



Revista

Visão Ágil

Setembro 08 - Edição 05 - www.visaoagil.com

SCRUM + FDD

Criando um Backlog Orientado a Negócio

**Os 7 pecados capitais
de um time ágil.**

Confiança
O caminho para a
agilidade.

Lean
A arte de eliminar o
desperdício!

**As 5 Doenças do
gerenciamento de
projetos - Causa 5**

Estréias:

Sessão Perfil:

Entrevista com Ian Roughley

Entrevista com Marcio Marchini

Agile Page

Poster com descrição de técnicas ou metodologias ágeis a cada edição!

e mais...

Maré de Agilidade:

Agilidade no DF

Comunidade Visão Ágil

Referências Ágeis

**EM NOVEMBRO, A COMUNIDADE BRASILEIRA
DE DESENVOLVIMENTO DE SOFTWARE
VAI GANHAR UM FORTE ALIADO! NÃO PERCA...**

Evento de lançamento

InfoQ 
Comunidade de Desenvolvimento de Software
Brasil

Dia 01/11 em São Paulo

Inscreva-se já. Vagas limitadas
www.fratech.net/infoq

Realização:

Fratech

Patrocínio:

LOCAWEB
SERVIÇOS DE INTERNET

BACKLOG

O que você encontrará nesta Edição

ARTIGOS



Felipe Rodrigues

CONFIANÇA - O caminho para a Agilidade



Victor Hugo Germano

LEAN - A Arte de eliminar o Desperdício



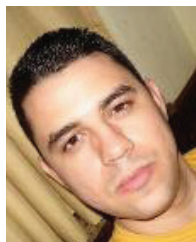
Adail Retamal

AS 5 DOENÇAS DA GERÊNCIA DE PROJETOS



Manoel Pimentel

CRIANDO UM BACKLOG ORIENTADO A NEGÓCIOS



Emerson Macedo

OS 7 PECADOS CAPITAIS DE TIMES ÁGEIS

PERFIL



IAN ROUGHLEY

O líder do projeto Struts 2 fala sobre seus projetos pessoais, sua experiência com métodos Ágeis, Integração Contínua e testes.

EDITORIAL

Ventos de mudança na Revista Visão Ágil

COMUNIDADE

Maré de Agilidade

Confira a cobertura do evento que agitou o Distrito Federal.

Referências Ágeis

Encontre conteúdo de qualidade em blogs de experts Ágeis



AGILE PAGE

Feature Driven Development

Origem, processos e ferramentas. Tudo numa única página!

PERFIL



MARCIO MARCHINI

O Brasileiro que encontrou o movimento Ágil no Canadá e está voltando para mostrar seu talento como CTO da Audaces Automação.

Ventos da mudança na Revista Visão Ágil

Caro Amigo Agilista,

Paz e Bem!

Inspeção e adaptação, são os elementos chaves da força das metodologias ágeis, pois nos fornecem uma forma factível e efetiva de aplicar o ciclo PDCA (Plan, Do, Check, Action) para nossa melhoria contínua, por isso, nós da Visão Ágil, como bons agilistas, não podíamos deixar de praticar esse “mantra da agilidade” na própria elaboração das edições de nossa revista.

Dessa forma, chegamos à quinta edição da Revista Visão Ágil, edição esta, que é extremamente especial por representar uma série de mudanças em nossa forma de trabalho e até mesmo no tipo e formato do conteúdo que oferecemos a você caro amigo leitor agilista.

Uma das maiores mudanças que passamos desde a última edição, foi no aspecto das pessoas por trás da Visão Ágil, pois tivemos grandes mudanças em nossa equipe editorial, pois agora, estão somando forças ao nosso trabalho, os meus grandes amigos Felipe Rodrigues e Victor Hugo (agilistas de primeira categoria), que sempre nos ajudaram como autores, com apoio logístico e com a participação e organização de eventos, mas agora, mergulharam mais fundo ainda nessa idéia da Visão Ágil, pois assumiram também o importante papel de Editores da revista.

Nessa edição então, tivemos uma melhor distribuição da carga de trabalho, nas atividades de planejamento, organização, seleção dos artigos, editoração e revisão. Também temos um maior critério e maior atenção ao processo de aceitação e revisão dos artigos, pois sabemos da responsabilidade e desafio que é levar informação de qualidade a você caro leitor agilista.

Outra grande mudança, foi que como você pode perceber, passamos alguns meses sem lançar nossa tradicional edição bimestral, porém, nesse período, não estivemos parados, pelo contrário, pois devido ao nosso importante papel como comunidade ágil aqui no Brasil, estivemos ampliando nosso raio de ação, através da participação em eventos e principalmente através da criação de novos canais de sucesso sobre agilidade como o nosso Blog Colaborativo Visão Ágil, nossa Biblioteca Pública e nosso programa Visão Ágil Academic Leader, que está formando líderes dentro das faculdades para divulgar e ajudar o entendimento da agilidade dentro dessas instituições.

Para concluir caro amigo agilista, temos certeza que você irá gostar ainda mais dessa edição, e queremos deixar reforçado o convite para você participar da forma que você quiser de nossos canais disponíveis através de nosso site www.visaoagil.com

Muito Obrigado e Boa Leitura,

Manoel Pimentel Medeiros - Diretor Editorial



CONFIANÇA

O caminho para a agilidade



Felipe Rodrigues de Almeida é Arquiteto de Sistemas com experiência de 5 anos em desenvolvimento de sistemas distribuídos. Atualmente trabalha em projetos pela Fratech, atuando na arquitetura de aplicações críticas. Participa ativamente do desenvolvimento do framework

Struts2 e mantém o projeto open-source BoxSQL. Palestrante no QCon em Londres. Passa o tempo livre curtindo e cuidando de seus 3 cães.

Lendo uma entrevista de Stephen R. Covey, autor dos livros Os 7 hábitos das pessoas altamente eficazes e de O Poder da Confiança fui conduzido a refletir sobre a confiança que meus clientes tem em meus serviços. Será que sou confiável o suficiente? Se sou confiável, será que meus clientes sabem disso?

Como coach de equipes e arquiteto de soluções da Fratech, me pergunto se transmito confiança

aos meus clientes. Questiono até que ponto eles confiam em meu trabalho e mais importante ainda, como posso aumentar a relação de confiança entre meu time e o cliente.

Antes de continuar, precisamos definir confiança. Confiança tem a ver com acreditar. Nós não confiamos em alguém se acreditarmos que sua capacidade é duvidosa. Dessa forma, não podemos confiar em quem não conhecemos. Isso nos leva a outro fato importante: Confiança é o resultado do conhecimento sobre alguém. Quanto mais informações corretas sobre quem necessitamos confiar, melhor.

Segundo Stephen, esse sentimento é a confiança nascida do fato de você acreditar no caráter de alguém e em suas competências e capacidades. Mas nenhuma dessas definições é tão precisa quanto a seguinte frase de Jack Welch:

“Eu poderia lhe dar a definição de confiança procurando no dicionário, mas você somente a reconhecerá quando senti-la.”

Dessa forma não precisamos ir muito longe pra perceber que o melhor medidor de confiança é o relacionamento próximo com aqueles em quem confiamos e que confiam em nós.

Mas quais são os benefícios que a confiança

pode trazer para nós e para o cliente? Quais os problemas provenientes da falta de segurança? Como criar uma relação de confiança em uma área tão desconfiada quanto a de gerência de projetos?

Quais são os benefícios que a confiança pode trazer?

Muitos. Aumento de faturamento, Redução de custos, liberdade para trabalhar, inovação, menos burocracia, motivação, produtividade, estabilidade e mais alguns. Não é preciso ser um gênio para entender como a confiança pode no trazer esses benefícios. Vamos traçar uma relação entre esses itens tão desejados nas empresas.

Há várias linhas possíveis, mas vamos começar por produtividade. Para que um funcionário seja produtivo precisamos nos assegurar que ele está motivado. A motivação é estimulada pela liberdade nas tarefas do dia a dia, que por sua vez exige menos burocracia e favorece a inovação. Com a possibilidade de inovar, surgem várias forma de aumentar os clientes (quem é que não gosta de soluções inovadoras?).

A produtividade também favorece a redução de custos. E pra terminar, tudo isso traz muita estabilidade para o funcionário produtivo e para a empresa, que terá clientes mais satisfeitos e leais. Alguém ainda tem dúvida dos benefícios que a confiança traz? Então onde está o problema? Porque ninguém quer cultivar confiança?

Quais os problemas provenientes da falta de confiança?

A falta de confiança interna em uma empresa gera, em seus funcionários, um sentimento de desconfiança também em relação aos clientes.





Cria-se uma situação desconfortável para todos os envolvidos, resultando em péssimo marketing interno e externo. A imagem de sua empresa é refletida por seus funcionários, portanto se eles refletem desconfiança, nenhum cliente irá confiar em você. Essa situação gerou um mercado muito instável e desconfiado, cheio de armadilhas que nada mais são tentativas de proteção excessiva.

Trazendo isso para o gerenciamento de projetos, e aqui estamos falando de todo tipo de projeto, podemos identificar a super proteção nas estimativas como falta de confiança. Segundo a teoria da Corrente Crítica (CCM - Critical Chain Management) proveniente dos conceitos de TOC (Theory of

Constraints) a super proteção nas estimativas são responsáveis pela maior parte dos atrasos e fracassos em projetos de todo tipo. Para resolver isso, a TOC propõe que a confiança seja restabelecida.

Outra evidência clara da falta de confiança é refletida nos contratos de projetos, onde é determinado um prazo, um escopo, um cronograma, porém a com algumas cláusulas de proteção para casos de falhas. Por exemplo, cláusulas que garantem que a qualidade ou o cumprimento dos prazos estão

atrelados à determinadas obrigações dos clientes. É comum que os contratos sejam estipulados já com o fracasso em mente. Ao primeiro sinal de problema, as cláusulas pro-

13 Comportamentos da confiabilidade

Um bom começo é basear-se em 13 comportamentos que vão construir e nutrir sua confiabilidade:

1. Falar diretamente com as pessoas
2. Demonstrar respeito
3. Ser transparente
4. Apresentar direitos e deveres
5. Mostrar lealdade
6. Apresentar resultados
7. Estar em constante desenvolvimento
8. Confrontar a realidade
9. Esclarecer as expectativas
10. Praticar contabilidade em sua empresa
11. Antes de tudo, ouvir
12. Manter suas promessas
13. Confiar nos outros

tetoras são colocadas em prática. Quem aqui nunca ouviu, em uma reunião de gerência, algo como, atrasamos porque o cliente não passou as informações. Ou viu situações onde as empresas tem de continuar desenvolvendo mesmo quando os recursos previstos em contrato já se esgotaram, mas o cliente ainda não tem o que precisava.



O resultado disso é que muitas vezes quando os dois lados deveriam buscar bom relacionamento e lucratividade, eles buscam formas de se proteger. Por isso é tão difícil convencer o seu cliente que escopo aberto é a melhor solução pra ele. Por isso é difícil para um gerente entender que programação em par é eficaz. Por isso as empresas tendem a cobrar cada vez mais, numa atitude que passa a mensagem de que “eu to pagando, então o problema é seu”.

Como criar uma relação de confiança?

Primeiro temos de identificar quais são os fatores que nos fazem confiar em alguém. Tentemos a confiar naqueles que agem a nosso favor, considerando nossos interesses. Confia-

mos em quem veste a camisa de nosso time.

Par conquistar a confiança de seu cliente é preciso oferecer segurança ao seu cliente. É preciso que ele entenda porque é tão importante confiar em você e principalmente é preciso que ele sintam-se seguro o suficiente para confiar em você. Passar conhecimento sobre aquilo que você faz com sua empresa, como você faz e porque você faz são fatores que favorecem a confiança do cliente. É preciso que sua empresa seja interessante aos olhos do cliente.

Clientes precisam saber que agimos de acordo com seus melhores interesses. Precisam saber que somos mais do que meros egocêntricos tentando engordar o gado. É importante transmitir isso para nossos clientes e não só citar a confiança em campanhas de marketing.

Agir verdadeiramente pensando nas necessidades dos clientes é o principal fator para conquistar a confiança.

E se a confiança estiver abalada?

É comum encontrar casos em que a relação de confiança já está abalada. Mas isso não quer dizer que a confiança não possa ser restabelecida ou até mesmo aprimorada.



O primeiro passo é identificar o motivo da quebra de confiança. Não importa se foi um ato consciente de traição, mau juízo, um erro honesto, um fracasso de competência ou um simples equívoco, o caminho para a restauração é o mesmo - aumentar a sua credibilidade e agir de forma que inspire confiança.

Você não pode forçar as pessoas a confiar em você ou em sua empresa, mas poder cultivar uma mudança de postura, deixando claro em suas novas ações que é uma pessoa credível e digna de confiança.

Como manter-se confiável em um ambiente insegurança?

Em ambientes desse tipo, todos jogam e se você não participar nem um pouco do jogo, seus resultados podem não ser tão bons. Mas é preciso bom senso e considerar a seguinte situação: Quando todos estiverem jogando em uma organização, aí a organização estará inteiramente baseada em comportamentos falsos, com falta de transparência. Isso vai contra as premissas de uma boa relação de confiança. Por isso existe retenção de informações, manipulação de pessoas, resultados etc. Essas práticas são também conhecidas como CYA (Cover Your Ass - http://en.wikipedia.org/wiki/Cover_your_ass).

Não estimule práticas CYA em sua empresa. Procure estabelecer relações transparentes, evitando jogar o jogo sujo. É um trabalho difícil e contínuo, mas que se feito com consciência será recompensador.

Conclusão

Mais uma vez, fica claro que devemos manter sempre os objetivos dos clientes como nossas diretrizes ao executar um serviço ou mesmo em outras áreas de nossa vida. Para

que tenhamos bons resultados é preciso conquistar bons resultados para nossos clientes.

Ficou claro também que relações de confiança são extremamente valiosas e devemos prezar aqueles que confiam em nossos serviços. É preciso confiar para que confiem em nós e assim estamos unindo forças para o sucesso e rompendo a idéia de que é cada um por si.

Entendo que as práticas ágeis favorecem a confiança mútua estreitando o relacionamento com nossos clientes e baseando-se em feedback antecipado e contínuo. Isso nos possibilita prever e antecipar uma situação ruim, podendo prevenir uma quebra de confiança.





TECNOLOGIA DA INFORMAÇÃO

VISÃO ÁGIL

INFOQ

WORKSHOPS

COACHING

MENTORING

PRÁTICAS ÁGEIS

SCRUM

DOMAIN DRIVEN DESIGN

TEST DRIVEN DESIGN

THEORY OF CONSTRAINTS

PRÓXIMOS TREINAMENTOS

Testes ágeis: Como testar em seus projetos - 18/10 - São Paulo
Domain Driven Design & Modelagem Ágil - 08-09/11 - Brasília

NÃO PERCA

Lançamento da InfoQ Brasil - Dia 01 de Novembro de 2008
Participe e ajude a fortalecer a comunidade brasileira de software!

acesse: www.fratech.net

Lean

A arte de eliminar o desperdício!

Nota sobre o Autor: Victor Hugo Germano é Bacharel em Ciência da Computação e Especializado em Gestão Estratégica de TI. Atualmente atua como integrante do Escritório de Projetos na empresa Audaces Automação, auxiliando no processo de implantação de metodologias Ágeis em Desenvolvimento de Software. Autor do blog A Maldita Comédia, também participa do grupo de desenvolvedores do Coding Dojo Floripa. Pode ser contatado através do correio eletrônico: victorhg@gmail.com.



Observação: Os textos contidos neste artigos são uma adaptação da série de artigos sobre Lean existentes em seu blog A Maldita Comédia



"Lean Thinking" (ou "Mentalidade Enxuta") é um termo cunhado por James Womack e Daniel Jones que denomina uma filosofia de negócios baseada no Sistema Toyota de Produção. Observando detalhes das atividades básicas envolvidas no negócio, identifica o que é o desperdício e o que é o valor a partir da ótica dos clientes e usuários.

As práticas envolvem a criação de fluxos contínuos de valor, Pull Systems baseados na demanda real dos clientes, análise e melhoria contínuas da cadeia de produção, desde as matérias primas até os produtos acabados, e o desenvolvimento de produtos que efetivamente atendam às necessidades dos clientes. A adoção dessa filosofia tem trazido resultados extraordinários para as empresas que as praticam.

Originalmente concebida por Taiichi Ohno e colaboradores, essencialmente como práticas de manufatura, tem sido gradualmente dis-

seminadas em todas as áreas da empresa em diferentes setores, tornando-se efetivamente uma filosofia e uma cultura empresarial. Em software não poderia ser diferente. Scrum detem em suas raízes muitos valores deste modelo, e é imprescindível para qualquer gestor, conhecer tais valores e práticas.

O que é Desperdício?

Desperdício é tudo aquilo que não adiciona valor a um produto, valor este que deve ser percebido pelo usuário.

Se um componente de software não utilizado pelo usuário é mantido no sistema, isto é desperdício.

Se um ciclo de desenvolvimento de software se esforça para levantar um livro de requisitos que não é lido, isto é desperdício.

Se os desenvolvedores produzem mais funcionalidades que o imediatamente necessário, isto é desperdício.

Em desenvolvimento de software, divisão excessiva do trabalho, como numa corrida de obstáculos, é desperdício.

Para o modelo Lean, o ideal é encontrar o que o cliente deseja, e então criar exatamente o que ele necessita, virtualmente no exato momento da solicitação. Para isso é necessário responder rapidamente às demandas e mudanças que ocorrem durante o desenvolvimento de um produto.

Qualquer coisa que entre no caminho de rapidamente satisfazer uma necessidade

do cliente é considerado desperdício!

Nas palavras do próprio Taiichi Ohno:

“Tudo que estamos fazendo é observar a linha do tempo entre uma requisição enviada pelo cliente e o momento em que recebemos o dinheiro. E estamos reduzindo esta linha do tempo removendo os itens que não adicionam valor.”

E valor? O que será?

Para poder eliminar os desperdícios, primeiro você deve reconhecê-los. Sendo desperdício qualquer coisa que não adiciona valor de negócio, é necessário discutir o que Valor realmente representa. Segundo Mary Poppendieck:

Não existe substituto no desenvolvimento de uma profunda compreensão do que os clientes realmente valorizarão que não seja a utilização real do software. Em nossa indústria, valor possui um hábito mutável, pois normalmente os clientes não sabem o que desejam. E além do mais, no momento em que eles vislumbrarem o sistema em funcionamento, a idéia do que eles desejam invariavelmente mudará. Ainda assim, grandes empresas de software desenvolvem um senso profundo de valor de negócio e continuamente surpreendem seus clientes. Pense no Google, regularmente e repetidamente surpreende clientes ao redor do mundo.

Encontrando desperdício

Em produção industrial, estoque é uma forma de desperdício. Estoque precisa ser gerenciado, movido, armazenado, substituído. Não apenas gasta tempo e esforço, ele amplia a complexidade do geren-

ciamento de toda a planta. Há muito tempo esse conceito foi incorporado na manufatura. Assim, pensar Lean é pensar em Just in Time Production

Por que em software seria diferente? Desperdício sempre será encarado como trabalho parcialmente feito. O problema? Aumenta a complexidade geral dos sistema, torna-se obsoleto, suga dinheiro.

Os sete desperdícios do Desenvolvimento de Software

Apresentado por Mary Poppendieck, na tentativa de organizar as principais fontes de problemas em projetos de software.

Trabalho Parcialmente Finalizado

Você deve possuir uma vaga idéia de como o sistema funcionará. Pode até ter uma noção de requisitos de negócio e condições de sucesso para uma funcionalidade. Quem sabe até mesmo algum código produzido.

Entretando, enquanto este código não estiver integrado a sua base de código principal, tiver sido validado e aceito por seu usuário, não há como prever o seu comportamento. Funcio-



nalidades pela metade apenas atravancam o caminho para trabalhos que poderiam ser feitos. Tornando-se facilmente obsoleto, trabalho parcialmente concluído é pior do que trabalho nenhum... Funcionalidades e códigos parcialmente finalizados ampliam a complexidade de seu sistema, e amplificam os custos de manutenção de sua aplicação.

Processos Extras

Você já se perguntou se todo o papel utilizado é realmente necessário? Algum procedimento para desenvolvimento de software requerem papéis para serem assinados por usuários, fornecendo rastreamento e permitindo a aprovação para uma mudança. Já verificou se o cliente realmente acha estes procedimentos valiosos? O fato de um documento ser um artefato de software não o torna valioso. Se você tem que esperar que um documento seja produzido para que você inicie o seu trabalho, reavalie a real necessidade de tal documento, e porque não, proponha mudanças. Lembre-se: procedimentos podem e devem ser questionados!

Funcionalidades a mais

Pode parecer ótimo entregar ao cliente mais funcionalidades do que ele está esperando... Pegá-lo de surpresa, e surpreendê-lo... e este é o desperdício mais sério.

Mais funcionalidades significa:

- Mais código para ser mantido.
- Mais testes para serem realizados.
- Mais documentos de especificação para serem criados.
- Mais conteúdo para gerar documentação.



- Mais bugs para aparecerem no sistema...

Enfim: jogar dinheiro no lixo... ainda mais se você pensar que 70% do total de funcionalidades de um sistema simplesmente não serão utilizadas.

Troca de tarefas

Organizar pessoas em múltiplos projetos é outra forma de desperdício. Por ser uma atividade essencialmente intelectual, é necessário, toda vez que ocorrer a troca entre tarefas de projetos diferentes, que o desenvolvedores foque-se em algo completamente novo. Contexto de projeto diferentes, realidades diferentes, procedimentos muitas vezes diferentes...

O tempo que ele levará para estar realmente produtivo em cada projeto reduzirá muito! Sem contar que, normalmente, ocorrerá uma guerra de interesses entre os projetos pela atenção do desenvolvedor...

Minha vó já dizia: fazer muitas coisas ao mesmo tempo é um pretexto para não finalizar nenhuma

Espera

O maior ponto de desperdício em desenvolvimento de software é esperar para que algo seja produzido. Esperar para iniciar projetos, realizar testes, aguardar o levantamento de requisitos detalhadas e desenvolver uma funcionalidade são consideradas desperdícios, e devem ser combatidos.

Atrasos distanciam seu usuário de receber um produto que satisfaça suas necessidades. Em muitos casos, atrasos são apenas a ponta do iceberg para problemas muito maiores. Em ambientes e mercados muito competitivos, atrasos e esperas podem significar milhões em investimento perdidos.

Poucos consideram custos realacionados a participação do mercado como sendo influenciada por atrasos, ou conseguem quantificar o Custo de Oportunidade envolvido em ser o primeiro a entrar num mercado... mas os principais líderes do mercado atualmente foram os pioneiros em seus nichos...

Movimento

Desenvolvimento de software é uma atividade que demanda muita concentração, e interrupções podem ser desastrosas no trabalho realizado. Quanto tempo um desenvolvedor leva para encontrar uma informação? Questões sobre requisitos de negócio ou informações técnicas levam quanto tempo para serem respondidas?

Documentos que transitam entre setores também são desperdícios. Quanto mais pontos um documento passar para iniciar sua implementação, maior será o ruído causado pela interpretação de informações (efeito telefone sem fio).

Defeitos

Você sabe o quanto custo a correção de um problema em fase desenvolvimento? e na integração? UM ABSURDO!!

Defeitos no sistema não agregam valor, não satisfazem o cliente e definitivamente não são baratos. Qualquer tipo de retrabalho causado por problemas de implementação de requisitos, interpretação de necessidades e correção de documentação devem ser severamente combatidos. Neste ponto os métodos ágeis diferem muito dos modelos mais tradicionais: Construa com qualidade, não tente aferir qualidade após o produto pronto. Crie um sistema que evite que os erros ocorram, e não que verifique se eles estão presentes em seu sistema para depois corrigí-los.



Tentar eliminar alguns dos desperdícios existentes no desenvolvimento de software é o primeiro passo para melhorar a qualidade de seu sistema... Tente!

Amplie o Aprendizado!

As origens do Lean estão diretamente ligadas à indústria. Entretanto, os princípios Lean podem ser aplicados diretamente ao desenvolvimento de software.

É preciso deixar claro: Criar software não é um processo produtivo; a atividade é melhor comparada a um processo de criação.

Mas qual a diferença?

Entenda o processo de desenvolvimento de software como uma atividade semelhante à do Chef de cozinha preparando um novo prato: é necessário reunir vários ingredientes, combiná-los de inúmeras formas e testar os resultados para poder aprimorá-los. Um processo produtivo seria eu, o Victor, após ver a receita no programa da Ana Maria Braga, reproduzi-la. Para um, a variações de experiências são extremamente importantes para atingir os melhores resultados... para o outro (eu e a Ana Maria Braga), a situação é diferente. Variações não são toleradas, sob pena de não ter o prato criado.

O método científico

Software está inserido no grupo de atividades chamadas problem-solving. Isto quer dizer que, de antemão, toda a intenção em criar um software está ligada à resolução de um problema.

Para problemas complexos, é necessário ter o máximo de informação possível para que

se possa propor uma solução apropriada. A abordagem coerente para tal é utilizar o método científico:

- Observe
- Crie uma hipótese
- Execute um experimento



- Verifique se os resultados são condizentes com sua hipótese

O ponto interessante de tal abordagem é que, caso suas hipóteses estejam sempre corretas, você não conseguirá aprender muito. Caso exista uma possibilidade de falha em suas hipóteses, a quantidade de informação adquirida em experimentar será imensamente mais rica.

Existem duas escolas bastante distintas na área

de software:

A primeira, motiva um desenvolvedor a ter certeza que todas as suas soluções - design, código e testes- estejam perfeitas à primeira execução. Neste ambiente existe pouco espaço para a geração de conhecimento através da experimentação. Esta abordagem funciona melhor em problemas bem estruturados, que normalmente possuem uma única solução já bem conhecida e um caminho preferido para alcançar tal solução. Por exemplo, os principais problemas apresentados em colégios.

A segunda escola afirma ser melhor possuir ciclos rápidos de experimente - teste - corrija do que garantir à primeira vista que um código está perfeito. Citando Edward Yordon:

Um pedaço de lógica em código normalmente precisa ser reescrito três ou quatro vezes para ser considerado um trabalho elegante e profissional. Então por que existe tanta resistência em refatorar código quando estamos sempre felizes em poder revisar documentos várias vezes até chegar a um resultado satisfatório?

A visão Lean no desenvolvimento de software

Transforme a experimentação e o feedback numa constante. Permita que os desenvolvedores possam, de forma rápida e barata, tentar formas novas de aumentar sua qualidade e produtividade. Utilize os resultados para aprender o máximo que puder e não tenha medo de tomar decisões erradas.

Iterações curtas permitem um feedback instantâneo dos clientes e usuários sobre a satisfação e a utilidade que seu sistema apresenta.

Retrospectivas são um momento importantíssimo do processo de desenvolvimento de software, já que é possível encontrar problemas que afetam a todos numa equipe, que dificilmente seriam percebidos.

“Learning is everything”

Como maximizar o feedback?

Crie o hábito de fazer análises em seus projetos. Lean espera a criação de uma Organização que Aprende, e a principal forma de fazer

isso é ampliar a quantidade de informação que chega a todas as pessoas da empresa. Seguem algumas dicas

Passo a passo para ampliar o feedback

1) Escolha o seu problema mais difícil e encontre uma maneira de ampliar o feedback

a. Amplie o feedback das equipes de desenvolvimento à gerência perguntando a cada equipe ao término de uma interação as seguintes perguntas:

- A equipe possuiu todos os conhecimentos necessários para esta iteração?
- Algum recurso necessário faltou?
- Como podemos modificar nosso trabalho de forma a tornar o desenvolvimento mais rápido e melhor?
- O que está atrapalhando o nosso caminho?

b. Aumente o feedback dos clientes e usuários para a equipe de desenvolvimento mantendo um grupo com foco no usuário final. Encontre as respostas para as perguntas:

- O quão bem esta sessão do software resolverá o problema para o qual ela foi criada?
- Como pode ser melhorado?
- Como esta última iteração afetou a sua percepção sobre as suas necessidades?
- O que você precisa para colocar este último incremento de software em produção?

c. Ampliar o feedback do produto para a equipe de desenvolvimento pode ser feito

através de:

- Ter os desenvolvedores escrevendo e executando testes de código à medida que o código de produção é escrito
- Ter analistas, clientes ou testadores executando e criando testes de aceitação à medida que os desenvolvedores trabalham no código. Permita que os desenvolvedores auxiliem nestes testes com o intuito de automatizar tudo.
- Permita que desenvolvedores tenham acesso aos testes de usabilidade de cada uma das funcionalidades próximas de serem finalizadas, para que eles possam verificar como os usuários reagem à implementação

d. Amplie o feedback dentro da própria equipe:

- Torne os testadores em parte integrante da equipe
- Incorpore todos os envolvidos no projeto desde as primeiras fases
- Estabeleça a política de que a equipe de desenvolvimento deverá manter o produto.

2) Inicie iterações com sessões de negociação entre clientes e desenvolvedores. Clientes devem indicar quais funcionalidades possuem a prioridade máxima, e os desenvolvedores devem selecionar e comprometer-se somente com aquelas que eles acreditam realmente serem capazes de completar no período fechado da iteração

3) Insira os gráficos de progresso para seu projeto em uma área comum para que a

equipe(s) possa visualizar sem problemas o que precisa ser feito e que todos possam ver para onde o projeto está convergindo

4) Se você dividir um projeto através de múltiplas equipes, faça um esforço a mais para dividir a arquitetura de maneira a permitir que elas possam trabalhar o mais independentemente possível. Encontre maneira dos times sincronizarem o próprio trabalho o maior número de vezes possível (build contínuo do sistema todo)

5) Equipes independentes devem considerar também a interface do usuário de forma independente

6) Encontre o seu problema mais complexo e faça sua equipe conseguir três soluções viáveis para resolvê-lo. Ao invés de escolher apenas uma das soluções, permita à equipe experimentar todas as três opções ao mesmo tempo.

Nas próximas edições, falaremos sobre os demais pontos que tornam o Lean Thinking uma evolução natural para as equipes de desenvolvimento de software.

Referências

[1] Mary Poppendieck: Lean for software development from concept to cash

[2] Mary Poppendieck: Lean Software Development - an Agile Toolkit

[3] Lean Enterprise Institute <http://lean.org>

[4] A Maldita Comédia: <http://malditacomedia.blogspot.com>

As Cinco Doenças do Gerenciamento de Projetos

5: Matemática do Projeto

Autor Original

Allan Elder é presidente da No Limits Leadership, Inc., uma empresa de consultoria dedicada a ajudar as organizações a entregar mais projetos, mais rápido, através da liderança eficaz. Allan trabalhou como diretor de GSI (Gerenciamento de Sistemas de Informação) para a segunda maior empresa de seguros corporativos e a maior empresa de segurança na Califórnia. Allan foi certificado como PMP, é um "Jonah" na Teoria das Restrições, possui bacharelado em Telecomunicações, mestrado em Gerenciamento de Projetos e Ph.D. em Organização e Gerenciamento. Além de seu trabalho em consultoria, Allan é o principal instrutor de gerenciamento de projetos para a Universidade da Califórnia, Irvine, prestou consultoria e lecionou para a UCI Graduate School of Business, e foi um Examinador Sênior para o California Award for Performance Excellence (CAPE) por três anos. Allan pode ser contatado em aelder@nolimitsleadership.com.

Tradutor



Adail Muniz Retamal é diretor da Heptagon Tecnologia da Informação Ltda, empresa de consultoria e treinamento focada na aplicação da Teoria das Restrições em geral, e da Corrente Crítica em particular, à Engenharia de Software, metodologias ágeis de gerenciamento e desenvolvimento de software, atuando como catalisador da mudança organizacional, além de ajudar as organizações a construírem sua estratégia e tática para alinhar seus esforços com suas metas. Adail

é Engenheiro Eletricista/Eletrônico, atuou como consultor, instrutor e arquiteto de soluções para a Borland Latin America por 4,5 anos. Lecionou em universidades públicas e privadas. É palestrante, articulista e está escrevendo um livro. Adail pode ser contatado em adail@heptagon.com.br.

Causa 5: $2 + 2 = 5$

Em gerenciamento de projetos as coisas nem sempre são o que parecem. Esse efeito negativo é bastante simples, embora geralmente negligenciado. Se você tivesse um projeto com duas tarefas, cada uma com 2 dias de duração, quanto tempo ele levaria? Sem considerar as questões acima, a resposta acima seria 4 dias. Porém, você sabe que a probabilidade é muito menor do que você pensava. Eis o cenário: você está trabalhando na Tarefa-1 e sua saída vai para o recurso para realizar a Tarefa-2. Digamos que você é um santo e que começa sua tarefa no prazo, trabalhou nela até terminar, sem multitarefa, e completou-a no final do dia 2, como prometido. É normal completar seu trabalho,

imediatamente levar seus resultados para a próxima pessoa (como se você realmente soubesse quem é) e entregá-los a ela para que comece? Não. Todavia, você conclui que terminou no prazo, então você relata o término para obter o crédito de ter terminado no prazo e dá o dia por encerrado. Na manhã seguinte você pega seu café, lê seu e-mail e faz qualquer outro trabalho pendente. Então você pega seus resultados de ontem e os entrega à próxima pessoa na fila. Agora perdemos metade de um dia.

Parte II do cenário: Será que a pessoa para quem você entregou os resultados irá parar imediatamente o que estiver fazendo, limpar a mesa e começar a trabalhar no próximo passo para esse entregável? Não. Ela vai lhe agradecer pela entrega, apreciar que você está geralmente no prazo e trocar fofocas com você. Ela reconhece que não há pressa para começar imediatamente, já que existe uma rede de segurança embutida nas estimativas de duração. Agora é a hora do almoço. Ela

retorna ao escritório, realiza mais algum outro trabalho, e pensa em iniciar sua tarefa, mas, é o final do dia, e nada como um novo início amanhã. Ela vai pra casa. Agora, outro meio dia está perdido. Quando ela começa a tarefa na manhã seguinte, já está um dia atrasada, causando atraso para outros, e agora o projeto está atrasado. Assim



é como uma tarefa de 2 dias mais uma de 2 dias resulta em 5 dias. Alguns argumentam que a segunda pessoa muito provavelmente irá acelerar e terminar em um dia, e o projeto não ficaria atrasado. Isto poderia acontecer. Provavelmente acontece frequentemente. Mas isso admite que a tarefa não era mesmo uma de dois dias, mas uma tarefa de um dia com um dia de segurança, que foi perdida, atrasando o projeto. Isso pode facilmente ser superado garantindo que cada nome de tarefa inclua a transferência. Por exemplo, em vez de "Completar o relatório" como uma tarefa, poderia ser "Marketing obtém o relatório completado". Agora a pessoa que realiza a tarefa não ganha crédito por completá-la até que esteja nas mãos do próximo recurso. Além disso, a



pessoa que realiza a tarefa não pode marcar sua própria tarefa como terminada. Apenas o recurso que precisa de sua entrega pode validar que você "terminou". Isto impede entregas atrasadas assim como fornece ao próximo recurso na fila a oportunidade de rejeitar resultados com má qualidade, que impactem seu trabalho. Este método foi chamado de efeito "papa-léguas" pelo Dr. Goldratt. O efeito é similar à corrida de revezamento, onde devemos ter certeza que a próxima pessoa na fila está sempre pronta para iniciar o trabalho numa tarefa do projeto quando entregue. Muitos gerentes de projetos vão tão longe quanto contatar o próximo recurso na fila e notificá-lo sobre quando sua tarefa predecessora terminará, para que tenha oportunidade de limpar sua mesa em preparação para o trabalho vindouro.



Conclusão

Agora você conhece as cinco principais doenças que fazem seus projetos atrasarem. Para remover esses obstáculos você precisará parar a multitarefa nociva, desenvolver um sistema que permita que as tarefas adiantadas e atrasadas se compensem entre si, que leve em conta a probabilidade dos eventos dependentes, que pare os efeitos da Lei de

Parkinson, que garanta que, quando uma tarefa é completada, o resultado aparece quase instantaneamente para a próxima tarefa na fila, e que pare a prática de adicionar segurança a cada tarefa.



com

David J. Anderson
Modus Cooperandi e
blog AgileManagement.net

Zen of Agile Management

21 e 22 Outubro 2008 São Paulo . SP

- ✓ Como criar ou destruir confiança
- ✓ Minha receita para o sucesso
- ✓ Métricas, medição, rastreamento e indicadores para a Gestão Ágil
- ✓ Identificação de gargalos e Gestão por Restrições (TOC)
- ✓ Identificação de perdas
- ✓ Gestão de problemas e riscos
- ✓ Planejamento simples de iterações
- ✓ Planejamento avançado e previsível de iterações
- ✓ Sistemas puxados e "Kanban"/Lean
- ✓ Seleção do mix de produtos e priorização
- ✓ Teoria das Opções Reais e o suporte à decisão
- ✓ Combinação de Agile e CMMI
- ✓ Cultivar uma organização ágil de alta maturidade (cultura "Kaizen")

Tradução simultânea e material em português
pt-br
Brasil/pt

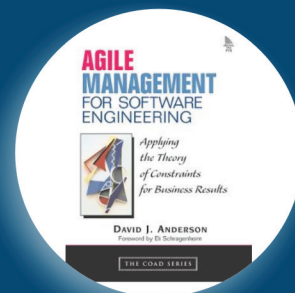
David J. Anderson



Realização:

HEP7AGON
TECNOLOGIA DA INFORMAÇÃO

Para inscrições e mais informações:
www.heptagon.com.br/ws-zen-agile-mgmt





PERFIL

Entrevista com Ian Roughley: Líder no desenvolvimento do framework Struts2

por Manoel Pimentel



Visão Ágil: Ian, como é seu trabalho nos projetos em que participa?

Ian Roughley: Nos projetos eu assumo uma variedade de papéis, desde desenvolvedor até arquiteto, gerenciando times de

desenvolvedores. Eu acho que ter todas essas perspectivas faz de mim um contribuinte mais efetivo. Da perspectiva técnica, a maior parte do meu trabalho utiliza Java e JEE, apesar de que eu estou começando a explorar e usar linguagens dinâmicas e frameworks como Ruby, Rails, Groovy e Grails. É uma época muito boa para ser um desenvolvedor – tendo tantas linguagens diferentes para escolher e sendo capaz de escolher a correta para cada tarefa específica.

VA: Você usa algum método ágil como XP, Scrum ou FDD? Se sim, como é?

IR: Embora desenvolvimento ágil seja mais utilizado hoje em dia do que era há alguns anos, ainda há muitas empresas onde agile é um conceito novo e são nessas empresas que eu normalmente trabalho. Isso quer dizer que boa parte do meu trabalho envolve introduzir processos ágeis no processo de desenvolvimento e eu gosto de usar uma abordagem bottom-up.

A abordagem bottom-up significa que ao invés de falar sobre os benefícios, eu mostro resultados implementando práticas ágeis. A ordem que eu implemento as práticas é a seguinte:

1. Integração Contínua – Primeiro, eu gosto de ter a integração contínua funcionando, assim eu tenho uma base estável para o

crescimento. Isso também oferece feedback imediato conforme os desenvolvedores colocam código novo no repositório. Também

... É uma época muito boa para ser um desenvolvedor – tendo tantas linguagens diferentes para escolher...

acho que alertas via email é fundamental aqui, assim todos sabem imediatamente o status do projeto.

2. Testing (Testes automatizados unitários, de integração e funcional) – Dependendo de quanto tempo o projeto está em desenvolvimento e do nível de teste existente, isso pode ser um passo fácil ou extremamente difícil. Em empresas que não estão usando práticas ágeis, introduzir testes não é somente escrever os testes, mas sim um problema muito maior de ajudar os desenvolvedores a entender porque testes automatizados devem ser feitos e quais problemas eles ajudam a resolver.

3. Planejamento e Iterações – a peça final do quebra-cabeça é utilizar planejamento e iterações. Este é de longe o maior passo conceitual para as empresas porque eles podem sentir que estão perdendo o controle sobre o gerenciamento dos projetos. Uma boa maneira de introduzir iterações é usar o plano de projeto existente, mas com intervalos específicos (digamos toda 2 a 4 semanas) reunindo todos os envolvidos em uma sala e demonstrando as funcionalidades nas quais a equipe trabalhou. Durante esta reunião, mais idéias surgirão e mudanças serão postas

à mesa. Depois de algumas dessas reuniões, é mais fácil realizar a transição entre um gerenciamento waterfall para o planejamento através de iterações.

Isso não para aqui, mas depois que essas 3 práticas estão implementadas, as empresas estão no seu caminho para encontrar seu processo de desenvolvimento ágil ideal.

VA: Você é um dos líderes do projeto Struts 2 que é um projeto open-source muito difundido. Vocês utilizam algum método ágil?

IR: Desenvolvimento open-source é um animal interessante. Os desenvolvedores estão distribuídos geograficamente por todo o mundo, o esforço de gerenciamento é mínimo e o tempo de dedicação varia muito para cada desenvolvedor. Se esse tipo de ambiente fosse introduzido em um ambiente corporativo, certamente falharia. Mas dependendo do envolvimento dos desenvolvedores em um projeto open-source, o projeto será um sucesso.

A principal ferramenta de planejamento utilizada para o Struts2 é o JIRA da Atlassian. Esta ferramenta é utilizada para organizar todas as novas funcionalidades e bugs e cada item atribuído a um desenvolvedor e um release (maior ou menor). A regra geral é que quando todos os itens de um release estão completos, a versão é liberada. Na maioria das vezes os itens são resolvidos, mas quando há uma circunstância especial (por exemplo um importante release de segurança) os itens podem ser movidos para uma próxima versão e a versão corrente é liberada. Pelo fato dos desenvolvedores trabalharem durante a noite ou em finais de semana no Struts e a quantidade de trabalho executado é balanceado por outras atividades pessoais,

o tempo exato de reales é difícil de calcular e isso é o melhor que pode ser alcançado.

Struts depende pesadamente de integração contínua usando Bamboo (outra ferramenta da Atlassian) para agendar o build e relatórios e o Apache Maven2 para executar o build e resolver as dependências. A base de código é muito bem testada.

Comunicação é a base para os ideais da Apache e é muito transparente – qualquer um pode acompanhar os itens que são importantes para si próprio e qualquer um pode perguntar e contribuir para o processo de design (seja ele um commiter ou não). Para facilitar isso, toda a comunicação sobre o desenvolvimento e design do framework acontece na lista de emails dos desenvolvedores. Qualquer comunicação fora da lista é desencorajada. Assim como as discussões, todo o código está em um repositório público, assim todos podem ver quais mudanças foram aplicadas. Isso significa que qualquer um pode baixar o código, corrigir problemas e enviar um patch para a lista de emails ou pelo JIRA. Se você quer se tornar um committer, este é o primeiro passo. Embora isso não seja exatamente programação em par, descobri que isso permite alcançar o mesmo objetivo – para desenvolvedores aprenderem sobre o código base e não apenas sua pequena parte.

VA: Como é feito o processo de modelagem nos seus projetos?

IR: Dependendo do cliente e do time, a modelagem pode seguir diferentes direções. Eu penso que na modelagem é importante capturar a intenção do design e comunicar isso para todos os envolvidos.

Sendo uma pessoa que gosta de coisas visuais, no nível mais simples eu uso o que eu chamo de modelagem por esboço ou “guardanapo”. Isso é um layout bem rápido de idéias ou elementos envolvidos (normalmente um diagrama de classes UML e às vezes um diagrama de sequência) que pode ser feito num quadro branco, papel ou usando qualquer programa de modelagem. O objetivo é compartilhar idéias com outros e confirmar que tudo que é necessário está incluso. Após o design ter sido codificado o modelo é descartado. Se for necessário modificações, um novo modelo é desenhado incluindo somente as peças relevantes. Isso garante que o modelo utilizado esteja sempre atualizado.

Quando uma abordagem mais rigorosa é necessária, digamos que o modelo é requerido para inclusão em documentos de design, os modelos de esboço são criados usando programas de modelagem e então inclusos na documentação. Geralmente eu acho esse um processo muito mais trabalhoso porque é muito difícil manter o mesmo nível de granularidade do design no documento – especialmente quando há diversos autores. O que eu quero dizer é que no modelo deveria haver relacionamentos e conceitos que são comunicados e não apenas um detalhamento de baixo nível. Por exemplo, se todas as propriedades de uma classe Java são fornecidas em um diagrama de classe UML no documento de design, então todas as vezes

que uma mudança for feita no código, o diagrama precisa ser gerado novamente e o documento de design precisa ser modificado. Isso acrescenta um esforço a mais significativo para implementar funcionalidades e corrigir bugs. Se o trabalho está sendo executado em um ambiente muito rígido, esta abordagem pode não ser possível.

VA: Qual a visão que seus clients tem sobre os processos ágeis?

IR: Isso varia. Eu recentemente trabalhei com uma empresa que abraçou o desenvolvimento ágil. Outros de meus clientes ainda trabalham com processos waterfall.

Ao meu ver, como desenvolvedores nós sempre teremos uma nova tecnologia com a qual queremos trabalhar ou uma nova forma de trabalhar com as tecnologias

existentes e o agile é apenas mais uma dessas. De um ponto de vista tradicional, empresas farão planejamento prévio de orçamento com 5, 6 meses ou 1 ano de antecedência, planejamento de marketing e vendas e planejamento de recursos de uma vez só. Isso se encaixa perfeitamente com o modelo waterfall. Quando dizemos que “entregaremos tudo o que você precisa mas não sabemos exatamente quando” faz com que os gerentes fiquem nervosos. Eu acho que a abordagem bottom-up é a melhor opção nesses casos – apenas iniciando o desenvolvimento ágil e técnicas de planejamento em um processo waterfall. Mostrar os resultados é a coisa mais

...O maior problema que eu vejo ao começar com um processo ágil é não dar tempo suficiente ao projeto. Se você não sabe nadar, não pode achar que vai pular na água e sair nadando bem em uma semana...

importante. As pessoas descobrem que o trabalho é entregar e que eles podem mudar a prioridade de uma funcionalidade a qualquer momento. Eu introduzi agile em algumas empresas desta maneira e funcionou muito bem.

Projetos que estão começando são frequentemente mais abertos ao desenvolvimento ágil, mas apenas à um conjunto de práticas. Comunicação (entre desenvolvedores e pessoal de negócio), mudança de prioridades e iterações são usadas, além de testes.

VA: Quando você começou a usar processos ágeis, quais problemas você encontrou?

IR: Como muitas coisas, o maior problema que eu vejo ao começar com um processo ágil é não dar tempo suficiente ao projeto. Se você não sabe nadar, não pode achar que vai pular na água e sair nadando bem em uma semana. O mesmo serve para desenvolvimento ágil – vai levar algum tempo. Considerando isso, o desenvolvimento ágil gira em torno de interação de grupo e comunicação, então é como se tivéssemos 5 pessoas aprendendo a nadar ao mesmo tempo. Se você está começando eu sugiro de 4 a 6 semanas ou 1 a 2 iterações para as pessoas começarem a entender como as coisas funcionam. Depois disso, ainda vai levar um tempo para todos ficarem confortáveis com o processo. Lembre-se – persista e molde o processo para aquilo que funciona com o seu time.

VA: Como você planeja e estima?

IR: Eu gosto de manter as coisas o mais simples possível. Quando uma nova funcionalidade é solicitada, um tempo é reservado para estimar rapidamente quanto tempo a implementação levará. Então esse item é

adicionado na fila de funcionalidades até que seja selecionado para inclusão em uma iteração. Uma vez que as funcionalidades estejam selecionadas para uma iteração, elas são re-avaliadas – para certificar-se de que a estimativa ainda é válida e que o sistema não mudou de forma significativa (no que diz respeito à funcionalidade em questão). Neste momento a estimativa deve ser mais precisa já que a sua aplicação é melhor compreendida pelo time.

Escolher as funcionalidades para cada iteração é sempre responsabilidade do cliente ou do usuário final, com uma exceção: Acho muito importante balancear tens de infraestrutura/desenvolvimento com funcionalidades de usuários final, então cada iteração irá incluir algumas dessas funcionalidades também. Isso permite que o software evolua e continue fresco.

VA: Em sua opinião, qual o ganho que os conceitos ágeis oferecem para atividades como requisitos, definição de arquitetura, modelagem, testes e gerenciamento de configurações?

IR: O maior ganho ao usar o desenvolvimento ágil para mim é em comunicação. Trazer o usuário final para perto dos desenvolvedores, a pessoa desenvolvendo a solução terá um melhor entendimento sobre o problema, resultando em uma solução melhor. Os usuários final também terão a sensação de participação, afinal eles podem ver como estão contribuindo para o produto e como tempo vão moldando o resultado final.



maré de agilidade

Saiba como foi o evento ágil no Distrito Federal



Sobre o Autor: Bruno Pedroso é desenvolvedor com mais de 10 anos de experiência. Atua hoje como coach de projetos e coordenador da área técnica da SEA tecnologia, dando grande enfoque à aplicação de valores e princípios XP.



No último dia 13 aconteceu em Brasília a primeira edição do evento Maré de Agilidade, organizado pela SEATEcnologia, com patrocínios RedHat, e apoio Fratech, APLTic, Sinfor e InfoQ. O evento, com entrada gratuita, reuniu no auditório do edifício Central Park (Setor Comercial Norte) cerca de 100 pessoas entre estudantes e profissionais da área, que puderam assistir a um dia inteiro de palestras e participar de algumas boas discussões.

Há pouco mais de um ano, desde que se formou a lista de discussão AgilDF (no google-groups), a comunidade brasiliense de agilistas vem se organizando para debater as novas visões sobre o desenvolvimento de software. A lista que conta atualmente com cerca de cem associados, ainda é modesta se compararmos ao volume de discussões de outras listas, como agile-brasil ou xprio, mas o envolvimento da comunidade tem sido de fundamental importância para a difusão das idéias ágeis no contexto específico das demandas de software ligadas ao poder público.

O evento foi composto por uma thread única com 5 apresentações, sempre culminando em pequenas discussões entre uma fala e outra. As palestras apresentadas foram as seguintes:

"O pensamento ágil", apresentado por Bruno Pedroso, abriu a manhã expondo as principais motivações do movimento ágil. Com enfoque mais teórico, a apresentação abordou questões básicas como valores e princípios ágeis, e as mudanças de paradigma em relação à gestão tradicional de projetos (ou gestão old-school, como se referiu o palestrante.) Abordou questões sobre a gestão de processos empíricos, ciclo PDCA, design



evolutivo, testes automáticos e outros. Ao final, as palavras-chave deixadas como síntese foram: ROI, Aprendizado, Adaptabilidade, Simplicidade. "Show de bola! Fez muita gente ali ficar curiosa sobre a agilidade e acho que esse era o objetivo", observou Pablo Nunes.

Em seguida, tivemos a apresentação do professor Dr. Wander Cléber Pereira, do instituto Ibmec, intitulada "Aspectos humanos da gestão de projetos: enfoque conversacional". Com uma abordagem bem diferente das velhas palestras sobre tecnologia com que estamos acostumados, o palestrante expôs questões básicas sobre os chama-

Ao final da palestra, além do sorteio de brindes, tivemos boas discussões sobre planos de comunicação e outros elementos típicos relacionados à documentação formal dos projetos. Infelizmente, não pudemos continuar com a discussão por conta do horário. Paramos para o almoço.

Na parte da tarde, tivemos uma ótima apresentação do Felipe Rodrigues, da Fratech, intitulada "Práticas ágeis para desenvolvimento de software". Com um enfoque mais prático, ele fez uma apresentação bem dinâmica e participativa, envolvendo bastante o público e respondendo às dúvidas na hora. Abordou

detalhes práticos de algumas metodologias específicas, como DDD, FDD e TDD (quanto D!). A apresentação foi seguida de bastante discussão, puxada pelo público cuja curiosidade e interesse pareciam vir crescendo ao longo do evento. "Na hora das perguntas, quando o público participava, foi ótimo, pois gerou um bate-papo saudável onde praticamente todos estavam interagindo e assim a disseminação do conhecimento aconteceu de forma mais clara e abrangente", comentou Wesley Costa. Segundo Pablo Nunes, "ele moti-



dos micro-processos de um projeto, ressaltando a importância e a complexidade dos elementos básicos de comunicação entre os membros de uma equipe. Tratados como observadores ativos, os integrantes do projeto são vistos como elementos multi-dimensionais complexos, cuja compreensão foi destrinchada e discutida com uma profundidade rara, especialmente se levarmos em conta a natureza do público.



vou as pessoas a perguntarem e tirarem suas dúvidas e de repente o tempo acabou. Muita gente saiu dali com a cabeça pipocando de idéias!"

As questões práticas foram talvez o macro-



tema da parte da tarde, pois logo em seguida assistimos a uma apresentação "em par", feita por Renato Willi e Bruno Pedroso, que mostraram de forma dinâmica alguns exemplos práticos e resultados reais que a SEA vem colecionando nos últimos anos com a utilização de XP e Scrum. Em um bate-bola bem coordenado, os palestrantes deram uma boa visão de como os princípios ágeis têm influenciado positivamente seus projetos. A apresentação relativamente rápida foi a resposta ágil dada a um pequeno atraso de cronograma - causado pelas discussões calorosas durante a palestra do Felipe - e à alta participação do público. "Preferimos abrir mais espaço para as perguntas no final, pois o pessoal parecia estar com bastante vontade de falar", colocou Renato Willi.

Ao final, assistimos à palestra "Produtividade, controle e desempenho em ambientes livres", apresentada por Alexandre Gomes que criou uma relação interessantíssima entre o movimento ágil e as últimas tendências tecnológicas e corporativas relacionadas à tecnologia Java. Diversidade, simplicidade e efetividade são algumas palavras-chave que remetem ao conteúdo exposto. Uma apresentação divertida e motivante que segurou boa parte do público até as 18:30 da tarde - em pleno sábado!

O evento terminou com mais alguns sorteios de brindes, apertos de mãos e últimos comentários sobre as palestras.

"Achei que as palestras foram muito bem preparadas e serviram para desvendar ainda mais o tema agilidade no desenvolvimento de software. Já tinha lido muito a respeito - principalmente nos blogs do pessoal da Globo.com e da Improvelt - mas nada como ver e ouvir (pessoalmente) as experiências de empresas tão próximas (da nossa realidade aqui de Brasília) na implementação das práticas ágeis.



Por meio das palestras, fiquei conhecendo a lista de discussão AgilDF e acredito que por ela ficarei ainda mais inteirado dos eventos, dicas e assuntos do mundo ágil. Parabéns a todos que se empenharam para a realização do evento. Estou esperando os próximos." (Leonardo L. Ferreira)

Da parte da organização do evento, a impressão que ficou foi que o tempo foi curto. Sentimos muita falta de ter montado uma mesa redonda, ou painel, de modo que o público pudesse participar mais ativamente e trazer à tona suas dúvidas e receios. "Nada que uma segunda edição do evento não

possa remediar", comenta Leonardo Antoniali, diretor geral da SEA.

Os slides das apresentações e comentários adicionais podem ser encontrados diretamente no blog da SEA: blog.seatecnologia.com.br.

Embora ainda um pouco tímido, se comparado aos eventos que ocorrem no eixo Rio-SP, o Maré de Agilidade serviu bem para mostrar que o movimento ágil está cada vez mais vivo e atuante na capital do país. Agradecemos a participação de todos, e esperamos ansiosos pelas próximas iniciativas da comunidade.



**liberdade,
produtividade,
qualidade
e agilidade
no desenvolvimento
de software**

www.seatecnologia.com.br

CLN 110 :: Bloco A :: Sala 104 :: Brasília - DF :: CEP 70774-540 :: Tel 61 3033-3355

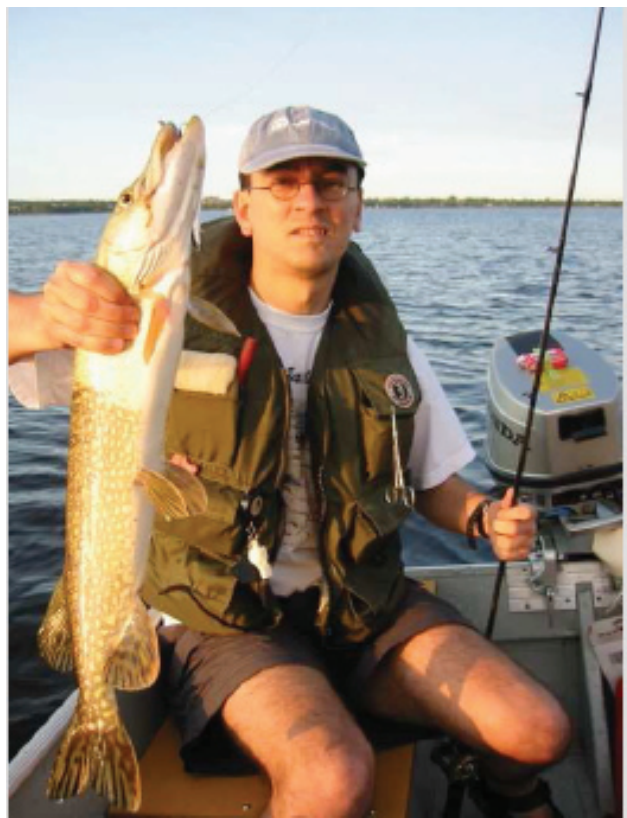




PERFIL

Entrevista com Márcio Marchini: CTO da Audaces Automação

por Victor Hugo Germano



Visão Ágil: Fale um pouco sobre sua formação e como você foi parar no Canadá?

Márcio Marchini: Entrei na Computação na UFSC em 1987. Durante o curso procurei me informar sobre o que seria mais "quente" no futuro, e o que me pareceu ser mais promissor naquela época era Orientação a Objetos; ainda desconhecida pela maioria na época - difícil de acreditar hoje em dia. Fui então trabalhar com Smalltalk no EDUGRAF, sob o Professor Luiz Fernando Bier Melgarejo.

Após a graduação fiz Mestrado em Engenharia de Produção também na UFSC (o curso de Mestrado em Computação ainda não era oferecido), onde completei minha dissertação: Lindatalk - Linda em Smalltalk. Os empregos disponíveis na região não eram interessantes - dBase, Clipper, COBOL, etc - e eu não tinha certeza se queria continuar fazendo um doutorado e virar professor.

Continuei trabalhando por mais um ano no EDUGRAF sob uma bolsa da RNP (Tadao

Takahashi), e nesse período estabeleci contatos no exterior para possível doutorado. Um deles com o Bjorn Freeman-Benson, na Carleton University em Ottawa (atualmente ele trabalha na Eclipse Foundation).

Em paralelo, estabeleci contato com a Object Technology International, Inc (OTI), pois faziam "embedded systems" em Smalltalk, e um ambiente de desenvolvimento em time para Smalltalk - o ENVY/Developer. A OTI era muito bem conceituada em comp.lang.smalltalk. Meu interesse era em tentar um estágio de 6 meses a 1 ano.

Nesse meio tempo tive um paper aceito no Supercomputing Symposium'94, em Toronto, o que me permitiu visitar tanto o Professor Bjorn Freeman-Benson quanto a OTI, ambos em Ottawa. Em conversa pessoal com o Bjorn fiquei sabendo que ele trabalhava part-time na OTI, e pretendia sair da Carleton e ir para a OTI full-time, pois admirava muito o trabalho e o ambiente lá.

...procurei me informar sobre o que seria mais "quente" no futuro, e o que me pareceu ser mais promissor naquela época era Orientação a Objetos;

Na OTI fiz duas apresentações (uma sobre Lindatalk e uma sobre Agora - um outro paper que iria apresentar na ECOOP'94) e achei o ambiente muito bacana. Muitas pessoas novas, vestidas informalmente, horários flexíveis, cada uma com o próprio escritório, respirando e suando Smalltalk.

A decisão estava clara: o doutorado ficou pra segundo plano e tratamos de estabelecer um convênio OTI-EDUGRAF. Fui trabalhar na OTI no início de 95. Seis meses viraram um ano, um ano virou dois... e eis que estou aqui em Ottawa já faz 13 anos!!! Desde então trouxemos vários outros colegas via EDUGRAF até a OTI.

VA: Como foi o trabalho na OTI? Vocês já eram agilistas naquela época?

MM: Eu estranhei no início. Não havia pessoas fazendo diagraminhas OMT ou de Booch para projetar os sistemas (UML ainda não existia, mas o mesmo se aplica). A modelagem era em Smalltalk.

A estrutura da empresa era extremamente achatada - desenvolvedores, team leaders, o CTO (Brian Barry) e o CEO (Dave Thomas). O Dave visitava as salas de todos, conversava bastante, era sempre muito simpático e visivelmente inteligente e com base sólida em computação. Todo mundo muito inteligente, um ambiente sensacional. Times tinham seus componentes, de forma que a estrutura do software refletia a estrutura da empresa em termos de modularidade.

Um dos componentes onde fui trabalhar (ENVY/Swapper - o equivalente do Serialization do Java) tinha uma bateria de testes de aceitação e regressão. Idem pra máquina vir-

sistema de controle de versão (ENVY/Developer). Coisas que simplesmente não se ensinavam no curso e não se usava no mercado em geral. Lembre-se que estamos falando de

...comparado com o movimento ágil moderno, algumas práticas estavam faltando (time boxes menores, retrospectivas de forma integral, etc).

um período pré-Google, pré-Alta Vista, pré-Java (embora que na ECOOP'94 já havia gente começando a usar Java) e pré-Open Source.

Não existia na OTI o "Big Design Up Front". No entanto, era uma empresa que "bebia o próprio champanha" como se diz por aqui - o sistema de e-mail era escrito em Smalltalk, o sistema de bug report era escrito em Smalltalk, e assim por diante.

O Dave chamava a forma de desenvolver software na OTI de JITS - Just In Time Software. Segundo ele, altamente inspirado em Just-In-Time (agora chamado Lean), e em engenharia de produtos. O lema da empresa era "Engineering Ideas Into Products" e realmente era isso.

Os times tinham bastante autonomia, e a coisa simplesmente funcionava. Talento atraindo talento, de tal forma que tinha gente muito esperta pra todo lado. Uma das perguntas na entrevista era "O que é um interpretador meta-circular?" (coisa que quem estudou Structure and Interpretation of Computer Programs saberá responder).

Uma das perguntas na entrevista era "O que é um interpretador meta-circular?"

tual, etc. Todo o histórico do código estava no

Dave, tendo sido professor na Carleton, sempre valorizou muito uma boa base em fundamentos em Computação. Mas comparado com o movimento ágil moderno, algumas práticas estavam faltando (time boxes menores, retrospectivas de forma integral, etc).

Mesmo assim, estava muito além do que era praticado nos demais locais. Inocentemente eu achei que era assim em todo o Canadá. Só com o tempo e mudança de emprego fui descobrir que a OTI era um oásis.

VA: Como foi a mudança: de desenvolvedor para consultor em Desenvolvimento Ágil?

MM: Foi gradual. Na Bedarra Research Labs (onde trabalho atualmente - fundada pelos mesmos Dave Thomas e Brian Barry) inicialmente trabalhávamos em "outsourcing de pesquisa" (um exemplo foi o Open Augment). Em alguns casos, foram pesquisa e desenvolvimento pra empresas do setor privado, que precisavam desenvolver algo de forma ágil e sabiam que não conseguiriam fazê-lo "in-house".

Como o Dave é palestrante frequente em eventos tipo ECOOP, OOPSLA, JAoo etc, muitas empresas acabam contactando-o, solicitando ajuda para melhorar o desenvolvimento de software nessas respectivas empresas.

Com a compra da OTI pela IBM e o sucesso de IBM/Smalltalk, depois VisualAge for Java

e posteriormente Eclipse (todos originados na OTI - agora IBM OTI Labs) o "segredo OTI" ficou mais conhecido, e diversas empresas tentaram adquirir o know-how através do Dave. Nessas empreitadas o Dave passou a dar consultorias nessa área, e eu como "fiel escudeiro" do Dave acabei caindo pra esse lado também.

VA: Como é o seu trabalho ao lado de grandes nomes como Dave Thomas, Uncle Bob e Michel Feathers?

MM: A parceria Bedarra - Object Mentor é relativamente recente (desde 2005). Atualmente Dave Thomas é também diretor na Object Mentor (empresa do Bob Martin, e onde o Michael Feathers trabalha) o que nos faz trabalhar em conjunto.

Na prática, tanto a Bedarra quanto a Object Mentor trabalham muito de forma virtual - muito e-mail, muito instant messaging, muito VoIP. Em alguns casos coincide de o contrato de consultoria permitir o encontro cara-a-cara. Esse ano tanto o Bob Martin quanto o Michael Feathers estiveram aqui em Ottawa em um grande contrato de consultoria nosso, e tivemos a oportunidade de nos encontrar "de verdade". No geral o Bob Martin e o Michael Feathers viajam muito, pois são muito procurados.

Ultimamente a atividade maior tem sido em torno do material "Blended TDD/Java" e

O movimento Open Source tornou várias coisas mais fáceis nesses últimos anos. Uma empresa que não usa um sistema de controle de versões tipo CVS, Subversion, git etc sabe que está vergonhosamente atrasada.

...Scrum está tendo uma aceitação bastante grande nas empresas, pois virou marca reconhecida...

"Reclaiming C++" - dois cursos desenvolvidos em parceria entre a Object Mentor e a Bedarra, que são oferecidos a empresas onde damos consultoria. São cursos SCORM, interativos, com instrutor da Object Mentor. O hosting é feito em uma outra empresa também do Dave - a www.online-learning.com (infelizmente o site não tem material publicitário sobre esses dois cursos, pois são cursos oferecidos apenas em contratos de consultoria).

VA: Nas empresas em que tem prestado serviço, como tem sido a aceitação das práticas do Scrum, Integração Contínua, TDD e Release Engineering?

MM: O movimento Open Source tornou várias coisas mais fáceis nesses últimos anos. Uma empresa que não usa um sistema de controle de versões tipo CVS, Subversion, git etc sabe que está vergonhosamente atrasada.

Novos formandos, em geral, sabem o que é um CruiseControl, de forma que é o pessoal mais novo que geralmente sabe mais sobre essas práticas de XP. Infelizmente muitas empresas grandes têm sistemas legados, com sistemas de build internos e arcaicos, e mudar é extremamente difícil.

...a empresa precisa formar os próprios "gurus internos", fazendo o movimento crescer de baixo ao mesmo tempo, senão, a consultoria sai e os maus hábitos voltam...

Scrum está tendo uma aceitação bastante grande nas empresas, pois virou marca reconhecida. Sempre há casos de desenvolvedores "das antigas" que nos perguntam, de forma incrédula, se Scrum (ou Ágil em geral) é usado em empresas grandes, como a dele (onde estamos dando a consultoria). Nessas horas digo apenas pra irem em video.google.com e assistirem a diversas palestras onde se nota o uso de Scrum em empresas como Yahoo, Google, etc. Ou estudar a origem de Eclipse.

Infelizmente não podemos citar os nomes de empresas onde damos consultoria, mas o interesse tem sido muito grande. A maioria das empresas já têm alguma forma de Release Engineering, variando em grau de maturidade. TDD e Pair Programming têm a aceitação mais difícil, praticamente impossível a curto prazo segundo o que temos notado. "Walk before you run", como dizemos por aqui. O uso de Code Review pra cada commit de código, por exemplo, geralmente tem aceitação inicial muito mais fácil que Pair Programming. Similarmente, há técnicas para se introduzir TDD de forma mais incremental.

VA: Qual o maior desafio, na sua opinião, que tem dificultado a adoção dessas práticas nas empresas?

MM: "Old Habits Die Hard", como se diz. O centro do problema são as pessoas, e não as tecnologias envolvidas. Desenvolvedores senior tendem a querer fazer as coisas como sempre fizeram, e às vezes 10 anos de experiência significam apenas uma experiência de

1 ano repetida 10 vezes (nas sábias palavras dos Pragmatic Programmers).

Gerentes também tendem a ver em Ágil uma forma de continuar a operar da mesma forma - microgerenciamento, comando e controle. É preciso respaldo para conseguir realmente mudar um grupo ou empresa.

Um nome como Dave Thomas ajuda, mas na prática o fato de acreditar na adoção tem que vir de cima também, e a empresa precisa formar os próprios "gurus internos", fazendo o movimento crescer de baixo ao mesmo tempo, senão, a consultoria sai e os maus hábitos voltam. É necessária uma massa crítica mínima que crie momento e inércia suficiente para mudar a longo prazo. Senão, a coisa degrading e implode a curto ou médio prazo.

VA: Qual o maior benefício que seus clientes enxergam nessa abordagem?

MM: O benefício imediato é poderem medir qualidade mais facilmente. Uma das primeiras coisas que implementamos é o que chamamos de CTIP - Continuous Test and Integration Platform. Criamos o CTIP em 2003, e ele inclui uma série de componentes Open Source combinados com 2 exemplos bem simples de projeto que mostram como implantar build contínuo com análise estática de código, testes de unidade, testes de aceitação, todos instrumentados e gerando relatórios de cobertura de código e qualidade. Fica transparente de imediato a qualidade do código existente, e o quanto a empresa realmente valoriza a qualidade - Há testes automatizados? Qual a cobertura deles? 10% ???

Mostramos os exemplos operando no CTIP e fica fácil ilustrar: "é aqui que vocês têm que

chegar como primeiro passo". Chegamos ao extremo de criar um InstallShield que instala CTIP automaticamente (versão cliente/desenvolvedor e versão servidor), o que significa que não há desculpa pra não adotar essas práticas facilmente.

Em seguida vem a forma de organizar os times de maneira mais modular, separar o software em componentes com "boundaries" bem definidas via APIs explícitas (API First Design) e organizar o trabalho dos mesmos de forma mais clara, com backlogs e assim por diante.

Planejar o que fazer e como fazer torna-se então muito mais concreto. Como me repetiu o Dave algumas vezes, "You Ship Your Organization".

...parece que foi outro dia que o Dave me mandou um email me falando que estava indo viajar pra uma reunião sobre o movimento Ágil que estava sendo formalizado com um grupo de conhecidos...

VA: De maneira geral, como você enxerga a adoção de práticas Ágeis no mercado de software fora do Brasil? Agile já é Mainstream?

MM: Parece que foi outro dia que o Dave me mandou um email me falando que estava indo viajar pra uma reunião sobre o movimento Ágil que estava sendo formalizado com um grupo de conhecidos. Agora, só ouço

falar em Scrum e coisas similares. Da mesma forma que Orientação a Objetos era a "coisa quente do futuro" e já virou coisa batida, o mesmo está acontecendo com o movimento Ágil.

Conte a quantidade de livros, de empresas dando consultorias, etc e irás notar que já está sendo adotado de forma consistente. Infelizmente, como se diz por aqui, "The devil is in the details". Da mesma forma que várias empresas buscaram em OO a solução milagrosa, o mesmo vai acontecer com Desenvolvimento Ágil. Lembrem-se do Fred Brooks: No Silver Bullet.

VA: Você está voltando para o Brasil, quais são os planos para o futuro?

MM: No lado familiar (motivador da mudança), espero poder conviver bastante com as nossas famílias e ver nossos filhos se divertindo muito convivendo com os primos, tios e avós.

No lado profissional, aceitei a oferta de assumir o papel de CTO na Audaces. Conheço os fundadores da empresa pessoalmente (fizeram Computação comigo) e admiro o sucesso deles nos dois pilares básicos: 1) construir o software; 2) vender o software. Como diz o Dave, a maioria das empresas falha em pelo menos uma dessas. No entanto, a Audaces já

tem mais de dez anos no mercado e precisa continuar inovando, continuar melhorando internamente também. As práticas do passado não necessariamente fazem sentido no mundo globalizado e competitivo atual.

Felizmente a semente de Ágil já foi plantada através de um contrato de consultoria com a Bedarra há mais de um ano. As melhorias, segundo os próprios funcionários envolvidos, têm sido enormes. E isso foi apenas o início.

Nosso objetivo é implementar na Audaces um padrão de excelência comparável com a OTI. Um ambiente do qual

você não sinta a menor vontade de sair, por estar aprendendo muito, sempre, estar construindo produtos de verdade, de estar muito à frente do que as demais empresas fazem em termos de desenvolvimento de software, tanto no Brasil quanto no exterior. É a nossa meta.

...Orientação a Objetos era a "coisa quente do futuro" e já virou coisa batida, o mesmo está acontecendo com o movimento Ágil...

..."The devil is in the details". Da mesma forma que várias empresas buscaram em OO a solução milagrosa, o mesmo vai acontecer com Desenvolvimento Ágil.

REFERÊNCIAS ÁGEIS

Conheça alguns sites e blogs indispensáveis para o mundo Ágil



Jeff Sutherland

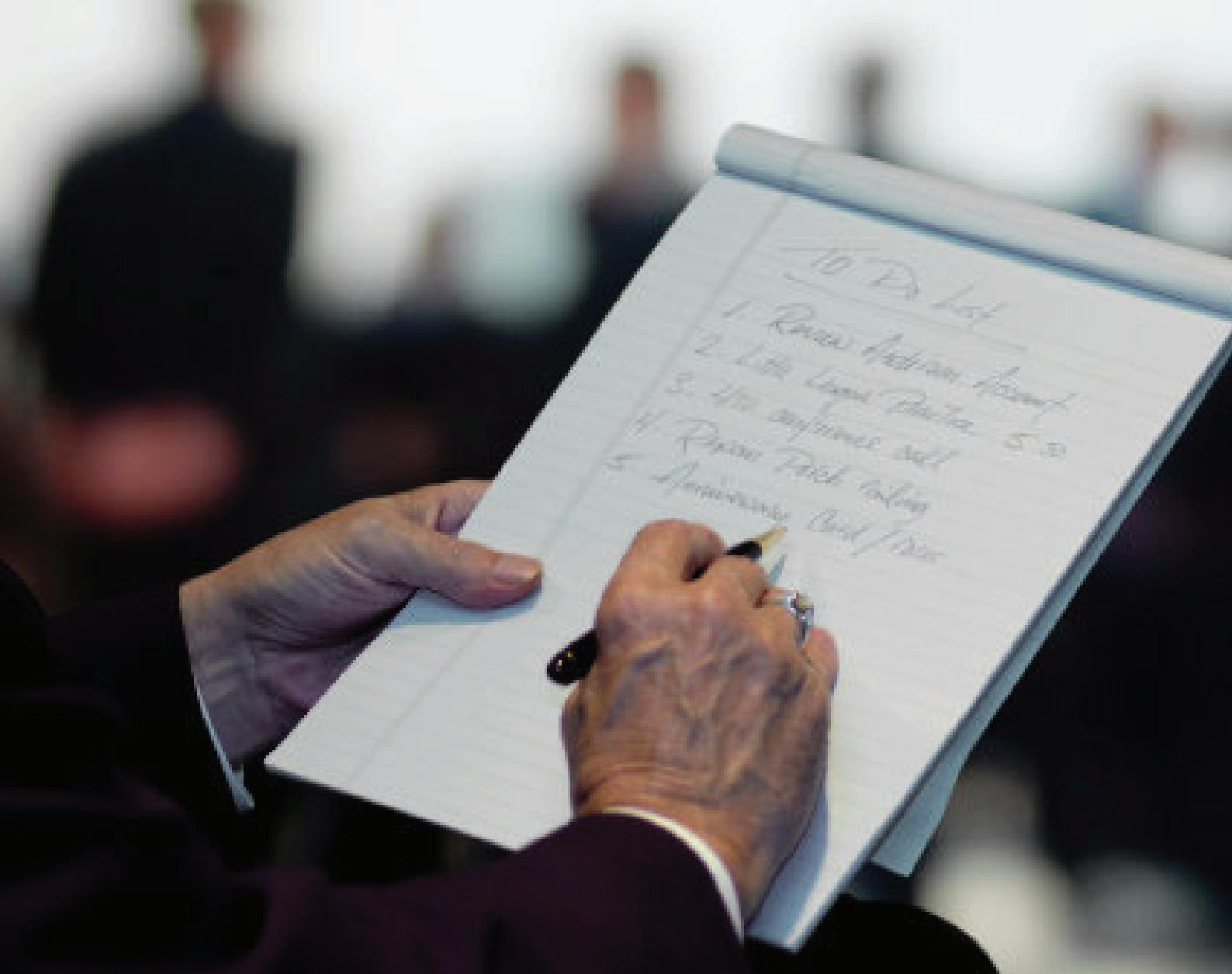
Além de ser co-criador do Scrum é um apaixonado por tecnologia e negócios, que constantemente compartilha várias ótimos artigos sobre Metodologias Ágeis, Scrum e afins.

Link: <http://jeffsutherland.com>

Um novo canal brasileiro sobre agilidade que está “bombando” é o Blog Visão Ágil, que é um blog colaborativo e conta com alguns dos mais importantes autores e líderes da comunidade ágil aqui no Brasil.

Link: <http://visaoagil.wordpress.com>





Criando um Backlog Orientado ao Negócio

Manoel Pimentel Medeiros (CSP)

É Engenheiro de Software, com 15 anos na área de TI, atualmente trabalha como Agile Coach para importantes empresas da área de serviço, indústria e bancária.

É Diretor Editorial da Revista Visão Ágil e da InfoQ Brasil, Possui as certificações CSM e CSP da Scrum Alliance e foi um dos pioneiros na utilização e divulgação de métodos ágeis no Brasil.

Já escreveu para importantes portais e revistas (nacionais e internacionais) ligados ao desenvolvimento de software.



Introdução

Sabemos que a metodologia Scrum têm um ótimo framework com um conjunto de práticas para o gerenciamento ágil de projetos, uma dessas práticas, na verdade é representado por um artefato chamado Product Backlog, que contém a lista de funcionalidades priorizadas que o Product Owner (O cliente ou um proxy de vários clientes) deseja obter em um determinado projeto de desenvolvimento de software.

Porém, o Scrum não oferece práticas orientadas à engenharia de software, ou seja, diferente da Extreme Programming e da FDD, Scrum não sugere práticas de desenvolvimento como por exemplo: modelagem, testes, inspeção, integração contínua, refatoração, etc.

Dessa forma, a equipe que está usando o Scrum num projeto, fica livre para decidir como serão tratadas essas questões inerentes a engenharia, por isso, esse artigo visa explicar uma interessante combinação entre Scrum para a parte gerencial e a FDD (Feature-Driven Development) com o foco em questões de engenharia de requisitos, portanto, você verá exatamente como melhorar o uso de seus Product Backlogs através da estrutura da FBS (Feature Breakdown Structure) proposta pela FDD.

Objetivos

É importante ficar bem claro que essa é uma forma para que possamos orientar ainda mais nosso Product Backlog, para a estrutura corporativa que o escopo atenderá dentro da empresa cliente do software.

Outro ponto importante, é que dessa maneira, poderemos aplicar algumas outras ferramentas de acompanhamento propostas pela FDD, como por exemplo o gráfico de Parking Lot, que mostra exatamente o progresso de quanto do escopo já foi concluído e quando desse mesmo escopo ainda está pendente, inclusive, nesse artigo, temos um tópico chamado “Acompanhamento do progresso com base na FBS” que aborda exatamente o funcionamento dessa ferramenta.

A idéia central da FBS

Vamos agora entender um pouco o que é e como funciona a estrutura da FBS, porém, primeiramente vamos tentar alinhar que se tra-



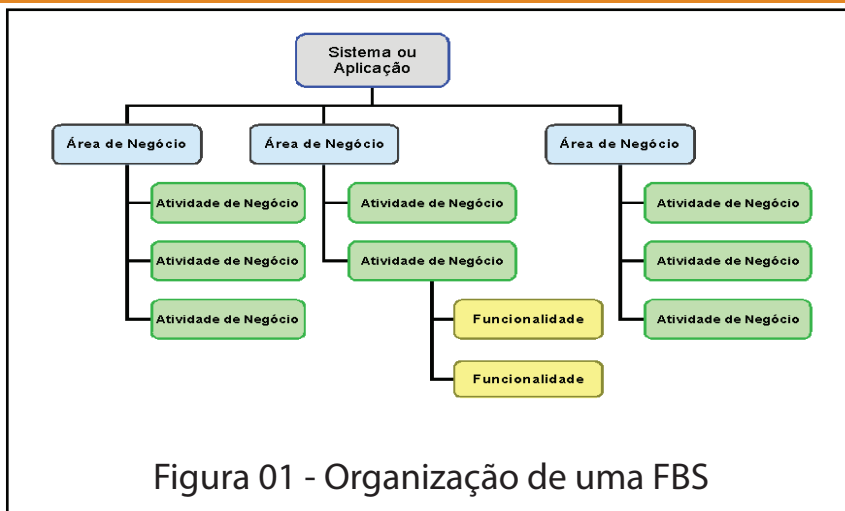


Figura 01 - Organização de uma FBS

duzirmos Feature Breakdown Structure para o português, teremos algo como Estrutura Analítica de Funcionalidades (ou características).

É importante observar que a ideia central da FBS, é a utilização das Features (Funcionalidades), afinal estamos falando da FDD (Desenvolvimento Orientado a Funcionalidades), portanto, cada feature, poderá representar um item do Product Backlog.

Veja também que uma Feature, deverá:

- Ser pequena o suficiente para ser implementada no máximo em 1 (um) Sprint.

- Ser algo que oferece valor para o cliente.
- Mapear passos em uma atividade de negócio.
- Pode ser um passo de um caso de uso.
- Às vezes pode ser o próprio caso de uso.
- E pode representar uma user story (Estória do Usuário).

Uma feature poderá ser construída usando o modelo A.R.O, que representa <Ação> <Resultado> <Objeto>, ou seja, uma determinada Ação que terá um Resultado no negócio e que envolverá alguns Objetos que rep-

Área	Atividade	Item	Business Value
Sec. Acadêmica	Gerenciamento de cursos	Controlar os cursos disponíveis pela instituição	100
Sec. Acadêmica	Gerenciamento de cursos	Definir as vagas disponíveis por cursos	90
Sec. Acadêmica	Vestibular	Realizar a Inscrição de candidatos ao vestibular	80
Sec. Acadêmica	Finanças	Emitir boleto para pagamento taxas de inscrição ao vestibular	70
Sec. Acadêmica	Vestibular	Fornecer o controle de resultados e aprovações do vestibular	60
Sec. Acadêmica	Finanças	Emitir boleto para pagamento da matrícula ao curso que o candidato foi aprovado	50
Sec. Acadêmica	Vestibular	Realizar a matrícula em um curso por um novo aluno aprovado pelo vestibular.	40

Figura 02:
Exemplo de um Product Backlog orientado à visão da FBS

resentem entidades do negócio. Veja abaixo, alguns bons exemplos de Features que seguem o modelo ARO:

- Calcular o total de uma venda
- Autorizar uma transação com cartão de um cliente
- Enviar uma nota fiscal para um cliente

Agrupando por áreas de negócio

Como você poderá observar na Figura 01, uma outra característica da FBS, é poder agrupar um conjunto de funcionalidades de uma aplicação por Áreas e Atividades de negócio.

Na verdade as áreas e funcionalidades são classificações muito comuns dentro das organizações, as áreas poderão representar um setor ou departamento de um negócio, por exemplo: Num sistema para uma faculdade, podemos ter uma Área chamada “Secretaria Acadêmica”. Já as atividades, representam um conjunto de processos em alto nível que

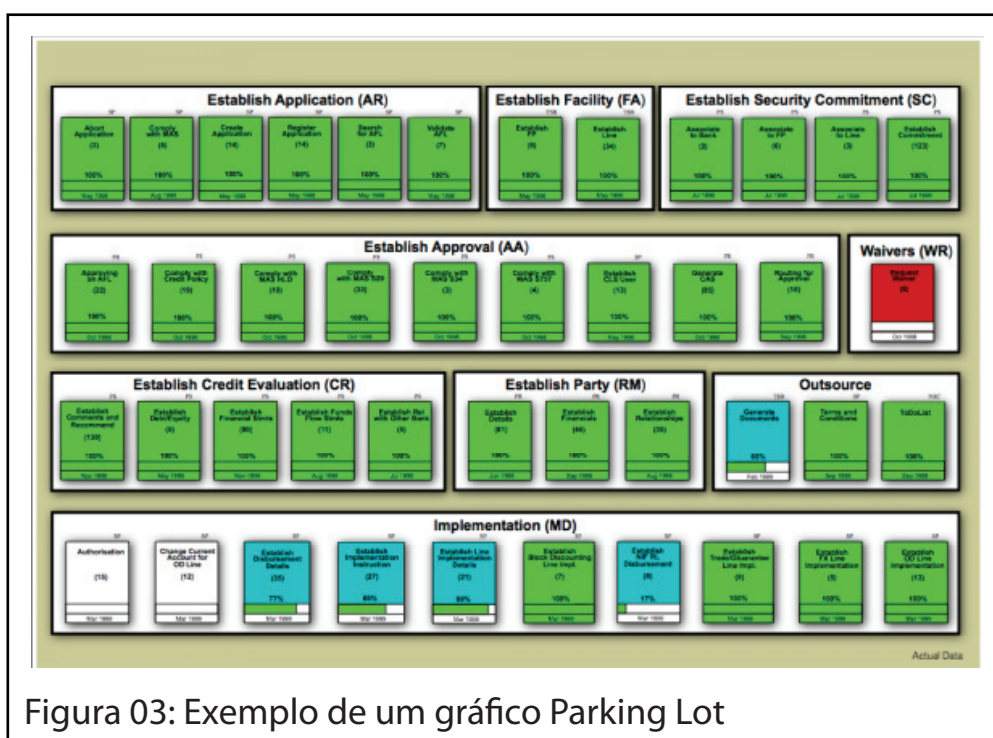
é executado por aquela área, por exemplo, uma Secretaria Acadêmica, é responsável por Matricular Alunos(as), e para que essa atividade aconteça de fato, será necessário ter um conjunto de funcionalidades de negócio como exemplo: “Realizar a Inscrição de candidatos ao vestibular” ou “Emitir boleto para pagamento taxas de inscrição ao vestibular”.

Aplicando a FBS dentro das fases do Scrum

Na prática, a FBS poderá ser combinada com a estrutura de Product Backlog do Scrum, mas além disso é importante observar outras práticas da FDD aderentes ao funcionamento do Scrum.

Lembre-se que no Scrum, podemos ter uma macro-fase chamada PreGame, que atua exatamente no início de um projeto, absorvendo algumas importantes atividades como por exemplo:

- Criação uma visão inicial do produto
- Iniciação de um modelo de negócios abrangente



- Início de uma arquitetura em alto-nível

Dessa forma, podemos perfeitamente integrar esse conceito de PreGame do Scrum com a prática chamada “Desenvolver um modelo abrangente” da FDD, pois assim, podemos obter uma visão inicial em alto-nível do escopo da aplicação.

O Product Backlog de acordo com a estrutura da FBS

Conforme o objetivo desse artigo, você agora irá observar como de fato podemos combinar a estrutura de FBS, com a semântica de um Product Backlog.

Lembre-se que por natureza, o Product Backlog conterà a lista priorizada de funcionalidades fornecidas pelo Product Owner, dessa forma, é possível aplicar a cada funcionalidade contida no Product Backlog, a visão de “Área” e “Atividades” proposta pela a FBS. Portanto, observe na Figura 02 um bom exemplo de como essa integração poderá acontecer num típico projeto de desenvolvimento de uma aplicação acadêmica.

Acompanhamento do progresso com base na FBS

Veja na Figura 03, que outro ponto importante que favorece o uso da FBS num projeto de desenvolvimento, é que sua estrutura de Áreas, Atividades e Funcionalidades, fornece a opção de usarmos a ferramenta gráfica de acompanhamento da FDD chamada Parking Lot, que para demonstrar o avanço de uma determinada Funcionalidade, utiliza alguns marcos como:

- Estudo Dirigido Sobre o Domínio
- Desenho (Projeto)

- Inspeção do Desenho

- Codificação

- Inspeção do Código

- Promoção ao Build

Esses marcos fornecerão uma visão de forma detalhada ou sintética (aglutinada) do progresso na conclusão de uma Funcionalidade, que irá totalizar diretamente para o progresso de uma Atividade inteira e essa por sua vez, irá totalizar para o progresso de uma Área inteira.

Conclusões

Estou certo de que você percebeu como essa possibilidade de combinação entre essas duas metodologias, pode potencializar o entendimento e a proximidade ao mundo real de nosso processo de engenharia de requisitos baseado em idéias e práticas ágeis. Portanto convido você a experimentar essa combinação de idéias do Scrum com algumas outras idéias da FDD e verificar na prática, se realmente ela irá trazer resultados positivos dentro de seus projetos.

Referências

<http://www.heptagon.com.br> - Principal divulgador e pioneiro na utilização da FDD no Brasil

<http://www.featuredrivendevelopment.com> - Site oficial sobre a FDD

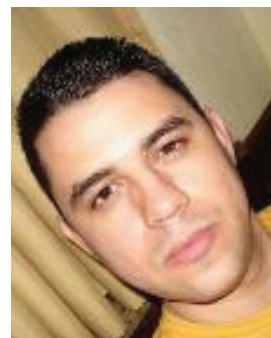
<http://www.scrumalliance.org> - Site da instituição Scrum Alliance

OS 7 PECADOS CAPITIAIS DE TIMES ÁGEIS

por Emerson Macedo



Emerson Macedo é Consultor Sênior da Concrete Solutions (<http://www.concretolutions.com.br>). Atualmente trabalha em tempo integral na globo.com. Trabalhou também em empresas de seguros, bancos, telecomunicações e internet. É também professor dos cursos de extensão em linguagens de programação das Faculdades Simonsen. Mantém seu blog sobre desenvolvimento de software em <http://codificando.com>



De alguns anos pra cá, houve um certo boom na adoção de metodologias ágeis[1] em todo mundo. Aqui no Brasil, como sempre, as coisas demoram um pouco a acontecer, mas parece que neste ano estão realmente começando a acontecer.

Em meio a toda essa empolgação, é comum que aconteçam alguns equívocos e também algumas coisas não saiam como esperado. Portanto, existem alguns erros que devem ser evitados, ou quando percebidos, banidos o quanto antes. Baseado na minha experiência e na de alguns amigos e colegas que trabalham em empresas ou equipes ágeis, enu-

merei 7 desses erros, os quais considero os mais críticos para a saúde de um time ágil. São eles:

1 - Engessar a Metodologia Ágil

Por mais incrível que pareça, muita gente acha que as iterações/sprints do SCRUM[2] devem ser sempre de 2 semanas. Existem alguns casos em que as prioridades do Product Owner[3] são tão voláteis que adotar um Sprint de 1 semana ao invés de 2 seria muito mais adequado. Isso é apenas um exemplo. Existem diversos. O problema maior é querer colocar o SCRUM/XP/Whatever acima de tudo.

Isso vai totalmente contra um dos itens do manifesto ágil que diz: "Indivíduos e interações são mais importante do que processos e ferramentas". Já foi dito por algumas pessoas que o SCRUM é um bom guideline, mas nem ele é solução pra tudo. Podemos perfeitamente adaptá-lo, descartando alguma coisa que não está nos servindo no nosso contexto ou adicionando alguma coisa que nos possa ser útil.

2 - Cobertura de testes inadequada ou inexistente



Certa vez, comentei em meu blog pessoal[5] sobre o drama que passei ao ter que adicionar uma funcionalidade num sistema sem uma boa cobertura de testes automatizados. É difícil ser ágil, quando não se consegue alterar os códigos do projeto também de forma rápida/ágil. Isso faz com que uma funcionalidade que pareça ser relativamente simples, demore bastante a ser implementada, justamente pela falta de confiança no código existente. Sem contar que perde-se muito tempo fazendo e re-fazendo testes manualmente, os quais ficam muito mais sujeitos a falhas do que se estivessem automatizados. É bem verdade que conseguir uma cobertura de testes na casa dos 100% é um pouco utópico, pois existem alguns testes que o custo de automatizar não compensa. Mas mesmo assim é fundamental que o time automatize a maioria de seus testes.

3 - SCRUM Master - O chefe

Essa é bem polêmica, mas ao mesmo tempo é comum demais nos times ágeis, principalmente no começo da adoção de uma metodologia ágil. Faça as seguintes perguntas à si mesmo: Quando vou me atrasar, qual a primeira pessoa que penso em telefonar? Quando tenho que ir ao médico, pra quem dou satisfação? O SCRUM Master do meu projeto é um facilitador para o time ou ele frequentemente toma decisões técnicas de projeto?

Muitas vezes, o SCRUM Master era algum Gerente ou Coordenador que ainda não se acostumou a deixar de controlar o time. Em outros casos, o time parece ainda não ter percebido que a filosofia ágil incentiva times auto-gerenciáveis. É claro que o SCRUM Master deve ser notificado, mas os mais interessados em



saber da ausência de um membro do time é o próprio time, afinal de contas, o restante dos membros precisarão suprir essa ausência de alguma forma, provavelmente cada um se esforçando um pouco mais.

4 - Ignorar o Débito Técnico

O Débito Técnico[4] de um projeto prejudica muito a velocidade do time e a qualidade daquilo que está sendo desenvolvido. Um amigo disse uma vez, que quanto mais o des-caso aumenta no que diz respeito ao débito técnico, mais o software caminha rumo ao apodrecimento[6].

5 - Na primeira dificuldade, voltar as antigas práticas

Como seres humanos, temos uma tendência natural. Quando não estamos conseguindo alcançar um objetivo de maneira imediata, partimos em busca de um porto-seguro. Especialistas dizem que uma prática leva cerca de 30 dias para se tornar um hábito. Portanto, o time deve ter paciência até que tudo entre nos eixos.

Outro problema que ocorre é que quando um time ou uma empresa começa a adotar uma

Metodologia Ágil, é normal que exista grande resistência. Geralmente surge uma forte corrente contrária, principalmente por parte de pessoas que são avessas a qualquer tipo de mudança. Essas pessoas serão as primeiras a arrumar um motivo para jogar tudo por água a baixo. A tendência do time diante dessas circunstâncias é de começar a fazer as coisas como antigamente, sem antes mesmo concluir o ciclo de alguns sprints/iterações tentando as novas práticas.

Um conselho, não desista. Muitos colegas de trabalho riam de mim quando eu comecei a estudar SCRUM e faziam gozação dos meus post-its nas paredes da minha baia, numa antiga empresa. Eu poderia simplesmente ter desistido, mas resolvi continuar e hoje vejo que foi uma ótima escolha.

6 - Entregar todas as estórias, custe o que custar

É perfeitamente normal e até compreensível que um time, ao ver uma estória[8] pendente no seu white board, se esforce ao máximo para entregá-la. Afinal de contas, compromisso é compromisso. Acontece que em muitos momentos o time trabalha fora do seu horário ou até mesmo começa uma tremenda correria para tentar terminar tudo, custe o

que custar. Esse esforço heróico geralmente leva consigo um monte de bugs no código, mais débito técnico e uma boa dor de cabeça a médio e até mesmo a curto prazo. Acredito que o problema esteja no entendimento de alguns pontos importantes da filosofia ágil.

A maioria das metodologias ágeis existentes prega que um time deve entregar para o cliente o maior valor possível no menor tempo possível. Isso significa que você não precisa necessariamente terminar todas as estórias,

até porque o comprometimento do time é com o GOAL do Sprint. É claro que entregar todas as estórias é bom, mas fazer isso sem medir consequências é besteira, pois o time acaba escondendo os reais problemas causadores da não conclusão dessas estórias. Particularmente, prefiro entregar 4 histórias com qualidade e atendendo corretamente o que foi pedido do que entregar 7 sem qualidade, sem testes automatizados e tendo que fazer hora extra junto com o restante do time.

Acredito que o melhor a fazer quando não for possível entregar todas as estórias é avaliar durante a retrospectiva os motivos reais do problema. É possível que tenha acontecido algum impedimento grave. Talvez alguma



estória que o time subestimou a complexidade e só percebeu isso durante o desenvolvimento da mesma. Talvez algum caso de aceitação foi mau escrito. Pode ser até que o time ainda esteja descobrindo sua real velocidade e por isso colocou mais pontos no sprint/iteração do que consegue realmente entregar. Entretanto, se o time conseguiu alcançar o Sprint GOAL, concluiu o Sprint com sucesso, ao contrário do pensamento comum. É importante também lembrar que estimativas não são valores exatos, mas sim valores aproximados.

7 - Confundir agilidade com pressa

Eu sinceramente acho que esse é o pior de todos. Talvez por isso ele seja o último dessa lista. É incrivelmente fácil que clientes, Product Owners e o próprio time confundam agilidade com pressa.

O problema de quem pede:

Alguns Product Owners (ou os clientes), durante a adoção de metodologias ágeis, podem começar a achar que as funcionalidades do software passarão a serem entregues num piscar de olhos. A pergunta mais clássica deles é: Por que ainda não estamos entregando as funcionalidades em uma velocidade extraordinária?

Existem algumas explicações, as quais devem ser bem observadas:

- Agora as funcionalidades começaram a ser entregues, pois antes, sequer eram.
- O time pode ainda estar se estruturando e sua velocidade aumentará gradativamente.

- A cultura anterior desconsiderava qualidade no software. O time pode estar se acostumando/aprendendo a trabalhar de forma a obter maior qualidade no que se produz. A velocidade aumentará, a medida que as práticas fundamentais estiverem arraigadas.
- Por último, o time pode já estar na sua velocidade ideal. Não adianta forçar. Assim como os servidores que rodam as aplicações da empresa não podem exceder a sua capacidade de trabalho, os profissionais também não.

Provavelmente esse time, com os membros atuais, não terá velocidade superior a que está sendo apresentada. Neste momento, o planejamento das entregas do software deve ser feito em cima dessa realidade, o que é muito melhor do que a forma antiga, onde não se tinha nenhuma idéia sobre as entregas e tentava-se fazer previsões com chutes fantásticos nos percentuais de um Gant Chart[7].

O problema de quem faz:



Os Desenvolvedores podem ficar empolgados com a ideia do agile e começar a fazer as coisas correndo. Quem fez treinamento de SCRUM deve lembrar bem do exercício das bolinhas, onde todo mundo queria correr para conseguir aumentar o número a ser entregue, como se aquilo fosse algum tipo de meta. Logo todos percebiam que essa tentativa de aumentar a velocidade causava problemas na qualidade e não era interessante.

Outro fator engraçado é quando os desenvolvedores começam a considerar que quem conclui mais post-its de tarefas está sendo mais produtivo. Isso faz com que cada um queira acabar o mais rápido possível o que está fazendo para logo pegar outra tarefa. Essa atitude é uma grande amiga dos Bugs, já que o que mais importa é passar tudo para o status de Done.

O ideal:

Eu acredito que a velocidade do time sempre

se ajusta naturalmente depois de um tempo. A única coisa relevante que eu acho que possa ser feito é tentar colocar na equipe 1 ou 2 pessoas mais experientes em práticas de engenharia que possam ajudar o time diariamente em algumas ações fundamentais para um bom desenvolvimento.

Conclusão:

É certo que tudo que escrevi aqui é apenas a minha opinião. Posso estar enganado em alguns pontos. Entretanto, se você já trabalha com Agile ou está começando a fazê-lo, provavelmente irá notar alguns desses pontos que mencionei na sua experiência diária.

Acredito que o que precisamos nos lembrar sempre é que atualmente, o fundamental é seguir os princípios do Manifesto Ágil[9], mesmo que você deixe todas essas metodologias ágéis da moda de lado e cria a sua própria, observando as necessidades da sua empresa.

Referências:

[1] - http://en.wikipedia.org/wiki/Agile_software_development

[2] - <http://en.wikipedia.org/wiki/SCRUM>

[3] - <http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms>

[4] - <http://www.martinfowler.com/bliki/TechnicalDebt.html>

[5] - <http://codificando.com/2008/09/11/codigo-do-panico/>

[6] - <http://gc.blog.br/2008/07/20/cuidando-para-que-o-software-nao-apodreca/>

[7] - http://en.wikipedia.org/wiki/Gantt_chart


[8] - http://en.wikipedia.org/wiki/User_story

[9] - <http://agilemanifesto.org/>



FDD (Feature-Driven Development)

Origem

 1997-1998, Cingapura – Contexto: Desenvolvimento de um grande sistema de empréstimos para um banco internacional

O que é uma Feature (Funcionalidade) ?

Ser pequena o suficiente para ser implementada no máximo em 1 (uma) Sprint.

Ser algo que de valor agregado para o cliente.

Mapear passos em uma atividade de negócio.

E pode representar uma user story (Estória do Usuário).

Uma feature poderá ser construída usando o modelo **A.R.O.**, que representa **<Ação> <Resultado> <Objeto>**, ou seja, uma determinada **Ação** que terá um **Resultado** no negócio e que envolverá alguns **Objetos** que representem entidades do negócio.

Exemplos de Features com o modelo ARO:

Calcular o total de uma venda

Autorizar uma transação com cartão de um cliente

Processos

Concepção e Planejamento



Requisitos



Construção



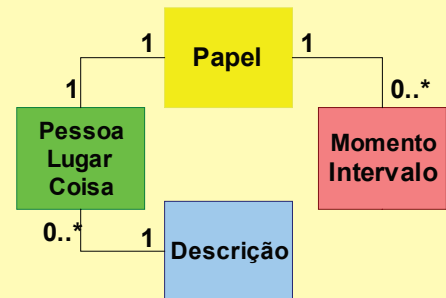
Pacotes de Trabalho (Iterações)

Incremento de Software

Práticas

- Modelagem de objetos do domínio
- Desenvolvimento por feature
- Responsabilidade individual de classe (código)
- Equipes de features
- Inspeções
- Builds regulares
- Gerenciamento de configuração
- Relatório/visibilidade de resultados

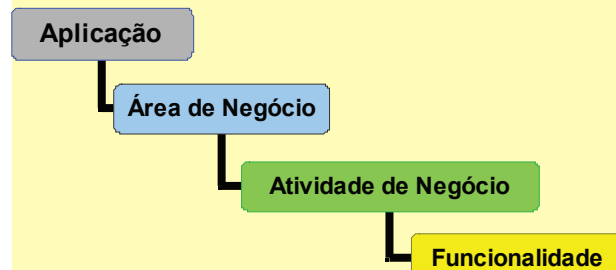
Ferramentas



UML em Cores

Migration Tool			
[21-6] ReadOnly Merge (3/3) 100% 2006/12/15	[21-7] ProgressBar (read) (0/1) 50% 2007/1/31	[21-6] ProgressBar (write) (0/1) 0% 2007/1/31	[21-8] Merge Utility (0/1) 0% 2007/1/31

Parking Lot



FBS - Feature Breakdown Structure

Referências

<http://www.heptagon.com.br> - Principal divulgador e pioneiro na utilização da FDD no Brasil

<http://www.featuredrivendevelopment.com> - Site oficial sobre a FDD

Agile Page - Organização:





Revista

Visão Ágil

INFORMAÇÕES



EQUIPE VISÃO ÁGIL

EDITOR CHEFE

Manoel Pimentel - manoel@visaoagil.com

EDITORAÇÃO E ARTE

Felipe Rodrigues - felipero@gmail.com

Victor Hugo Germano - victorhg@gmail.com



JUNTE-SE A NÓS!

Estamos atrás de pessoas dispostas a construir um canal realmente atraente na divulgação de métodos ágeis. Gotou da idéia? Você pode escrever artigos, revisar matérias ou editar a revista!



PUBLICIDADE

Se você está interessado em fazer alguma ação de marketing em parceria com a Revista Visão Ágil, sinta-se livre para entrar em contato.

Vamos trabalhar juntos!!

CANAIS VISÃO ÁGIL

Site: <http://www.visaoagil.com>

Blog: <http://visaoagil.wordpress.com>



ATENDIMENTO AO LEITOR

emails:

manoel@visaoagil.com / comunidade@fratech.net