ordem?

Quais as primeiras e quais as últimas?

Esta questão tem o objetivo de entender se o entrevistado realizou as ações de uma forma ordenada e consciente e se sim, quais os motivos que o levaram a executá-las nesta ordem e qual foi a ordem adotada.

Pergunta 08 - Foi necessária alguma adaptação física tais como: mover a equipe para uma sala a parte, trocar mesas, comprar quadros, post-its, etc.?

Esta questão tem o objetivo de identificar se foram necessárias adaptações físicas para implantar métodos ágeis e se sim, quais foram as necessidades, as mudanças e como era antes da mudança.

Pergunta 09 - O que foi crucial para o sucesso do projeto que foi aplicado ágil?

Esta questão tem o objetivo de identificar se houve algum ponto que foi considerado crucial para a implantação de métodos ágeis. Item tão importante ao ponto de forma que caso este não fosse verificado, poderia resultar no fracasso.

Pergunta 10 - Quais foram os obstáculos enfrentados e como foram resolvidos?

Esta questão tem o objetivo de identificar quais os obstáculos enfrentados pelos entrevistados, na implantação de métodos ágeis e como estes obstáculos foram superados.

Pergunta 11 - Como “venceu”/“convenceu” quem estava resistindo?

Esta questão tem o objetivo de entender como foram vencidas as pessoas que resistiam as mudanças e porque elas resistiam.

Pergunta 12 - Como convenceram os desenvolvedores e a gerência?

Esta questão tem o objetivo de entender como foram convencidos os desenvolvedores e gerência em particular para realizar a adoção de métodos ágeis.

Pergunta 13 - Qual o background da equipe em ágil?

Esta questão tem o objetivo de conhecer qual era o conhecimento prévio do time sobre métodos ágeis.

Pergunta 14 - Qual o ambiente da empresa antes da implantação (cultura, metodologias de gestão e hierarquia)?

Esta questão tem o objetivo de identificar qual o contexto existente na corporação ou time antes da implantação de métodos ágeis.

Pergunta 15 - Quantas pessoas compunham a equipe que foi implantada ágil?

Esta questão tem o objetivo de identificar a quantidade de integrantes do(s) time(s) onde foi implantado métodos ágeis.

Pergunta 16 - Além deste projeto, foi aplicado em outros? Quantos? Deram tão certo quanto este?

Esta questão tem o objetivo de entender se a implantação foi feita em apenas um projeto ou se houveram outros e quais os resultados obtidos em cada um deles.

Pergunta 17 - Após esta experiência, como a empresa guiou os demais projetos? Ainda usando ágil?

Esta questão tem o objetivo de entender se a empresa continuou utilizando métodos ágeis após a implantação ou se foi descontinuada, e se sim, por quê.

Pergunta 18 - Algo mais que você gostaria de acrescentar?

Esta questão tem o objetivo de colher mais informações sobre a experiência do entrevistado referente a métodos ágeis e a implantação dos mesmos.

3.2.2.2. Estudos de Caso

Visto que muitas pessoas já fizeram a implantação de métodos ágeis, foram analisados casos ao qual estavam documentados a fim de complementar os dados para a pesquisa. Estes estudos de caso, por tratarem de experiências reais, se adequam ao escopo da pesquisa.

Estes casos podem ser conferidos nas referências [3], [45], [46], [47], [48] e [49].

3.2.3. Análise dos Dados

Com base nos dados coletados, foi feita a análise para identificação de situações recorrentes em determinados contextos e quais as soluções adotadas a fim de identificar os candidatos a se tornarem padrões.

A análise consiste em ouvir o áudio das entrevistas e ler os estudos de casos repetidas vezes buscando por similaridades de problemas, identificando os contextos e soluções adotadas.

Durante a análise, foram surgindo algumas dúvidas sobre as entrevista ou referente aos estudos de caso. Nestes casos, as dúvidas foram enviadas por *e-mail* ao entrevistado ou autor, com o objetivo de ser sanada.

Os critérios para a identificação de um padrão, neste trabalho, são:

- Um problema recorrente, ou seja, que ocorra em mais de uma fonte dos dados da pesquisa, entrevistas ou estudos de caso.
- Ao menos uma execução prática da solução para determinado problema recorrente.

O processo de Análise dos Dados se prolonga até o final do processo de Escrita de Padrões, pois novas informações vão sendo encontradas e percebidas no decorrer da documentação. Por este motivo é comum encontrar a afirmação de que o processo de escrita de padrões é iterativo e incremental.

3.2.4. Escrita dos Padrões

O processo de escrita dos Padrões Para Implantar Métodos Ágeis teve início com as informações obtidas através dos estudos de caso, no entanto até este momento não havia sido documentado nada.

A documentação passou a emergir após a realização das primeiras três entrevistas realizadas com Daniel Cuckier (Locaweb), Guilherme Chapiewski (Globo.com e Yahoo!) e Maurício Matsuda (Webgoal).

Os padrões **Derrube as Barreiras**¹ e **Escreva na Parede**² foram os primeiros padrões identificados e documentados. Estes dois primeiros padrões resultaram no artigo “Padrões Para Implantar Métodos Ágeis” apresentado no Mini PLoP Brasil 2011 [51].

Após a criação do artigo, diversas mudanças foram realizadas nestes padrões, sendo a mais discrepante, a mudança no formato do padrão. Posteriormente foi optado por formatos mais maduros e claros do que o formato utilizado anteriormente.

A última entrevista, realizada com Fábio Akita (Locaweb e outras empresas), veio logo em seguida. O que já havia sido escrito foi aperfeiçoado diante das novas informações e foram criados outros padrões.

O processo de escrita destes padrões envolveu diversas revisões e iterações. A cada nova informação coletada, um padrão escrito anteriormente era revisto com o objetivo de poder aperfeiçoá-lo com a nova informação.

4. Linguagem de Padrões

Cada padrão representa uma entidade isolada [5]. A aplicação de apenas um dos padrões aqui descritos, não é suficiente para a implantação de métodos ágeis. A linguagem de padrões apresenta como cada padrão se relaciona uns com os outros, e é uma sugestão de sequência de execução.

Existem padrões mais abrangentes, de mesma abrangência e menos abrangentes. A abrangência denota que: padrões mais abrangentes contêm em si, padrões menos abrangentes e ao seu redor, padrões de mesma abrangência. Segue um exemplo tirado da própria linguagem de padrões sugerida aqui.

¹ Vide Seção 5.1.

² Vide Seção 5.2.

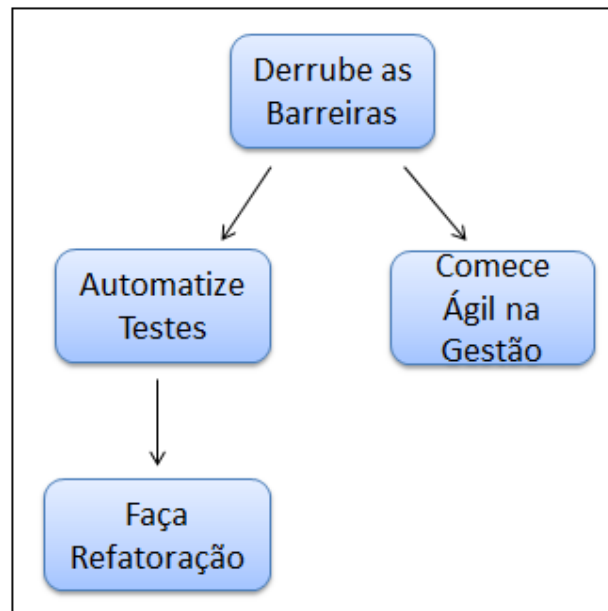


Figura 4 - Excerto da linguagem dos Padrões Para Implantar Métodos Ágeis.

O exemplo da Figura 4 demonstra que o **Derrubar as Barreiras** é mais abrangente que **Começar Ágil na Gestão** e **Automatizar Testes**, este último por sua vez, é mais abrangente que **Fazer Refatoração**. A aplicação de um padrão menos abrangente sugere que primeiro seja aplicado o padrão mais abrangente deste.

A linguagem apresenta os padrões de forma a demonstrar esta abrangência. No entanto, cada situação particular, poderá alterar, ao seu próprio critério, a sequência de execução dos padrões e quais padrões da linguagem serão usados, ficando com aqueles que mais se adequam ao contexto e descartando os demais.

4.1. Classificação dos Padrões

Na **Erro! Fonte de referência não encontrada.** a seguir, são apresentados os sete padrões divididos por três categorias: Comunicação, Conhecimento e Técnica. A seguir, segue uma breve explicação sobre cada categoria.

- **Comunicação:** são padrões que trabalham a comunicação do time e das pessoas interessadas no projeto.
- **Conhecimento:** são padrões que trabalham o conhecimento dos integrantes do time e das pessoas que influenciam nas decisões de como será feito o projeto.
- **Técnica:** são padrões que sugerem alguns métodos como soluções de problemas técnicos ou de escolha de metodologia para se começar a aplicação de métodos ágeis.

Na Figura 5 trazemos um mapa mental sobre os padrões apresentados, separados por suas devidas categorias. Cada padrão tem ao seu lado um número que representa a confiabilidade atribuída. Ainda, ao lado de cada padrão existem outros padrões em vermelho ou em azul.

Os em vermelho representam padrões que podem colaborar para a aplicação daquele padrão, estes padrões foram retirados do livro *Fearless Change* [2] e seus nomes foram colocados em português. Os padrões foram traduzidos por Daniel Cuckier em sua dissertação de mestrado [1].

Os em azul são padrões que já devem ter sido aplicado ou deve ser aplicado em paralelo, na maior parte das vezes, são padrões deste mesmo trabalho.

Tabela 2 - Resumo dos Padrões Para Implantar Métodos Ágeis.

TEMA	PADRÃO	RESUMO
Comunicação	Derrube as Barreiras	Visto que times ágeis precisam se comunicar constantemente é necessário derrubar as barreiras que interferem na comunicação.

Comunicação	Escreva na Parede	Para que as pessoas se sintam mais envolvidas e responsáveis pelo andamento do projeto, irradie as informações e permita que as pessoas participem da sua atualização.
Conhecimento	Monte um Canivete Suíço	Para realizar entregas mais rápidas, tenha pessoas no time que consigam realizar todos os processos de construção e testes do software.
Técnica	Comece Ágil na Gestão	Quando se decide implantar métodos ágeis, a primeira dúvida que surge é: “Por onde começar?”.
Técnica	Faça Testes	Para ter sempre um código com qualidade, dedique um tempo durante o desenvolvimento para a criação dos testes unitários, de integração e funcionais necessários.
Técnica	Faça Refatoração	À medida que se adiciona mais e mais linhas de código, o software vai ficando cada vez mais complexo e “sujo”. A refatoração serve justamente para deixar o código o mais limpo possível.
Técnica	Automatize Validações	Visto que em times ágeis os <i>commits</i> são feitos com muita frequência e por vários integrantes, automatize o processo de testes e building.

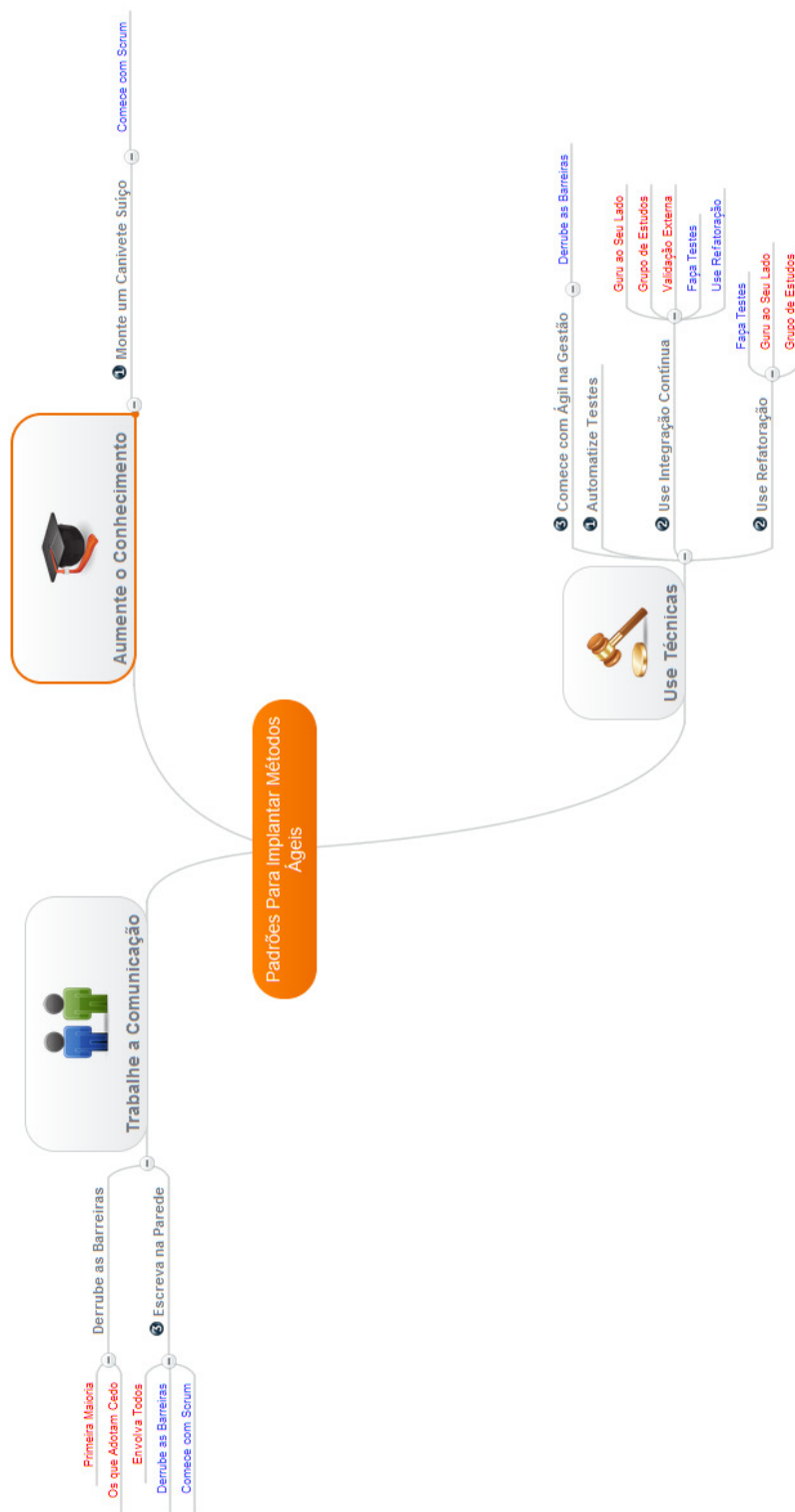


Figura 5 - Mapa Mental dos Padrões Para Implantar Métodos Ágeis

para a solução do problema. As setas azuladas, que passam de um padrão ao outro, indicam a relação de maior e menor abrangência, de forma que o padrão de origem da seta é mais abrangente que o padrão para o qual a seta aponta. Seguindo estas setas obtêm-se também algumas sequências para a execução dos padrões.

4.3. Padrões Relacionados

Os padrões relacionados a linguagem de padrões apresentada neste trabalho são os indicados em vermelho na Figura 5 e em quadrados vermelhos na Figura 6.. São eles: Primeira Maioria, Os que Adotam Cedo, Envolve Todos, Guru ao Seu Lado, Validação Externa, Grupo de Estudos [1][2].

Na Figura 6 foram apenas incluídos os que são mais abrangentes, os demais padrões relacionados servem de apoio mas não são mais abrangentes aos padrões ao qual apoiam.

Todos os padrões aqui documentados partem do princípio que o leitor é um Defensor Dedicado ou um Evangelista,

5. Padrões Para Implantar Métodos Ágeis

Neste capítulo são apresentados em detalhes os Padrões Para Implantar Métodos Ágeis. Os nomes dos padrões aparecem em negrito, caixa alta, alinhados a esquerda e numerados. Ao lado do nome do padrão, pode constar de um até três asteriscos que indicam o grau de confiabilidade atribuído o mesmo, conforme apresentado na Seção 2.3.3.

5.1. DERRUBE AS BARREIRAS***



Figura 7 – Queda do Muro de Berlim. A comunicação sem barreiras.

Para a coleta de dados e informações para este trabalho, foram realizadas entrevistas pessoalmente com os candidatos ao invés de utilizar questionários ou ferramentas de mensagens instantâneas, pois a comunicação frente-a-frente possibilita informações muito mais ricas. Durante uma das entrevistas o próprio entrevistado comentou que passar as mesmas informações da entrevista por e-mail, por exemplo, seria praticamente impossível. O mesmo ocorre no dia-a-dia dos projetos de desenvolvimento de software. Quanto mais próximas estão as pessoas, mais ricas e com menos ruídos será a comunicação.

Visto que times ágeis precisam se comunicar constantemente é necessário derrubar as barreiras que interferem na comunicação.



Você é um Evangelista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização. Você está buscando meios que tornem os integrantes do seu time mais unidos e companheiros uns com os outros.

Você quer que seu time se comunique mais e sejam mais empáticos uns com os outros.

As barreiras da comunicação podem ser: físicas, tal como mostram a Figura 8 e a Figura 9 com divisórias de mesas laterais e frontais, integrantes de times em salas diferentes, dentre outras; ou psicológicas: ambiente com muita pressão, prazos apertados que impedem que se tenha uma folga de tempo para ajudar o próximo e assim por diante.

A má comunicação é característica existente em muitas organizações e uma das principais causas de problemas em projetos [30]. Problemas de comunicação podem resultar em entregas em desacordo com os pedidos dos clientes ou sem valor ao negócio, problemas de relacionamento profissionais, maior *turnover* de pessoas da equipe, dentre outras.

Em organizações mais formais, tornar o ambiente mais propício para a comunicação pode ser menos trivial do que em organizações casuais. As próprias pessoas podem ser barreiras, pois podem não reagir positivamente ao ter que se comunicar mais.

É certo de que não é de um dia para o outro que a comunicação irá aparecer e proliferar em um time, mas é preciso motivá-la em algum momento para que isso passe a ocorrer, pode-se usar os padrões Early Adopter e Early Majority. Segundo a Experiência de *Hawthorne*, realizada pelo sociólogo Elton Mayo, concluiu que quanto maior a integração social no grupo maior será sua vontade de produzir [31].



Figura 8 - Barreiras físicas em forma de divisórias frontais.

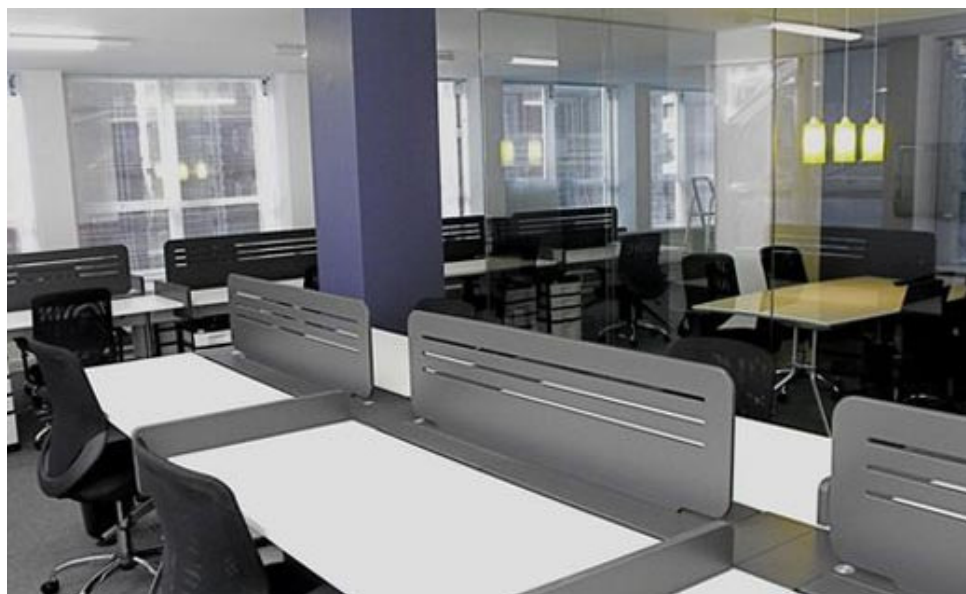


Figura 9 – Mesmo as divisórias menores interferem na comunicação e integração do time.

Portanto:

Remova as barreiras físicas ou psicológicas existentes, de forma a permitir uma melhor comunicação.

Uma importante barreira a ser derrubada é a da distancia. Ocorre quando integrantes de um mesmo time estão alocados fisicamente em salas diferentes. Neste caso, traga quem está distante para perto do restante do time.

Outras situações que podemos encontrar barreiras são as divisórias de baias, muito utilizada hoje nas organizações. Estas divisórias são por menos que pareçam, barreiras na comunicação livre dos indivíduos. Neste caso, retire as divisórias ou troque-as por uma grande bancada ou mesa, tal como apresentado na Figura 10 adiante.

Além dos aspectos físicos citados, temos os aspectos psicológicos. Em projetos com prazos muito apertados resta pouco tempo para as pessoas conversarem tanto informalmente quanto para se ajudarem com tarefas do trabalho, resultando em individualismo. Para amenizar a tensão existente neste tipo de ambiente, é necessário ter mais controle do que é necessário fazer e o tempo que se tem para executar. Além de ser flexível caso algo não possa ser concluído no tempo previamente estimado. Para isso **Comece Ágil na Gestão**³.

Ainda sobre aspectos psicológicos, pode-se definir uma Identidade ao Grupo. Esta pode originar do próprio nome do projeto. Esta identidade colabora na emersão do “espírito de equipe”.

³ Vide Seção 5.7.



Figura 10 - Escritório da empresa Jera sem divisórias e integrantes próximos.

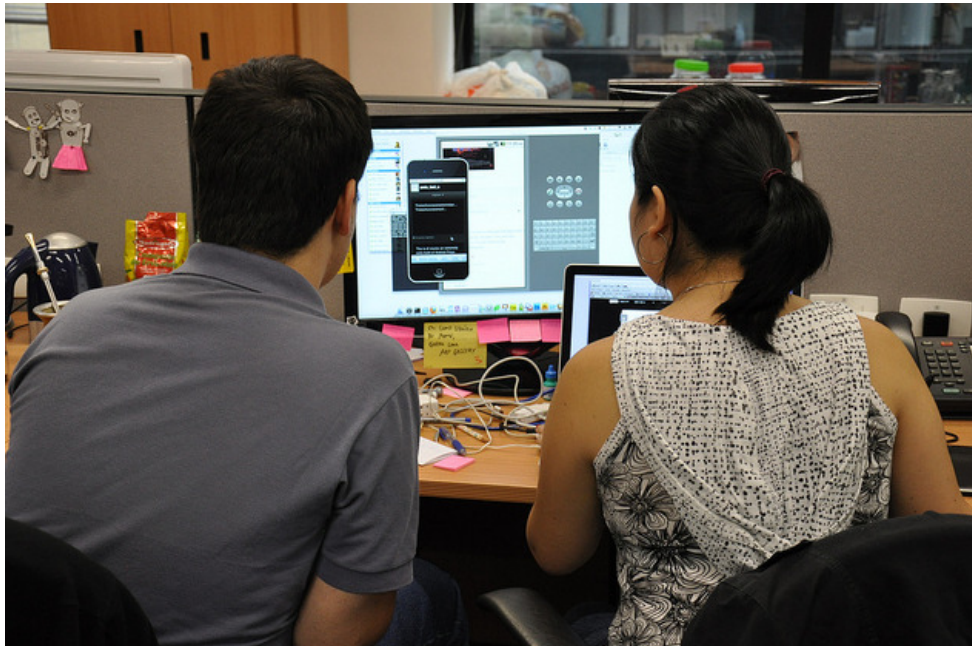


Figura 11 - Exemplo do Yahoo! ausência de divisórias laterais.

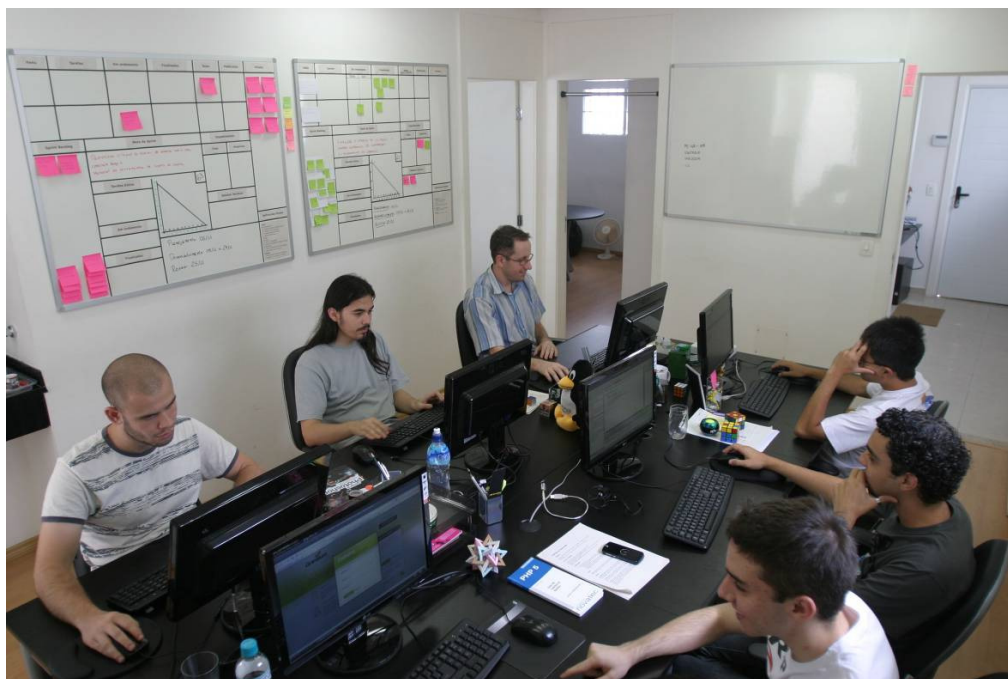


Figura 12 - Na Webgoal, ambiente sem divisórias e os integrantes próximos.

=====

Derrube as Barreiras propõe que toda a equipe fique fisicamente em um mesmo ambiente de forma a facilitar o acesso uns aos outros promovendo que os membros interajam e também que o ambiente seja amistoso.

Após a aplicação do padrão o ambiente tende a ser menos formal e mais amistoso e com menos pressões, ocorriam conversas sobre temas do trabalho ou temas aleatórios, aumentando a satisfação no trabalho.

O ambiente se torna mais colaborativo e o grupo passa a pensar junto em formas de serem mais produtivos e eficientes. Sentem-se mais a vontade e seguros para propor soluções criativas e não convencionais para os problemas. Aumenta a produtividade e o

comprometimento com o objetivo em comum, pois se criou a organização informal dentro do grupo. Torna-se mais fácil a resolução de dúvidas.

O ambiente por vezes pode ser se tornar mais barulhento do que anteriormente. Como as pessoas têm mais liberdade para falarem umas com as outras por vezes não tão próximas, tende a aumentar em alguns momentos, o volume da voz.

Podem ocorrer mais conversas do que o necessário ao projeto, o que por vezes incomodam algumas pessoas. O time pode torna-se mais disperso devido a intimidade gerada por compartilharem tarefas de trabalho, espaço físico e conversas relacionadas ou não ao projeto.

Na WebgGoal foram retiradas as divisórias das mesas de trabalho e com alguns ajustes, criou-se uma grande bancada. Em outro momento, a empresa passou a usar uma mesa , conforme mostra a Figura 12. Outros lugares utilizaram uma grande mesa também, parecidas com as de reuniões dentro de uma única sala para comportar todos os integrantes do time.

No Yahoo! as mesas de trabalho não possuem divisórias laterais, ver Figura 11, de forma que as pessoas possam se comunicar mais facilmente. Toda a equipe do projeto está disposta em um mesmo ambiente e próximo uns aos outros não havendo integrantes de outras equipes entre eles.

Na empresa Locaweb estas mudanças foram feitas conforme surgiam as necessidades. Uma destas necessidades foi a adaptação física para utilizar programação pareada: “Fizemos eles mudarem as mesas que as pessoas trabalhavam para poder fazer pareamento, isso é fundamental, fundamental!”.

5.2. ESCREVA NA PAREDE***



Figura 13 - Seja qual for a informação, coloque-as na parede.

Muito mais fácil gerenciarmos aquilo que podemos ver, ou aprender aquilo que vemos constantemente. Muito simples ir até determinado endereço se houverem placas que me dizem se estou no caminho certo. É este o conceito deste padrão. Disponibilizar a informação para todos, de forma a aprender mais, saber mais e validar mais. Conforme dito pelo Guilherme na entrevista "Tudo que é visualizado, para mim é mais efetivo..."

Para que as pessoas se sintam mais envolvidas e responsáveis pelo andamento do projeto, irradie as informações e estimule que as pessoas participem da sua atualização.



Você é um Evangelista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização. Você já aplicou **Derrube as Barreiras**⁴ e sente que as pessoas estão envolvidas entre si, mas não estão envolvidas com o projeto.

Você deseja equalizar a informação dentro do time, difundir processos internamente existentes e dispende menos tempo com atualização de cronogramas, gráficos, planilhas de atividades dentre outros.

As informações de um projeto são muitas e é importante gerenciá-las. O controle é uma das atividades mais importantes no gerenciamento de projetos [18]. Para isso, a forma tradicional de gerenciamento de projetos costuma utilizar documentos e *softwares* de gestão, onde uma pessoa denominada como líder do projeto, tem a responsabilidade de mantê-los atualizados.

A experiência no desenvolvimento de *software* tem mostrado que as mudanças de requisitos e escopo é a única certeza durante todo o processo. Os requisitos nunca param de mudar [20]. Isso torna a atualização destas informações muito trabalhosa de serem feitas por uma única pessoa que normalmente, ainda tem sob sua responsabilidade, muitas outras tarefas.

As falhas são mascaradas pelas informações contidas nas ferramentas [19], resultando em distorções que podem não ser percebidas a tempo para se tomar uma ação. Mas quando as informações são constantemente visualizadas, são também constantemente validadas, garantindo uma informação mais coesa com o real status do projeto.

⁴ Vide Seção 5.1.

Quando é necessário acessar arquivos ou páginas para visualizar o andamento, isso necessita de uma ação explícita dos membros do time. A informação estando na parede o acesso fica fácil e estimula a colaboração por parte dos integrantes do time em manter a informação coesa, desta forma eles se sentem responsáveis pelas informações exposta e decisões que são feitas no projeto.

O fato das informações estarem expostas facilita a visualização por parte de qualquer pessoa, inclusive mal intencionadas, fragilizando a questão da confidencialidade e segurança das informações. Mas embora isto ocorra dificilmente alguém que não compreende o processo utilizado pelo time, irá compreender informações mais relevantes sem alguma, mesmo que breve, explicação.

A Figura 14 e a Figura 15 são exemplos de ambientes não informativos.



Figura 14 - Ambiente não informativo e não interativo. Não há quadros nas paredes.



Figura 15 - Ambiente não informativo e não interativo. Não há quadros nas paredes.

Portanto:

Coloque as informações nas paredes e torne o ambiente mais informativo.

Existem muitas maneiras de colocar as informações na parede. **Envolva Todos**, pois a ideia é compartilhar informações e motivar a equipe. Fazendo com que cada membro se sinta responsável, atualizando, criando ou retirando quando não for mais necessário.

Nos casos onde é difícil saber os status das atividades, pode ser usado um Quadro de Tarefas. Neste quadro, crie três colunas: A fazer, Fazendo e Feito. Desta maneira, as atividades que estiverem na primeira coluna estão pendentes. As atividades indicadas na segunda coluna estão sendo feitas. E as atividades indicadas na terceira coluna já foram concluídas. Existem diferentes maneiras de disponibilizar e modificar este quadro. Quanto maior a maturidade da equipe em métodos ágeis, mais naturalmente isso irá emergir.

Outro caso que pode ocorrer é o de longas discussões sobre arquitetura e, desta maneira gerar muitos arquivos de forma a documentá-la. Disponibilize lousas para que possam ser utilizadas, possibilitando assim que as ideias sejam expressas visualmente. Deste modo, automaticamente a informação será documentada e disponível a todos, com pouco esforço.

Além disso, pode ocorrer a necessidade da disseminação de um conhecimento específico no time, um processo definido internamente, uma metodologia nova ou algo similar. Para isso, é necessário um impresso com as informações que se deseja disseminar e, que fique disponível na parede em um local onde as pessoas sempre olhem. Assim irá ocorrer a aprendizagem pela repetição, e em pouco tempo, o conhecimento estará disseminado.

As Figuras 17, 18 e 19 são exemplos de ambientes informativos.

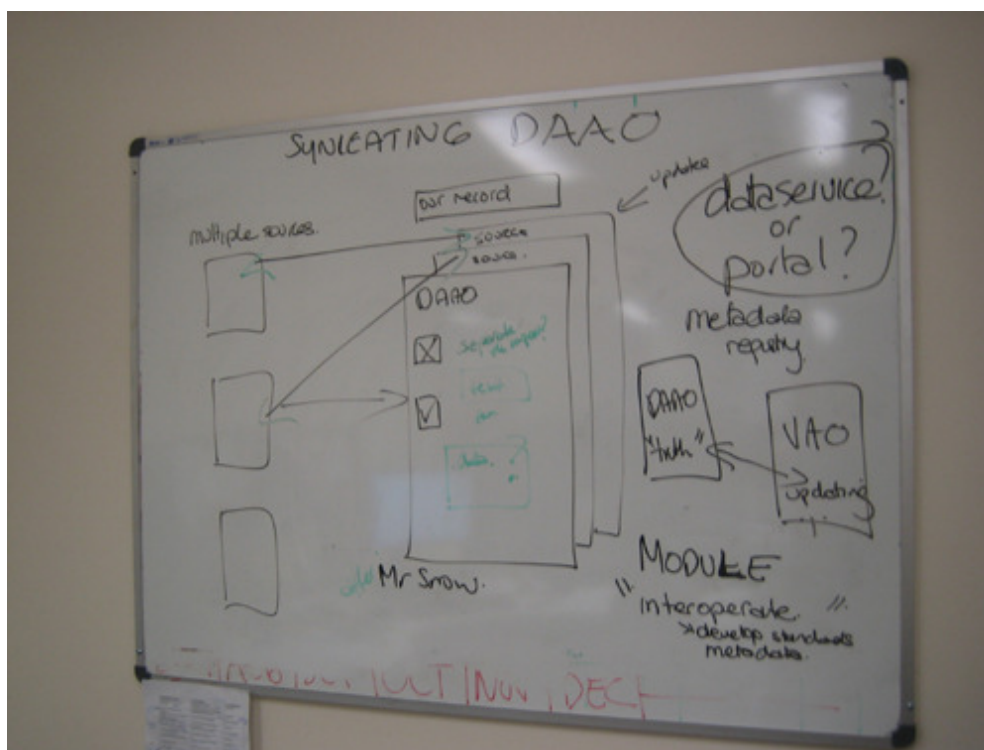


Figura 16 - Quadro expondo arquitetura de sistema.

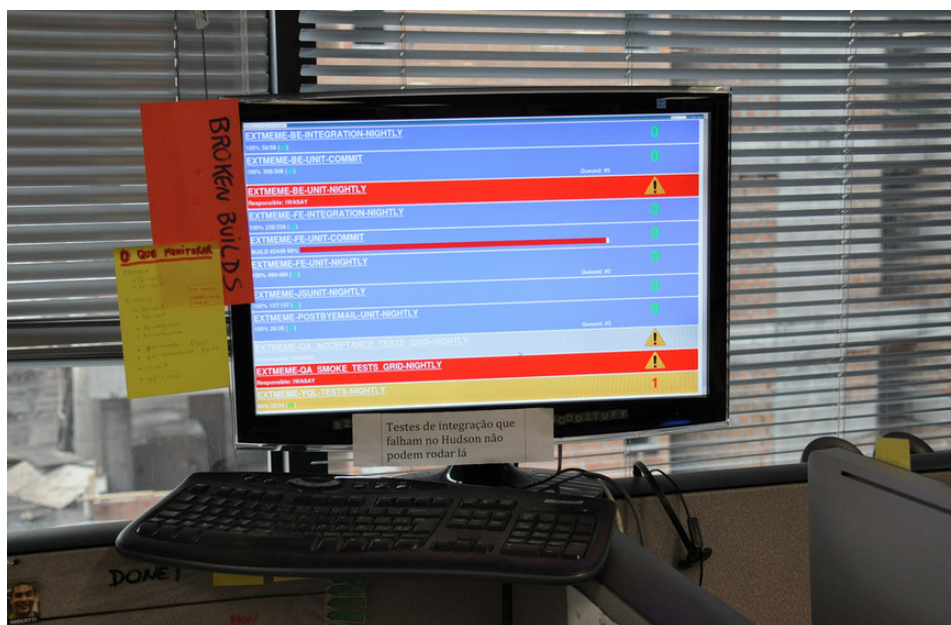


Figura 17 - Ambiente do Yahoo! com monitor visível para o time que mostra os resultados dos build's.



Figura 18 - Ambiente Yahoo!. Informativo e interativo.

=====

Escreva na Parede propõe que as informações do projeto fiquem expostas visualmente no ambiente de trabalho e que possam ser criadas novas ou atualizadas as já existente por qualquer integrante do time. Além disso, pode ser visualizada por todos os interessados de forma rápida e fácil.

Após a aplicação do padrão são esperadas mudanças físicas no ambiente e comportamental nas pessoas. As mudanças físicas no ambiente são facilmente perceptíveis, pois as informações passaram a ficar expostas nas paredes. As mudanças psicológicas são mais peculiares e notáveis com o passar do tempo, pois conforme as pessoas se sentem responsáveis pelas informações expostas, mais elas se sentem parte do projeto e seus resultados. Tornam-se mais comprometidas com os resultados e juntas passam a buscar formas de desempenhar as tarefas de formas melhores.

O ambiente se torna mais agradável e produtivo. A comunicação obtida depois que foram **Derrubadas as Barreiras**⁵ passa a incluir assuntos e temas relacionados ao projeto, fazendo com que as pessoas reflitam mais em o que está sendo feito e como, e problemas passam a ser identificados antecipadamente.

Passam a ocorrer menos reuniões longas com explicações sobre o andamento do projeto, visto que as informações para acompanhamento do andamento do projeto estão expostas e com pouco conhecimento ou explicação pode-se entender o status atual.

⁵ Vide Seção 5.1.

Para reduzir o número de bug's do projeto, foi criado um quadro onde cada post-it representava um. Todos sempre souberam que existiam alguns bug's, mas ao criar o quadro, o impacto de ver tantos post-it's foi tão grande que, os integrantes perceberam a gravidade dos problemas na saúde do projeto. Desta forma, o time se organizou para eliminar todos os post-it's. Agora todos tem como manter o quadro o mais vazio possível;

Monitore o trabalho constantemente sem invadir o espaço dos outros! Para manter os testes unitários saudáveis, e assim o software como um todo, no Yahoo! foi colocado o software de integração contínua exposto numa grande televisão. Quando algum teste fica inválido, imediatamente é apresentado no monitor um item em vermelho e o nome do desenvolvedor que fez a alteração no código. Desta maneira, a pessoa ao ver que invalidou o teste, imediatamente corrige e assim prosseguem com o trabalho sabendo que o código continua íntegro e válido.

Em times ágeis, os próprios integrantes discutem e definem o que será utilizado no projeto em relação a arquitetura, tecnologia dentre outras decisões corriqueiras que surgem em projetos de software. Para que a informação desenvolvida na discussão não seja perdida, são utilizados quadros brancos. Estes quadros servem para escrever e desenhar as ideias que surgirem nestas discussões. Ao se concluir uma arquitetura, por exemplo, a mesma é deixada no quadro, pois em caso de dúvidas, é só recorrer a ele, e se necessário, evoluir a ideia a partir daquele ponto [52].

Visualize o progresso das tarefas! Para visualizar e controlar o andamento de um grupo de tarefas ao longo de todo o fluxo pelo qual elas devem passar é comum que times ágeis utilizem um Quadro de Kanban, este quadro é bem similar ao Quadro de Tarefas. A complexidade ou não do quadro depende da maturidade em Métodos Ágeis e das

necessidades da equipe. “O objetivo é criar um bom fluxo através do sistema e minimizar o lead time... Comece simples e adicione mais colunas conforme o necessário” [15].

5.3. COMECE ÁGIL NA GESTÃO***



Figura 19 - Neném dando seus primeiros passos. Fonte:

<http://bebegravidéz.com/tag/primeiros-passos/>

Para alcançar qualquer meta, a primeira coisa que se precisa fazer é “Dar o primeiro passo”. Porém nem sempre é fácil identificarmos como faremos isso e por vezes, esta questão acaba por nos paralisar e desistimos sem nem ao menos ter começado. Da mesma forma, para introduzir métodos ágeis precisamos saber qual primeiro passo dar.

Quando se decide implantar métodos ágeis, a primeira dúvida que surge é: “Por onde começar?”.



Você é um Evangelista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização.

Você está convencido que métodos ágeis é uma opção melhor para fazer o desenvolvimento de software do que a forma como é feita hoje na sua organização e decidiu passar a usá-los, mas não sabe por onde começar.

Existem diversas metodologias ágeis para a gestão, uma delas é o Scrum que dentre os métodos ágeis é um dos mais conhecidos. O guia oficial do Scrum pode ser encontrado no sítio Scrum.org [66]. Mais informações sobre o método pode ser verificado em [11] e [67]. Outro método de gestão ágil muito falado na atualidade é o Lean, mais informações em [68].

Introduzir a gestão ágil em times que sempre trabalharam sem este conceito pode não ser um processo simples, pois irá exigir certa mudança de cultura na maneira de fazer as atividades, de como interagir com os colegas de trabalho e na forma de agir em geral. O cliente também terá que ter uma disponibilidade maior para sanar dúvidas e tomar decisões.

A forma de contratação de novos integrantes deve ser ajustada para incluir algumas características que são encontradas constantemente em pessoas que trabalham com métodos ágeis, tais como: pró-atividade, trabalho em equipe, conhecimento técnico, comunicativo, dentre outras. Ou ainda, pode-se incluir um treinamento sobre o assunto logo após a contratação.

Pode ocorrer uma resistência por parte da alta-gerência sobre o assunto, pois os projetos não terão um prazo e escopo fechados visto que, principalmente no princípio da implantação, normalmente não se sabe qual a velocidade do time para execução das tarefas.

Ao passar a utilizar um método ágil na gestão, será possível mostrar resultados em menos tempo, comparado aos métodos tradicionais, mesmo que não se entregue o software todo, será possível entregar funcionalidades, e isso deixa os líderes e clientes mais tranquilos e satisfeitos a respeito do projeto.

As funcionalidades entregues serão sempre as de maior valor para a corporação, de forma a eliminar o tempo gasto com funcionalidades que são pouco ou que nem são utilizadas, pois pode ocorrer de depois de implementadas algumas funcionalidades o ROI já seja atingido e a corporação decida de que não será mais necessário realizar as demais funcionalidades inicialmente previstas.

O escopo do projeto será mais maleável, visto que ele será feito e entregue por partes, possibilitando mudar de direção de acordo com novos objetivos e aumentar ou reduzir o escopo.

Portanto:

Introduza métodos ágeis começando pela gestão.

Quando se deseja implantar métodos ágeis, não se sabe muito sobre o assunto pode-se iniciar pela gestão. No caso de optar pelo uso do Scrum, o time pode seguir o guia contido no site para introduzir este método, desta forma, esta tarefa irá se tornar mais simples. No caso de optar por Lean existe o livro “A Mina de Ouro” que é o guia de estudos oficial [69]. Pode-se encontrar guias similares também para outros métodos.

Procure um Anjo Corporativo e encontre argumentos que alinhe os objetivos do método com os objetivos da empresa, após isso busque por um Patrocinador Local e sugira realizar uma Rodada de Testes.

Para que a ideia e conceitos sejam difundidos mais rapidamente dentro do time ou mesmo da corporação, crie um Grupo de Estudos sobre o tema.

Se ainda assim, sentir resistência por parte da alta-gerência sobre a implantação deste método, aumente a credibilidade da ideia trazendo uma Grande Personalidade.

Após a adoção de um método ágil para a gestão, pode-se partir para métodos ágeis voltados para a engenharia do software, codificação, qualidade, dentre outros. Pode-se por exemplo **Fazer Testes**⁶.

=====

O padrão **Comece Ágil na Gestão** propõe introduzir método ágil na corporação por meio de um método ágil voltado para a gestão de projetos. Desta forma, realizando entregas em períodos curtos de tempo, disponibilizando nestes períodos sempre as funcionalidades de maior valor ao negócio.

As funcionalidades implantadas passam a estar mais alinhadas com as expectativas dos clientes, visto que o mesmo poderá, na maior parte desses métodos, acompanhar o desenvolvimento delas durante todo o processo.

Após implantar ágil na gestão, é importante alinhar todo o restante do processo, principalmente desenvolvimento e testes, nos conceitos ágeis. Isso pode ser feito **Fazendo Testes e Fazendo Refatoração**⁷.

⁶ Vide Seção 5.5.

⁷ Vide Seção 5.7

Se não for, pode-se correr o risco de estar desenvolvendo muito código ruim, pois com a gestão ágil entende-se que as entregas serão mais rápidas. Se o desenvolvimento for feito de forma tradicional, o tempo da entrega pode não ser suficiente para que o código seja tecnicamente bom, nos forçando a adotar a primeira solução que pensamos. Dificilmente a primeira solução é a melhor e da forma tradicional, dificilmente algo já feito é alterado, sem que haja uma solicitação de negócio que exija determinada alteração. Então o código ruim ficaria lá por muito tempo.

TODO: colocar usos conhecidos.

5.4. MONTE UM CANIVETE SUÍÇO*



Figura 20 - Canivete Suíço representa um time multifuncional.

É comum vermos empresas departamentalizadas por funções e processos, o que faz com que os times de desenvolvimento de software sejam compostos, em sua maioria, basicamente por desenvolvedores e/ou analistas de sistemas, e as demais etapas de criação do software é feita por outros departamentos especialistas em cada etapa: desenho de interface, elaboração da

arquitetura, desenvolvimento das funcionalidades, testes, dentre outras. O problema deste formato é que o excesso de etapas/departamentos especializados em cada uma delas expande o lead time de desenvolvimento de software [17].

Para realizar entregas mais rápidas, tenha pessoas no time que consigam realizar todos os processos de construção e testes do software.



Você é um Evangelista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização. Para a realização do projeto, são envolvidos diversos outros departamentos, o que acaba por estender o *lead time* do projeto. Uma opção é, em paralelo a este padrão, **Começar Ágil na Gestão⁸**.

Você deseja incorporar em sua equipe, todas as fases, do desenvolvimento até a entrega, da criação do projeto para fazer entregas de software funcionando mais rápidas.

Em corporações maiores, onde a estrutura organizacional já se vê bem estabelecida e forte, pode ser uma tarefa mais difícil à aplicação do padrão. Já em empresas de menor porte, que costuma ser mais flexível, esta tarefa pode se tornar mais simples.

⁸ Vide Seção 5.7.

Há também a questão de que podem ser necessários integrantes “especialistas-generalistas”, ou seja, profissionais que são especialistas em um assunto, mas que conhecem com menor intensidade outros assuntos e são abertos para novos conhecimentos. Profissionais especialistas podem ser mais caros e ter um para cada equipe de desenvolvimento pode ser um custo que a empresa não considera favorável.

Portanto:

Envolva Todos em diversos tipos de tarefas treinando-as e fazendo com que adquiram conhecimento em diversas etapas do projeto.

O desenvolvimento de um projeto de software envolve diversas etapas, tais como: estrutura da informação, desenho de interface, estudo de navegabilidade, construção de arquitetura, desenvolvimento das funcionalidades, dentre outros. Se o caso é que algumas destas tarefas estão fora do escopo do time, traga-a para dentro do escopo do time, e treine o time para executá-las.

Por exemplo, se o layout do site for definido por uma equipe de marketing, é interessante ter no time ao menos um membro com habilidades de designer. Desta mesma maneira pode ser feita uma analogia a outras áreas, como no caso de uma área de testes, ser necessário ter ao menos um membro com habilidades de *tester*. No caso de uma área de arquitetura, é interessante ter no time alguém com maior conhecimento em arquitetura.

Em todos os casos, é necessário que estes profissionais tenham também outras habilidades para auxiliar nas demais tarefas para o desenvolvimento do projeto.

O ideal é que se chegue o mais próximo possível da ideia de departamentalização por produto [37], de forma que os conhecimentos do time sejam suficientes para a execução do

projeto. É importante salientar que o conhecimento deve conter no time, e não em algum integrante em particular. A passagem de conhecimento de um integrante ao outro, tornando todos aptos a realização de praticamente qualquer tarefa, é fundamental para a aplicação deste padrão. Essa passagem de conhecimento pode ser alcançada com um Grupo de Estudos ou com Pair Programming.

=====

Tenha um Canivete Suíço propõe que os integrantes da equipe detenham todo o conhecimento necessário para executar todas as etapas de criação do software.

Os integrantes do time adquirem novas habilidades visto que ocorre a transferência de conhecimento, podendo resultar em maior motivação geral da equipe.

Após a aplicação do padrão, o desenvolvimento do projeto tende a ser mais dinâmico, pois os problemas e dúvidas passam a ser resolvidos mais rapidamente sem a necessidade de esperar pela disponibilidade de pessoas de outras áreas, agendar reuniões para tratar dos assuntos, alinhar agendas, etc. Isso ocorre porque todos que são necessários ao projeto fazem parte do time e possuem as habilidades necessárias para executar as tarefas.

As pessoas com menos facilidade em adquirir novas habilidades ou que não são abertas a novos conhecimentos, acabam por ganhar destaque e pode ser identificadas como fora do perfil do time ou reaproveitadas em tarefas que exigem maior foco e dedicação.

Na empresa Yahoo! foi dito pelo Guilherme que os times são compostos por pessoas com habilidades para o desenvolvimento do produto, inclusive existe alguém que representa o cliente para retirar eventuais dúvidas de negócio.

Para transmitir as habilidades dentro da empresa, na Webgoal, quando não era possível que todos fizessem determinado curso, os que faziam repassavam o curso aos que não fizeram. Outra prática muito usada por eles é a o Pair Programming do XP com rotação de duplas e projetos, ou seja, as pessoas vão trocando de duplas e projetos, assim todos sabem sobre os projetos, interagem entre si e tem a oportunidade de adquirir novas habilidades uns com os outros.

Montar um canivete suíço significa que além das habilidades é necessário pessoas com o perfil certo, adequado a metodologia. Para tentar identificar pessoas com este perfil. A Webgoal na contratação realizava perguntas sobre o que a pessoa gosta o que faz no tempo livre, e similar e após a entrevista, convida a pessoa a passar uma tarde na empresa, para o candidato também poder avaliar se ele gosta do ambiente ou não.

No Yahoo! a decisão de contratação era feita com em consenso e participação ativa de todo o time. Após a entrevista, onde todos os integrantes participavam, o time conversa e decide se a pessoa deve ou não ser contratada. O importante neste momento é o líder não usar de sua posição para sobrepor a decisão do time, para que esta reunião não perca a credibilidade perante a equipe.

5.5. FAÇA TESTES**



Figura 21 – Testes são usados para a validação de conteúdo. Fonte:

<http://www.sempretops.com/wp-content/uploads/enem12.jpg>

Quando os testes são manuais e precisam ser repetidos com frequência muitas vezes eles acabam não sendo feitos por questões de prazo. Com a automação, consegue-se executar eles com uma frequência maior, com mais precisão e de forma mais ágil.

Para poder saber sempre se seu código continua funcionando, dedique um tempo durante o desenvolvimento para a automação dos testes.



Você é um Evangalista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização. Você está buscando por soluções para reduzir os bugs gerados após uma série de *commits* realizados durante o dia pelos integrantes do time.

Você quer garantir mais qualidade ao código e ter mais confiança para realizar alterações ou manutenções.

TDD do inglês *Test Driven Development*, ou Desenvolvimento Orientado a Testes é uma prática do método ágil XP (*eXtreme Programming* – Programação Extrema) que tem como um de seus objetivos antecipar a identificação e correção de falhas durante o desenvolvimento [33]. Os investimentos para passar a utilizar esta prática não costumam ser altos, porém o começo pode ser trabalhoso, pois a princípio normalmente não existe nenhum teste.

A forma como tradicionalmente os desenvolvedores validam o código que foi desenvolvido através de testes manuais e pontuais que verificam se as funcionalidades estão funcionando conforme esperado [55]. Denominamos este ato de depuração do código. Um dos problemas desta forma de validação das funcionalidades é que estes testes não ficam para a posterioridade e se qualquer alteração for realizada, este processo deverá ser realizado todo novamente [4].

Introduzir o conceito de Testes em um time que sempre trabalhou sem este conceito pode não ser um processo simples, pois irá exigir certa mudança de cultura para realizar a criação de testes de forma correta. A forma de contratação de novos integrantes deve ser ajustada para incluir este conhecimento.

Pode ocorrer uma resistência por parte da alta-gerência sobre o assunto, pois parte do tempo de desenvolvimento será utilizado para a criação dos testes, que não são funcionalidades a serem entregues ao cliente.

No entanto, de acordo com pesquisas, as falhas de software são tão comuns e danosas que se estima que causem um prejuízo anual de mais de 60 bilhões de dólares. E embora se

saiba que não é possível remover todos os erros, mais de um terço deste prejuízo poderia ser eliminado se fosse utilizado uma infraestrutura de testes melhor, permitindo identificar e remover falhas mais cedo e de forma eficaz [33].

Portanto:

Crie testes para validar o código do sistema.

O Existem no mercado frameworks que auxiliam na criação de testes de código. Normalmente não há custo na utilização deles, no entanto isso pode variar de acordo com a tecnologia adotada.

Na Tabela 3 a seguir, são listados alguns exemplos de frameworks e ferramentas para se trabalhar com a criação de testes e também automatização da execução dos mesmos. Uma tabela com listagem mais abrangente pode ser conferida em [53].

Tabela 3 - Lista de frameworks de criação de testes de código.

LINGUAGEM	FRAMEWORK	INFORMAÇÕES
Java	JUnit	Framework para testes unitários [55][56].
	Emma	Ferramenta de cobertura de testes [57].
	JMock	Biblioteca para a criação de Mocks [58].
	DBUnit	Extensão do JUnit para BD [59].
	Fit	Ferramenta para automatizar testes de aceitação [60].
.Net	csUnit	Framework para testes unitários [61].
PHP	PHPUnit	Framework para testes unitários [62].

Python	PyUnit	Framework para testes unitários [63].
Ruby	Test::Unit	Framework para testes unitários [64].

Em muitos casos, times oriundos de métodos tradicionais, podem não conhecer nada sobre teste de código, neste caso você pode sugerir um Grupo de Estudos, desta forma, torna-se mais fácil disseminar o conhecimento.

Se sentir resistência por parte da alta-gerência sobre a implantação desta prática, você pode aumentar a credibilidade da ideia trazendo uma Validação Externa ou mesmo apresentando os resultados de pesquisas que falam sobre as consequências, principalmente financeiras, decorrentes das falhas de software tais como o caso da Sony. “Estima-se que o prejuízo da Sony com a falha da PSN (*PlayStation Network*) ocorrida em 2011, que a deixou fora do ar por mais de uma semana, foi superior a 24 bilhões de dólares” [34] e o caso da Apple que saiu na MacWorld: “Saia justa: Apple libera pacote de correções que dá pau no sistema” [54].

=====

O padrão **Automatize Testes** propõe que para toda funcionalidade existente no sistema existam testes que garantam sua integridade. Desta forma, trazendo mais qualidade nos códigos desenvolvidos e segurança para realizar manutenções e novas implementações.

É possível fazer modificações sem medo de invalidar algo que já existia, pois se isso ocorrer estará invalidando algum outro teste e será necessário corrigi-lo. Após a aplicação

deste padrão, pode-se querer utilizar o padrão **Automatize Validações**⁹ para automatizar a execução dos testes.

A probabilidade de gerar erros em produção após novas implementações será reduzido e caso ocorram, será muito mais fácil de serem identificados.

Os desenvolvedores tendem a ficar mais seguros ao realizar manutenções ou criarem novas funcionalidades e isso irá resultar numa segurança maior em todo o time e clientes.

O tempo para desenvolver uma funcionalidade utilizando TDD e o tempo para desenvolver a mesma funcionalidade sem a utilização de TDD pode ser diferente, sendo que no primeiro caso, pode ser que o tempo seja superior, visto que existe um desenvolvimento a mais a ser feito, o dos testes. O ganho irá ocorrer na manutenção geral do software, pois inevitavelmente, com a utilização do TDD, o tempo de manutenção tende a ser menor, gerando também um custo menor.

O gráfico da Figura 22 apresenta o aumento do custo da mudança de acordo com o tempo de desenvolvimento, comparando duas formas de desenvolver software: tradicional, representada pela linha azul, e ágil usando TDD, representado pela linha vermelha.

⁹ Vide Seção 5.4.

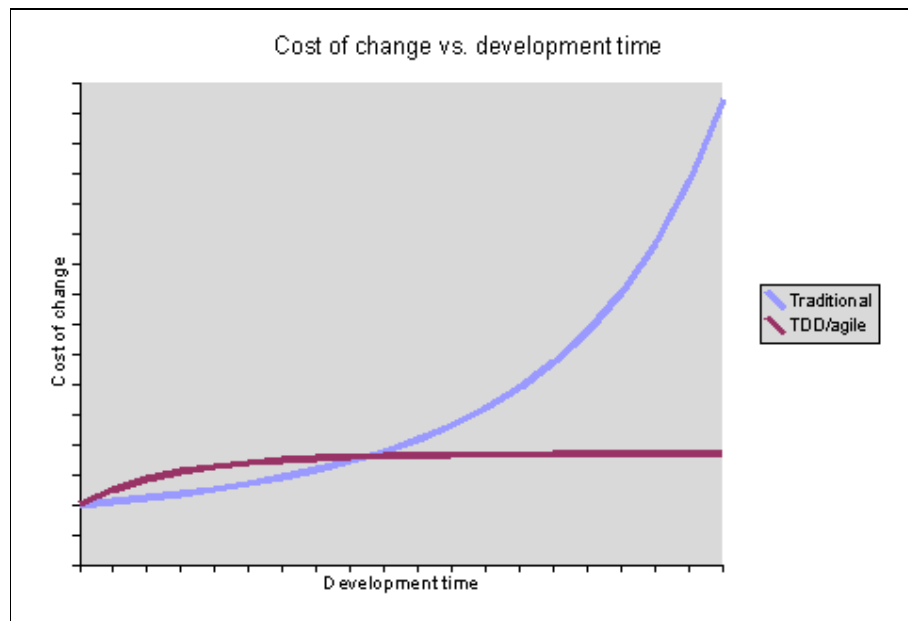


Figura 22 - Custo da Mudança vs. Tempo de Desenvolvimento [65].

De acordo com o gráfico visto na Figura 22, ao longo do desenvolvimento, com TDD o custo da mudança sofre um aumento não muito alto e após permanece estável, mesmo havendo mais mudanças. Já da forma tradicional, as primeiras mudanças também geram um aumento não muito alto, porém a partir de determinado momento, o valor do custo da mudança cresce de forma exponencial.

*A utilização de testes pode ser observada na maior parte das entrevistas e casos analisados. Podemos dizer que após **Começar com a Gestão Ágil** esta é a próxima prática que ocorre.*

Nos casos onde não havia nenhum padrão para a validação de código feito, ocorreu num primeiro momento, a adoção do hábito de criar testes. O hábito de criar testes é o primeiro passo para a utilização de testes automatizados em uma equipe. Nesta fase, os testes

são criados ainda manualmente podendo se utilizar de algumas ferramentas que auxiliam nesta tarefa. A evolução aqui, se dá na formalização do teste e na reutilização dos mesmos, ou seja, aquele teste poderá ser executado quantas vezes forem necessárias. O mais comum no princípio é a criação de testes unitários e com o tempo isso ir evoluindo para testes de integração, funcionais, interface e qualquer outro. Caso exista código legado, esta fase inicial é mais trabalhosa devido ao fato de ter que criar os testes para o código que já existe.

*O próximo passo para a adoção de testes é automatizar a execução dos mesmos. Depois de muitos testes criados, as novas modificações e implementações realizadas, ao serem “commitadas” é importante que sejam rodados todos os testes existentes, para garantir que nada foi impactado com a mudança feita. Se este processo for manual, será muito dispendioso. O ideal é que seja **Automatizado as Validações**.*

5.6. AUTOMATIZE VALIDAÇÕES**



Figura 23 – Velha dando bronca.

Ricardo tem 4 anos e sua mãe o viu fazendo algo errado e foi logo lhe dar uma bronca. A bronca nem sempre é algo ruim, pois pelo menos agora, o Ricardo sabe que fez algo errado e poderá corrigir. Automatizar validações é como ter uma mãe que sempre está de olho no que você faz para não lhe deixar fazer besteira.

Visto que em times ágeis os *commits* são feitos com muita frequência e por vários integrantes, elimine o tempo dispendido na validação manual automatize o processo de testes e *building*.



Você é um Evangalista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização. Você **Automatizou os Testes**¹⁰ e está buscando soluções que facilitem o processo de execução dos mesmos, *build* e *deploy*.

Você quer automatizar as tarefas que validam o trabalho feito pelo time e que precisam ser executados de forma repetitiva.

Um dos motivos para a aparição de muitos *bugs* durante o processo de homologação de um software que foi desenvolvido com métodos tradicionais, é que os testes realizados durante o desenvolvimento não contemplam todas as alterações realizadas no projeto. Isso

¹⁰ Vide Seção 5.6.

ocorre porque cada desenvolvedor, durante o desenvolvimento de suas atividades, realiza os testes locais. E apenas após todos os integrantes concluírem o desenvolvimento, é feito o *merge* dos códigos-fonte e implantado em um ambiente, para que o cliente possa testar. Em nenhum momento, antes dos testes pelo cliente, o código foi integralmente testado.

Introduzir o conceito de Integração Contínua em um time que sempre trabalhou da forma tradicional pode não ser um processo simples. Irá exigir a escolha de uma ferramenta de Integração Contínua e que todos os integrantes do time saibam como usá-la. Entretanto usar a ferramenta também não é o suficiente, a mesma deve ser usada de forma eficaz para que a implantação dê certo. A forma de contratação de novos integrantes deve ser ajustada para incluir este conhecimento, ou o treinamento necessário durante o período de adaptação.

Pode ocorrer uma resistência por parte da alta-gerência sobre o assunto, afinal até o momento estavam trabalhando sem esta metodologia.

Outro ponto importante a ser lembrado é a necessidade em infraestrutura tecnológica, onde pode requerer um servidor com o software instalado e disponível na rede. Este fato pode gerar um custo adicional para a empresa, caso a mesma não tenha equipamento disponível para manter o servidor de Integração Contínua.

Integração Contínua é um dos pilares da agilidade. A grande vantagem da Integração Contínua está no *feedback* instantâneo, que funciona da seguinte maneira: a cada *commit* no repositório, o *build* é feito automaticamente, com todos os testes sendo executados de forma automática e falhas sendo detectadas. Se algum *commit* não compilar ou invalidar qualquer um dos testes pré-existente, a equipe toma conhecimento instantaneamente (através de e-mail, por exemplo) indicando as falhas e o *commit* causador das mesmas. A equipe pode então corrigir o problema mais rápido. Reduzindo os riscos de introduzir erros ao criar novas funcionalidades ou refatorar [28].

Portanto:

Automatize tarefas de validação do software desenvolvido de forma a poderem ser executadas com frequência sem reduzir a velocidade do time.

Atualmente existe no mercado um conjunto heterogêneo de ferramentas de Integração Continua incluindo *Freeware*, *Open Source*, proprietárias ou mesmo pagas. Deste modo, a escolha deve considerar também o orçamento disponibilizado pela organização para implantar esta prática. Durante a escolha da ferramenta busque saber se há um Guru ao Seu Lado que conheça alguma ferramenta sobre o assunto, se houver, pode tornar mais simples o processo de escolha e treinamento.

Em muitos casos, times oriundos de métodos tradicionais, podem não conhecer nada sobre o assunto. Nesta situação, pode-se montar um Grupo de Estudos sobre o tema. Tornando assim mais fácil disseminar o conhecimento dentro do time.

Além disso, pode ocorrer a resistência por parte da alta-gerência sobre a implantação deste método. Neste caso é possível aumentar a credibilidade da ideia trazendo uma Validação Externa.

=====

O padrão **Automatize Testes** propõe automatizar a integração dos códigos-fonte e execução dos testes durante o desenvolvimento das funcionalidades. Desta forma, trazendo mais qualidade nos códigos desenvolvidos e segurança com relação às mudanças: você poderá

fazer modificações sem medo de invalidar algo que já existia, pois será avisado caso algo saia do esperado.

O risco de realizar mudanças após a implantação deste padrão será menor do que antes e o número de *bugs* encontrados nos testes pelo cliente também. A qualidade do código tende a ser melhor continuamente, ou seja, nunca será muito ruim porque a validação é automática e constante.

A produtividade da equipe tende a se torna maior, pois não será mais necessário dispendar tempo com tarefas como *build*, *deploy*, execução dos testes e demais atividades repetitivas inerentes ao processo de desenvolvimento de software.

Pode-se ir adotando a automação de forma gradativa, iniciando com tarefas mais simples como o *build* e aos poucos sendo inserido em processos mais complicados como testes funcionais ou migração de bancos.

Guilherme utilizou Integração continua na Globo.com e no Yahoo!. Ao entrar na equipe do Yahoo!, já era utilizado o software freeware Hudson. No caso do Guilherme ele notou que apesar de existir o software de Integração Continua, ele não estava sendo corretamente utilizado, pois o processo utilizado pelos desenvolvedores era desenvolver, commitar e apenas depois caso ocorressem problemas, criar os testes. Guilherme resolveu então colocar o resultado dos building em um televisor de forma que ficasse facilmente visível por toda a equipe. Após isso, a equipe pode perceber que existiam muitos testes invalidados, então mudaram o processo e começaram a realizar a maior quantidade de testes possíveis antes de commitar o código, evitando invalidar outros testes. Guilherme nos conta que a qualidade do código melhorou muito e que pela sua experiência: “Usar Continuous Integration direito, com tudo sendo “super” automatizado e “super” visualizado, resulta em um ganho de produtividade absurdo.”.

Na Globo.com os resultados dos testes eram também exibidos em um monitor posicionado de forma que fosse facilmente visualizado por todos do time, porém não era dado muita atenção quando um commit invalidava o build. Ao perceber isto, Guilherme criou um plugin para o software de Integração Continua ao qual a cada commit que invalidava o build era apresentado na tela uma mensagem: “Build xpto falhou. Foi o Fulano!”. Esta situação gerava um senso de competitividade e brincadeira entre a equipe, e resultou em mais qualidade, pois as pessoas passaram a fazer melhor seus testes unitários, para não invalidar nada.

5.7. FAÇA REFATORAÇÃO*



Figura 24 – Cozinha antes e depois da refatoração.

Se você escovar os dentes um pouquinho todo dia, terá os dentes saudáveis. Senão você em algum momento precisará de um tratamento de canal que é algo demorado e doloroso. Da mesma forma, sistemas tendem a se deteriorar ao longo do tempo, a medida que novas funções são inseridas, alterações são feitas, erro são corrigidos e mais código é introduzido [32].

À medida que se adiciona mais e mais linhas de código, o software pode ir ficando desnecessariamente complexo e com uma estrutura não adequada. Refatoração com frequência o código se torna mais fácil de trabalhar.



Você é um Evangalista ou um Defensor Dedicado e quer implantar métodos ágeis na sua organização. Você está **Automatizando os Testes**¹¹ e percebeu que muito código poderia ser melhorado. Notou também que algumas funcionalidades do sistema têm, gradativamente, estado com uma performance cada vez menor. O código fonte está cada vez mais difícil de entender e assim o time perde muito tempo com qualquer alteração simples.

O código existente fica cada vez mais difícil de ler e ruim de realizar qualquer alteração. Em consequência disso, o desempenho das funcionalidades do sistema tem decaído constantemente.

O objetivo da refatoração é melhorar alguma implementação de código sem alterar o funcionamento da funcionalidade do sistema.

É comum ouvirmos a frase de que “Será mais fácil reescrever todo o sistema do que alterar o código”. Isso ocorre porque com o passar do tempo, são adicionadas mais e mais

¹¹ Vide Seção 5.6.

linhas de código referente a algum novo requisito, correções de bugs de produção, e etc., gerando um amontoado de código. São muitos os motivos que nos fazem criar este amontoado, dentre eles podemos citar falta de comunicação entre o time que inviabiliza a alteração de algo que já foi criado, muitas dívidas técnicas, pessoal não qualificado, foco em processos onde se pensa menos na nomenclatura e clareza do código, complexidade técnica, pessoal desmotivado, dentre outros. Assim, dificilmente codificamos a solução ideal de primeira, mesmo porque, normalmente nem sabemos qual é.

Mas por que alterar o que já está funcionando? Há uma citação de Martin Fowler que diz que qualquer tolo consegue escrever código que um computador entenda, mas que bons programadores escrevem código que humanos entendam. É fácil escrever um código que compila, o difícil é fazê-lo expressivo o suficiente para que os outros membros do time consigam compreender também. O ideal é que o código fale a língua do leitor sem que ele tenha que forçar a mente para entender. Se for fácil entender, será fácil alterar [35].

Introduzir o conceito de Refatoração em um time que nunca o fez não é uma tarefa simples, pois será necessária uma mudança de cultura. Outro ponto a ressaltar é que pode ser mais difícil encontrar profissionais no mercado com este conhecimento.

Inicialmente a aplicação deste padrão pode ser mais trabalhosa, visto que não era feito refatoração antes. Mas com o tempo tende a ser cada vez mais fácil e dinâmico.

Portanto:

Ao identificar que alguma parte do código pode ser melhorada, refatore-a imediatamente.

Refatoração é um conjunto de técnicas para se melhorar o código. Como qualquer técnica, é necessário dispendir um tempo para aprendê-las. A maior parte dos times tradicionais não conhecem essas técnicas, para isso você pode sugerir um Grupo de Estudos sobre o assunto.

Algumas IDE's (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) já vêm com recursos que facilitam a aplicação das técnicas de refatoração, tornando mais fácil este trabalho.

A refatoração esporádica é trabalhosa e arriscada, pois se torna mais difícil o controle sobre o comportamento das funcionalidades devido a grande quantidade de código alterada de uma só vez. O ideal é que a refatoração seja constante. Quanto mais constante for a refatoração, mais serão reduzidos o trabalho, tempo e risco e o sistema estará constantemente legível e com bom desempenho.

=====

O padrão **Faça Refatoração** propõe manter o código sempre o mais limpo e legível possível mantendo ele simples de realizar alterações. Desta forma, possibilitando que as alterações sejam facilmente realizadas e os códigos já existentes sejam mais reaproveitados, pois como são entendidos facilmente, passa a ser preferível verificar se uma pequena alteração no que já existe atende aos novos requisitos ou soluciona algum bug do que criar um código totalmente novo para resolver.

Pode indiretamente, melhorar o desempenho do sistema visto que a qualquer momento pode-se optar por melhorar a lógica de uma função já existente, mudar o código para utilizar

uma nova ferramenta ou técnica que surgiu após a solicitação da funcionalidade que gerou a primeira implementação, etc.

Com o apoio dos padrões **Automatize Testes**¹² e **Automatize Validações**¹³, o risco de realizar as refatorações sem que se altere o funcionamento de determinada funcionalidade, é reduzido, pois os testes funcionais e de integração ao serem executados, irão garantir que a funcionalidade ainda façam o que deveriam fazer. A qualidade do código tende a ser melhor continuamente.

*Na maior parte dos casos estudados, percebeu-se que esta prática era adotada após **Fazer Testes**, pois o time passou a se sentir mais seguro para realizar refatorações de código visto que qualquer erro resultante da refatoração era provavelmente identificado com a execução dos testes.*

Não foi identificados casos onde a refatoração foi feita em ambientes que não tinham testes, e o risco de isso ser feito é alto, pois não há um validador formal de código, apenas os testes pontuais e manuais. Portanto, não é aconselhável refatorar antes de ter testes que validem o código.

Ao adotar a refatoração, o trabalho inicial pode ser mais doloroso caso exista código legal muito antigo. Mas com e realizando frequentemente a refatoração, o tempo e trabalho serão reduzidos e passam a ser natural para o time, pois se tornará um hábito.

O processo de refatorar é benéfico por trazer maior legibilidade, desempenho e outros, mas em contraposto, ele também é bem arriscado, por normalmente, não se tratar de

¹² Vide Seção 5.6.

¹³ Vide Seção 5.4.

um desenvolvimento de funcionalidade solicitado pelo cliente, sendo assim, não é esperado que o sistema haja de maneira estranha, resultante da refatoração. Para prevenir isso, muitas equipes começaram a refatorar após um treinamento sobre o assunto.

6. Conclusão

A competitividade das empresas tem crescido constantemente nos últimos anos. Apenas ter um produto ou serviço de qualidade e com baixo custo não são mais diferenciais, são características mínimas exigidas pelo consumidor. O único diferencial que nos resta é a inovação, e como já disse Waldez Ludwig: A inovação vem das pessoas [70][71][72].

Métodos ágeis sejam de gestão ou não, são focados principalmente em pessoas e no potencial que elas possuem. Foi na descoberta destas formas diferentes e melhores de desenvolver *software*, chamados de métodos ágeis, que passamos a entregar *softwares* mais rapidamente, de maior valor ao negócio e inovadores.

Contudo, por muitos anos guiamos os projetos de software de forma centralizada, controladora e preditiva com escopos extensos e sólidos. Denominamos estas como formas “tradicionais”. Talvez um dia estas formas realmente fossem as mais eficientes. No entanto, os dias atuais exigem flexibilidade de escopo, entregas mais rápidas e inovação, que é sabido ser características de métodos ágeis. O que não se sabe ao certo é quais passos seguir para realizar esta mudança de forma de trabalhar.

Para auxiliar nesta tarefa, elaboramos os sete Padrões Para Implantar Métodos Ágeis: **Derrube as Barreiras, Escreva na Parede, Comece Ágil na Gestão, Monte um Canivete Suíço, Faça Testes, Automatize Validações e Faça Refatoração**, com o objetivo de servir como mais uma fonte de informação que poderá auxiliar as pessoas que desejam implantar métodos ágeis em sua empresa ou time.

7. Contribuições

Documentamos neste estudo, detalhadamente, sete novos padrões que podem ser usados para a implantação de métodos ágeis com base em dados reais de pessoas que fizeram esta implantação com sucesso em algum projeto de software.

8. Trabalhos Futuros

Como trabalhos futuros sugeriram, temos as sugestões abaixo:

- A realização das entrevistas em outros ambientes, tais como instituições públicas, e em ambientes de outros países e culturas.
- A adição de novos padrões.
- A aplicação dos padrões de forma a validar os mesmos.

Referências

- [1] CUKIER, D. **Padrões para Introduzir Novas Ideias na Indústria de Software**. São Paulo: IME-USP, 2010. 150 p. Dissertação (Mestrado em Ciência da Computação), Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo.
- [2] RISING, L.; MANNS, M. L. **Fearless Change: Patterns for Introducing New Ideas**. Boston: Addison-Wesley, 2004. 273 p.
- [3] MELO, C. O.; FERREIRA, G. R. M. **Adoção de métodos ágeis em uma instituição pública de grande porte** Um estudo de caso. In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS (AGILE BRASIL), 14 p. 2010. Centro de Eventos da PUCRS (CEPUC), Porto Alegre. Disponível em:
http://www.agilcoop.org.br/files/WBMA_Melo_e_Ferreira.pdf
- [4] IMPROVE IT. **Improvecast 20 - Entrevista com José Papo na série Experiências Ágeis** (Parte 2). 2007. Disponível em: <<http://improveit.com.br/podcast/improvecast-20-entrevista-jose-papo-experiencias-ageis>>. Acesso em: 13 nov. 2011.
- [5] MELO, C. O. et al. **Um estudo exploratório dos fatores associados ao estímulo do aprendizado em times ágeis na indústria**. In: VII EXPERIMENTAL SOFTWARE ENGINEERING LATIN AMERICAN WORKSHOP, Goiânia, 2010.
- [6] FILHO, D. L. B. **Experiências com desenvolvimento ágil**. São Paulo: IME-USP, 2008. 170 p. Dissertação (Mestrado em Ciência da Computação), Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo.
- [7] SERIO, L. C. D.; DUARTE, A. L. C. M. **Competindo em tempo e flexibilidade – Casos de empresas brasileiras**. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 1999. 1 CD-ROM.
- [8] AMO, S. de. **Técnica de Mineração de Dados**. In: XXIV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, Salvador, 2004.
- [9] FOWLER, M. **Analysis Patterns: Reusable Object Models**. Addison-Wesley Professional, 1996.

- [10] BECK, K. et al. **Manifesto para Desenvolvimento Ágil de Software**, 2001. Disponível em: <<http://agilemanifesto.org/iso/ptbr/manifesto.html>>. Acesso em: 11 jul. 2011.
- [11] COHN, M. traduzido por BROD, C. **Uma Introdução ao SCRUM**. Disponível em: <http://www.mountangoatsoftware.com/system/hidden_asset/file/52/PortugueseScrum.pdf>. Acesso em: 12 jul. 2011.
- [12] ELSSAMADISY, A. **Agile Adoption Patterns: A Roadmap to Organizational Success**. Addison-Wesley Professional, 2008. 362p.
- [13] KAPLAN, R. S.; NORTON, D. P. **A Estratégia em Ação: Balanced Scorecard**. 4. ed. Rio de Janeiro: Campus, 1997. 360p.*
- [14] BRAGA, A. **A Gestão da Informação**. Portugal: Universidade da Beira Interior, 1996. Dissertação (Mestrado em Gestão). Disponível em: <http://www.ipv.pt/millennium/19_arq1.htm>. Acesso em: 26 ago. 2011.*
- [15] KNIBERG, H.; MATTIAS, S. **Kanban e Scrum: Obtendo o Melhor de Ambos**. C4Media, 2009. 139 p. Disponível em: <<http://www.infoq.com/br/minibooks/kanban-scrum-minibook>>. Acesso em: 27 ago. 2011.
- [16] Blog da Locaweb. **Kanban**, 2009. Disponível em <<http://blog.locaweb.com.br/metodologias-ageis/kanban>>. Acesso em: 27 ago. 2011.
- [17] VITOR, P. **Cálculo do Lead Time em User Stories**, 2010. Disponível em <<http://www.infoq.com/br/articles/lead-time-user-stories>>. Acesso em: 27 ago. 2011.
- [18] FILHO, A. C. N. **Uma Análise Comparativa de Softwares de Gestão de Desenvolvimento de Projetos: DOTProject e MS-Project**. Curitiba – PR: Faculdade Expoente, 2009. 60 p. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação).
- [19] SOUSA, C. **O uso de uma ferramenta facilita muito no dia a dia, mas não faz milagres**, 2011. Disponível em: <<http://projetoseti.com.br/governanca/pessoas-processos-e-ferramentas>>. Acesso em: 28 ago. 2011.
- [20] SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**, 2011. Disponível em <<http://www.scrum.org/scrumguides>>. Acesso em: 28 ago. 2011.

- [21] PRADA, R. **O que é Bug?**, 2008. Disponível em:
<<http://www.tecmundo.com.br/seguranca/213-o-que-e-bug-.htm>>. Acesso em 05 fev. 2012.

- [22] MALHEIROS, M. R. T. L. **O Processo de Pesquisa na Graduação**, 2004. 27 p.
Disponível em:
<http://www.profwillian.com/_diversos/download/prof/marciarita/Pesquisa_na_Graduacao.pdf>. Acesso em: 13 dez. 2011.

- [23] TOGATLIAN, M. A. **Tipos de Pesquisa**, 2001. Disponível em:
<http://www.togatlian.pro.br/docs/pos/unesa/tipos.pdf>. Acesso em: 13 dez. 2011.

- [24] SIMOES, R. **Fundamentos do Scrum** - 2a. Parte, 2011. Disponível em:
<<http://www.scrumrj.org/2011/03/11/fundamentos-do-scrum-2a-parte>>. Acesso em: 13 dez. 2011.

- [25] INSTITUTE, P. M. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)**. 4. ed. Estados Unidos da América: Project Management Institute, Inc, 2008. 337p.

- [26] KERZNER, H. **Advanced Project Management: Best Practices on Implementation**. 2. Ed. Hoboken, Nova Jersey: John Wiley & Sons, 2004. 864p.

- [27] KERIEVSKY J. **A Timeless Way of Communicating: Alexandrian Pattern Language**. In: PATTERN LANGUAGES OF PROGRAMS CONFERENCE, 2010. Estados Unidos da América. Disponível em:
<<http://www.slideshare.net/JoshuaKerievsky/a-timeless-way-of-communicating-alexandrian-pattern-languages>>. Acesso em: 09/01/2012.

- [28] GUERRA C. **Integração Contínua e o processo Agile**, 2008. Disponível em
<<http://blog.caelum.com.br/integracao-continua>>. Acesso em 11 jan. 2012.

- [29] INSTITUTE, P. M. **Estudo de Benchmarking em Gerenciamento de Projetos**, 2008, Project Management Institute - Chapters Brasileiros. Disponível em:
<http://www.techoje.com.br/bolttools_techoje/files/arquivos/benchpreliminar.pdf>. Acesso em: 11 jan. 2012.

- [30] QUARTOROLI, C.; MARTINS, L. C. S. **Gestão das Comunicações em Projetos de Tecnologia da Informação**. In: PM WORLD TODAY, 2010. Disponível em:
<http://www.pmies.org.br/clickadmin/midias/data/artigo-PMI_RIO.pdf>. Acesso em: 11 jan. 2012.

- [31] OLIVEIRA, C. A. **Teoria das Relações Humanas e sua Decorrência**. Disponível em:
<<http://www.professorcezar.adm.br/Textos/Teoria%20das%20relacoes%20humanas.pdf>>. Acesso em: 15 jan. 2012.
- [32] IMPROVE IT. **Refatoração**, 2008. Disponível em:
<<http://improveit.com.br/xp/praticas/refatoracao>>. Acesso em: 17 jan. 2012.
- [33] IMPROVE IT. **Desenvolvimento Orientado a Testes**. Disponível em:
<<http://improveit.com.br/xp/praticas/tdd>>. Acesso em 17 jan. 2012.
- [34] MAZETTO, L. **Prejuízo da Sony com falha da PSN pode ultrapassar os US\$ 24 bilhões**, 2011. IDG Now!. Disponível em:
<<http://idgnow.uol.com.br/seguranca/2011/04/27/prejuizo-da-sony-com-falha-da-psn-pode-ultrapassar-os-us-24-bilhoes>>. Acesso em: 17 jan. 2012.
- [35] OENNING, G. Refatoração: Mantendo uma Base de Código Limpa. **.NET Magazine** n.85. p.56-66.
- [36] IGOR, P. **O Ciclo ATDD + TDD: Test Driven Development Acceptance Test Driven Development**, In: SCRUM AMAZONIA, 2010. Disponível em:
<<http://www.slideshare.net/Pigor/palestra-tddcompleta-5460534>>. Acesso em: 19 jan. 2012.
- [37] OLIVEIRA, C. A. **Departamentalização**. Disponível em:
<<http://www.professorcezar.adm.br/Textos/Departamentalizacao.pdf>>. Acesso em: 23 jan. 2012.
- [38] GALLO, M.; LONGO, C. **A Influência dos Estilos de Liderança na Rotatividade de Pessoal: Um Estudo de Caso em uma Indústria de Produtos Alimentícios**. In: VII SIMPÓSIO DE EXCELÊNCIA EM GESTÃO E TECNOLOGIA, 2010. Rio de Janeiro. Disponível em: http://189.3.143.36/seget/artigos10/223_SEGET%202010.pdf. Acesso em: 05 fev. 2012.
- [39] **Commit**. Disponível em: <<http://pt.wikipedia.org/wiki/Commit>>. Acesso em: 05 fev. 2012.
- [40] MELHORAMENTOS, **Michaelis Moderno Dicionário da Língua Portuguesa**. Melhoramentos, 1998.

- [41] **Aquisição Programas.** Disponível em:
<<http://www2.ufpa.br/dicas/progra/proaqui.htm>>. Acesso em: 05 fev. 2012.
- [42] **Software Livre: Histórico, Definição e Importância.** Disponível em:
<<http://www2.ufpa.br/dicas/linux/li-lisol.htm#openso>>. Acesso em: 05 fev. 2012.
- [43] PRADA, R. **O que é Plugin?**, 2008. Disponível em:
<<http://www.tecmundo.com.br/hardware/210-o-que-e-plugin-.htm>>. Acesso em: 05 fev. 2012.
- [44] BASSI, G. **Rápido ou Ágil?** In: AGILE BRASIL 2010, 2010. Porto Alegre. Disponível em: <<http://www.agilebrazil.com/2010/material/Rapido%20ou%20agil-Giovanni%20Bassi.pdf>>. Acesso em: 05 fev. 2012.
- [45] SZABO, M. E.; SCHWEITZER, M. C. **Agilidade na UFABC** - Implantação da Metodologia Scrum na Divisão de Desenvolvimento. In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS - WBMA 2010, 2010, Porto Alegre, RS. Anais do Workshop Brasileiro de Métodos Ágeis.
- [46] CHALEGRE, V. C. et al. **Estudo de Caso da Adoção do Scrum no Desenvolvimento Distribuído de Software.** In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS - WBMA 2010, 2010, Porto Alegre, RS. Anais do Workshop Brasileiro de Métodos Ágeis.
- [47] SANTOS, A. C. C. dos. et al. **Realidade de uma Fábrica de Software utilizando Scrum no Desenvolvimento de Software.** In: WORKSHOP BRASILEIRO DE MÉTODOS ÁGEIS - WBMA 2010, 2010, Porto Alegre, RS. Anais do Workshop Brasileiro de Métodos Ágeis.
- [48] SOARES, F. S. F. et al. **Adoção de SCRUM em uma Fábrica de Desenvolvimento Distribuído de Software.** In: XXI SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE I WORKSHOP DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE, 2007. João Pessoa, PB.
- [49] MACHADO, M.; MEDINA, P. S. G. SCRUM - Método Ágil: uma mudança cultural na gestão de projetos de desenvolvimento de software. Revista **Intr@ciência**, v. 1, p. 58-71, 2009.
- [50] ALEXANDER, C. et al. **A Pattern Language:** Towns, Buildings, Construction. Oxford University Press New York, 1977.

- [51] CARVALHO, S. G.; GUERRA, E. M. **Padrões Para Implantar Métodos Ágeis**. In: MINIPLOP BRASIL 2011, 2011. São Paulo, SP.
- [52] AMBLER, S. **Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process**. Wiley, 2002. 400 p.
- [53] WIKIPEDIA. **List of Unit Testing Framework** Disponível em: <http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks>. Acesso em: 23 dez. 2012.
- [54] MACWORLD. **Saia Justa: Apple libera pacote de correção que dá pau no sistema**, 2012. MACWORLD / USA. Disponível em: <<http://macworldbrasil.uol.com.br/noticias/2012/02/03/saia-justa-apple-libera-correcao-da-correcao/>>. Acesso em: 07 fev. 2012.
- [55] BERNARDO, P. C.; KON, F. **A Importância dos Testes Automatizados**. Revista Engenharia de Software Magazine l(3) p 54-57, 2008. Disponível em: <<http://www3.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>>. Acesso em: 07 fev. 2012.
- [56] **JUnit.org** - Resources for Test Driven Development. Disponível em: <<http://www.junit.org>>. Acesso em: 07 fev. 2012.
- [57] IMPORVEIT. **Cobertura de Testes com EMMA**. Disponível em: <<http://improveit.com.br/xp/praticas/tdd/emma>>. Acesso em: 02 fev. 2012.
- [58] **jMock**. Disponível em: <<http://www.jmock.org/>>. Acesso em: 07 fev. 2012.
- [59] **DBUnit**. Disponível em: <<http://www.dbunit.org>>. Acesso em: 07 fev. 2012.
- [60] CORBUCCI, H. **Testes de Aceitação – Curso de Verão 2010**, 2010. Disponível em: <<http://ccsl.ime.usp.br/agilcoop/files/AgilCoop-Verao2010-Testes-08-Aceitacao.pdf>>. Acesso em: 07 fev. 2012.
- [61] **CSUnit**. Disponível em: <<http://www.csunit.org>>. Acesso em: 07 fev. 2012.
- [62] WEBER, A. **Testando suas classes com PHPUnit**, 2009. Disponível em: <<http://phpbrasil.com/artigo/76Gve1HwfXA4/1/testando-suas-classes-com-phpunit>>. Acesso em 07 fev. 2012.

- [63] **PyUnit** - The Standart Unit Testing Framework for Python. Disponível em: <<http://pyunit.sourceforge.net/>>. Acesso em: 07 fev. 2012.

- [64] **Test::Unit** – A Unit Testing Framework for Ruby. Disponível em: <<http://test-unit.rubyforge.org>>. Acesso em 07 fev. 2012.

- [65] LOPIS, N. **Backwards Is Forward: Making Better Games with Test-Driven Development**, 2006. Disponível em: <<http://gamesfromwithin.com/backwards-is-forward-making-better-games-with-test-driven-development>>. Acesso em: 07 fev. 2012.

- [66] **Scrum.org**. Disponível em: <www.scrum.org>. Acesso em: 23 dez.2011.

- [67] KNIBERG, H. **Scrum and XP from the Trenches**. Lulu.com, 2007. 140 p.

- [68] **Lean.org**. Disponível em: <http://www.lean.org.br/o_que_e.aspx>. Acesso em: 07 fev. 2012.

- [69] BALLÉ, F.; BALLÉ, M. **A Mina de Ouro: Uma Transformação Lean em Romance – Guia de Estudo**. Bookman. Disponível em: <http://www.lean.org.br/comunidade/guiaestudo/aminadeouro_guia.pdf>. Acesso em: 07 fev. 2012.

- [70] **Waldez Ludwig 1** no Programa sem Censura. Disponível em: <<http://www.youtube.com/watch?v=uyMY2uo-iqQ>>. Acesso em: 07 fev. 2012.

- [71] **Waldez Ludwig 2** no Programa sem Censura. Disponível em: <http://www.youtube.com/watch?v=_ixT4MSxjmE&feature=related>. Acesso em: 07 fev. 2012.

- [72] **Waldez Ludwig 3** no Programa sem Censura. Disponível em: <<http://www.youtube.com/watch?v=IjZjOamODFo&feature=related>>. Acesso em: 07 fev. 2012.

Glossário

Turnover: É um termo, do idioma inglês, utilizado para caracterizar o movimento de entrada e saída de profissionais empregados em um determinado período. O *turnover* é um dos principais problemas que atualmente vem preocupando os executivos e profissionais da área de recursos humanos das organizações. Reduzir ao máximo as saídas de pessoal tem sido uma tarefa quase que impossível, devido ao mercado de trabalho competitivo em que as empresas estão inseridas. O alto nível de rotatividade está relacionado à perda de produtividade, lucratividade e de saúde organizacional [38].

Bug: É quando a linguagem do computador entra em conflito e gera uma impossibilidade de continuar a execução de um programa. Também chamado de falha na lógica, ocorre quando o computador perde a finalidade de um determinado processo. Os *bugs* podem gerar falhas na segurança, especialmente quando ocorrem em programas que têm acesso à rede [21].

Post-it: São marcadores de papéis reposicionáveis.

Kanban: Metodologia de produção que utiliza controles visuais (cartões) como sinalizadores de estoque ou status de uma determinada etapa do processo [16].

Work In Progress: *Work In Progress* ou WIP (Trabalho em Processo) são os itens que estão dentro dos limites do processo, foram iniciados, mas não foram concluídos [17]

Lead Time: Tempo necessário para um produto percorrer todas as etapas de um processo, do início ao fim [17].

Commit: No contexto de ciência da computação e gerenciamento de dados, commit refere-se à ideia de fazer permanente um conjunto de mudanças experimentais. Sua tradução significa submeter [39].

Build: É o pacote gerado após o processo de *building*. *Building* é o processo de construção do pacote de implantação de um software.

Merge: Significa combinar as alterações realizadas simultaneamente em cópias diferentes de um mesmo código-fonte de forma a resultar em apenas um arquivo, contemplando ambas as alterações.

Repositório: Lugar onde se guardam coisas [40].

Refatoração: Trata-se do processo de limpar e reorganizar frequentemente códigos-fonte tornando-o mais limpo, mais claro e mais fácil de ser compreendido. Tais alterações não mudam o comportamento das funcionalidades, apenas melhoram a estrutura do código [32].

Freeware: Programa de distribuição livre. Notar que a língua inglesa tem apenas a palavra *free* para designar os termos "gratuito" e "livre", que têm significados diferentes em português. Assim sendo, o termo *freeware* não significa necessariamente que o programa é grátis. Isso só ocorrerá se o software tiver uma licença GPL [41].

Open Source: A liberdade do software livre se refere ao código ser aberto, ou seja, um programa é software livre quando pode ser usado, copiado, estudado e redistribuído sem nenhuma restrição. Conceitualmente, tal liberdade se opõe ao software proprietário, mas não ao software comercial, que é vendido almejando lucro [42].

Plugin: Na informática define-se *plugin* todo programa, ferramenta ou extensão que se encaixa a outro programa principal para adicionar mais funções e recursos a ele. Geralmente são leves e não comprometem o funcionamento do software e são de fácil instalação e manuseio [43].

Sprint: Cada *Sprint* pode ser considerada um projeto com horizonte não maior que um mês. Como os projetos, as *Sprints* são utilizadas para realizar algo. Cada *Sprint* tem a definição do que é para ser construído, um plano projetado e flexível que irá guiar a construção, o trabalho e o resultado do produto [20].

FOLHA DE REGISTRO DO DOCUMENTO			
1. CLASSIFICAÇÃO/TIPO TC	2. DATA	3. REGISTRO N°	4. N° DE PÁGINAS
5. TÍTULO E SUBTÍTULO:			
6. AUTOR(ES):			
7. INSTITUIÇÃO: Instituto Tecnológico de Aeronáutica – ITA			
8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:			
9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:			
10. APRESENTAÇÃO: (X) Nacional () Internacional ITA, São José dos Campos. Programa de Especialização em Tecnologia da Informação na Área de Gestão Estratégica de Projetos. Orientador: Eduardo Martins Guerra..Apresentação em _____, Publicada em 2012.			
11. RESUMO:			
12. GRAU DE SIGILO: (X) OSTENSIVO () RESERVADO () CONFIDENCIAL () SECRETO			