



**As 5 Doenças do
Gerenciamento de Projetos**
Causa nº 3: Síndrome do Estudante

**Utilizando Metodologias
Ágeis para
atingir MPS.BR nível F
na Powerlogic**

Agile e MPS.BR

Case de implantação de processos ágeis com MPS.BR

Métricas e o Desenvolvimento Ágil

**Aperfeiçoamento de
Projetos Ágeis**

OpenUP: Uma Introdução

**Brincando com a UML
em Cores**

Cobertura do Java Brasil 2007

Cobertura do Scrum Gathering em Londres

News e Referências Ágeis

Sprint Backlog

Conheça os temas selecionados para essa edição(iteração).



News
Página 04

As 5 Doenças do Gerenciamento de Projetos
Causa n° 3: Síndrome do Estudante
Página 05



Humor de Projetos
Página 08

Referências Ágeis
Página 09



Utilizando Metodologias Ágeis para atingir MPS.BR nível F na Powerlogic.
Página 10

Métricas e o Desenvolvimento Ágil.
Página 18



OpenUP: Uma Introdução
Página 25

Aperfeiçoamento de Projetos Ágeis
Página 29



Cobertura do Java Brasil 2007
Página 35

Brincando com a UML em Cores
Página 40



Cobertura London Scrum Gathering
Página 44

Editorial

Equipe Visão Ágil

Diretor Editorial:

Manoel Pimentel Medeiros

Diretor de Marketing:

Alexandre Magno Figueiredo

Assessoria Administrativa:

Manuella Bulcão Medeiros

Atendimento ao leitor

e-mail: manoelp@gmail.com
site: www.visaoagil.com

Edições Anteriores

Qualquer edição anterior pode ser baixada gratuitamente no site www.visaoagil.com

Artigos

Se você está interessado em ver seus artigos sobre práticas adágis publicados em nossa revista, por favor envie-os para manoelp@gmail.com para que possamos apreciá-los. Sua colaboração é sempre bem vinda.

Publicidade

Se você está interessado em fazer alguma ação de marketing em parceria da Revista Visão Ágil, entre contato conosco através do e-mail axmago@gmail.com, e teremos o maior prazer em trabalharmos juntos.

Olá caro amigo agilista, essa terceira edição é muito importante para nós, primeiro por comemorar mais uma edição de nossa revista, e segundo por nos possibilitar celebrar junto a você esse novo que está começando e que será muito importante e agitado para toda a comunidade agile no Brasil.

Para realmente criar uma edição especial, decidimos fazer um trabalho bem diferente das edições passadas, dessa forma, você amigo leitor, terá acesso a um número maior de artigos, com conteúdos inéditos, ricos e atualizados.

Nesta edição, para realmente implementar esse conceito, estamos trazendo vários temas como: Gestão de Projetos, Modelagem com UML em Cores, Métricas Ágeis, Open-UP, Aperfeiçoamento de projetos ágeis, cobertura do evento Java Brasil 2007 e até do mais importante evento internacional de Scrum, o Scrum Gathering de Londres.

Para abrillantar ainda mais esta edição, temos um completo case sobre a implantação de processos ágeis para obtenção da certificação MPS-BR na empresa PowerLogic, além claro, das já tradicionais seções de News, Referências Ágeis e Humor de Projetos.

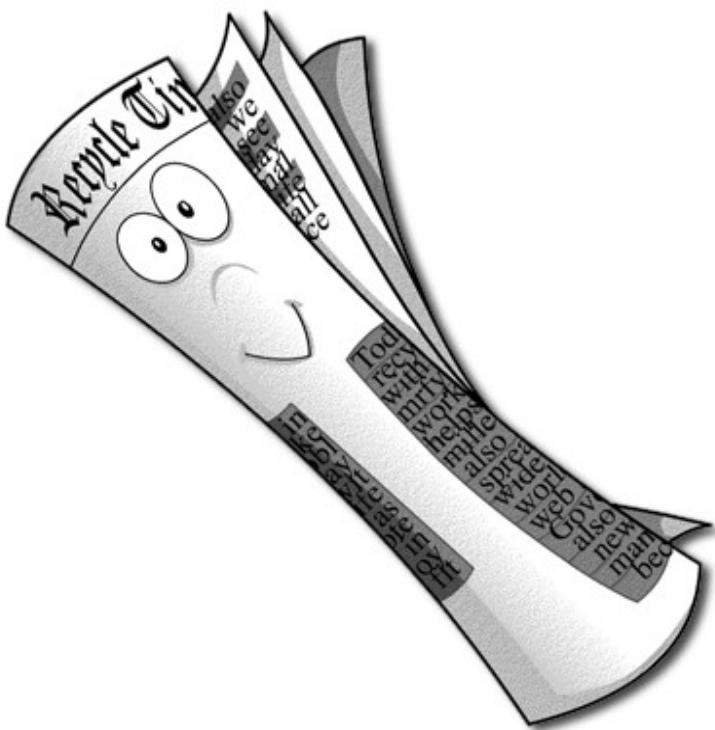
Portanto caro amigo, esperamos que você goste de mais esse nosso trabalho e, principalmente, queremos que saiba que a Revista Visão Ágil é um espaço para a participação e evolução de toda a comunidade agile no Brasil.

Paz e sucesso.

Manoel Pimentel Medeiros

Diretor Editorial





News

Veja aqui as novidades
do mundo ágil

Treinamento Gerenciamento de Projetos
de Software com Scrum, com
Alexandre Magno

agenda para fevereiro e março:
São Paulo: www.caelum.com.br
Porto Alegre: www.aquasoft.com.br

Workshop
AGILE IMMERSION

em março de 2008
www.fratech.net

FISL 9.0 - Fórum Internacional
Software Livre em Porto Alegre

em abril de 2008
<http://fisl.softwarelivre.org/9.0>

Workshop de Gestão e Desenvolvimento
Ágil com FDD em Florianópolis-SC

em março de 2008
www.heptagon.com.br

CHICAGO SCRUM
GATHERING

em abril de 2008
www.scrumalliance.org

Workshop
Modelagem Ágil com
UML em cores e DDD

em março de 2008
www.fratech.net

As 5 Doenças do Gerenciamento de Projetos



Causa nº 3:
Síndrome do Estudante



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, 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.

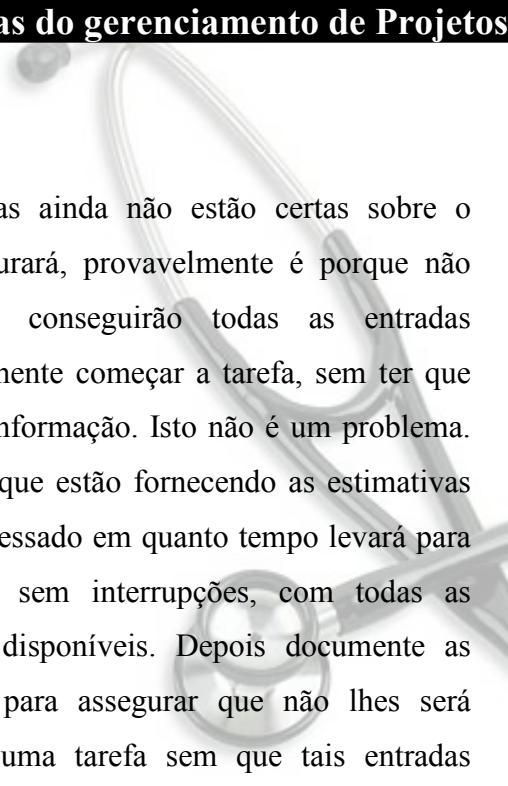
Autor: Allan Elder

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.

A Síndrome do Estudante também é conhecida como procrastinação. A grande diferença é a razão para adiar o trabalho. Procrastinar é ser preguiçoso ou irresponsável. A Síndrome do Estudante é um mecanismo de defesa natural. Significa adiar o trabalho até o último momento possível, não porque somos preguiçosos, pelo contrário, estamos trabalhando duro. Todos caímos presas da Síndrome do Estudante ocasionalmente. Ela ganhou esse nome pela forma como os estudantes lidam com o dever de casa. Imagine seu professor dizendo que você tem uma prova final em 19 semanas. Ele lhe dá todo o material, o livro, os objetivos que serão testados e a data. Quando você começa a estudar? Na véspera da prova. Por quê? Você tem tempo, é por isso. Outras tarefas pressionam mais e, portanto, você atrasa o início de uma tarefa até o último momento, para lhe dar tempo de completar outro trabalho, muito provavelmente também sendo feito no último momento.

Geralmente as pessoas dizem que não podem estimar com qualquer precisão o quanto uma tarefa irá durar. Eu discordo. A evidência para esta afirmação é que as pessoas realmente sabem o último minuto possível que elas têm para iniciar uma tarefa, ou arriscar atrasar. Quando foi a última vez que você adiou algo até o último minuto e foi capaz de escolher o último minuto real? Você trabalhou a noite toda para completar a tarefa e imprimiu o resultado exatamente antes da grande reunião. O papel ainda está morno na entrega, mas você conseguiu. Esse problema torna-se mais sério quando consideramos as implicações sobre a qualidade. Sim, você escolheu o último momento possível para completar uma tarefa, mas qual foi a consequência em potencial sobre a qualidade? Se algo desse errado não haveria tempo para consertar. Com qual frequência você erra a escolha do último minuto em tarefas importantes? Eu sugiro que é raro. Eu sei disso por experiência própria. Portanto, podemos concluir que a maioria das pessoas realmente sabem o quanto uma tarefa irá levar (dada alguma probabilidade) quando elas podem se dedicar a ela. Se você quer estimativas mais precisas (mas nunca precisas), identifique a data de entrega e pergunte-se: "Se eu iniciar esta tarefa no último minuto, quando isso seria?" Agora você tem a estimativa da duração da tarefa.

A dificuldade da estimação das tarefas não é conhecer o quanto a tarefa durará, mas “chutar” quantos outros fatores devem ser considerados, já que sabemos que não nos será permitido trabalhar na tarefa sem interrupção.



Se as pessoas ainda não estão certas sobre o quanto uma tarefa durará, provavelmente é porque não sabem quando elas conseguirão todas as entradas necessárias para realmente começar a tarefa, sem ter que parar e coletar mais informação. Isto não é um problema. Lembre aos recursos que estão fornecendo as estimativas que você só está interessado em quanto tempo levará para completar as tarefas sem interrupções, com todas as entradas necessárias disponíveis. Depois documente as entradas necessárias para assegurar que não lhes será pedido que iniciem uma tarefa sem que tais entradas estejam sob sua posse. Seu trabalho como gerente do projeto é garantir que eles tenham o que é necessário para realizarem o trabalho deles.

Enquanto você sobrecarregar as pessoas com tarefas e permitir que elas inflacionem os prazos para acomodar outras tarefas, você perpetuará a síndrome do estudante. Esse problema é ampliado pela multi-tarefa. Nós temos nossa equipe trabalhando em múltiplas tarefas com durações inflacionadas e esperamos que eles priorizem tais tarefas. Tarefas urgentes terão precedência sobre as tarefas importantes. Isto encoraja o embutimento de segurança nas tarefas, que fornece mais tempo do que o realmente necessário, então atrasamos o início das tarefas até que tenhamos absolutamente que começar, devido às demandas concorrentes. E o que acontece quando a tarefa realmente encontra um problema? Onde está a segurança? (Uma pergunta melhor seria: "Quando é a segurança?") Está no passado. Não podemos usá-la mais (ver **Figuras 5 e 6**). Note, no desenho, que quando embutimos a segurança nós, mentalmente, colocamos a proteção no final da tarefa. Porém, devido à síndrome do estudante, nós não começamos a tarefa imediatamente e, portanto, começamos a tarefa tarde.

Quando o problema ataca, a segurança com a qual contávamos já não está mais disponível.

Algumas pessoas enfatizam que tudo isto não pode ser verdade. Na verdade, quando minha equipe me fornece uma estimativa eles quase sempre atingem essa estimativa. Isto pode ser verdade. Entretanto, é verdade porque é uma profecia auto-realizável. A tarefa torna-se vítima das questões mencionadas anteriormente, fazendo com que seja completada como previsto, porque as pessoas querem ser vistas como confiáveis. Como você pode ver, se a tarefa é iniciada imediatamente e não há problemas sérios, a segurança embutida é consumida pela **Lei de Parkinson**. Se nós não começarmos a tarefa imediatamente e ocorre um problema, a tarefa será atrasada. Se começarmos a tarefa atrasada (síndrome do estudante) e nada dá errado, nós completamos a tarefa “**no prazo**”. De um jeito ou de outro, a segurança foi gasta. Esse é um tempo que fez com que a duração do seu projeto seja estimada muito além do que era necessário. Isso também torna seu projeto menos competitivo.



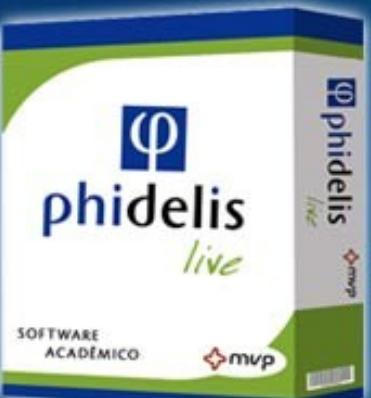
Figura 6

Humor de Projetos

Quem disse que a vida em
um projeto não é divertida?

LUMópolis

UML e Cotidiano de Projetos de forma divertida



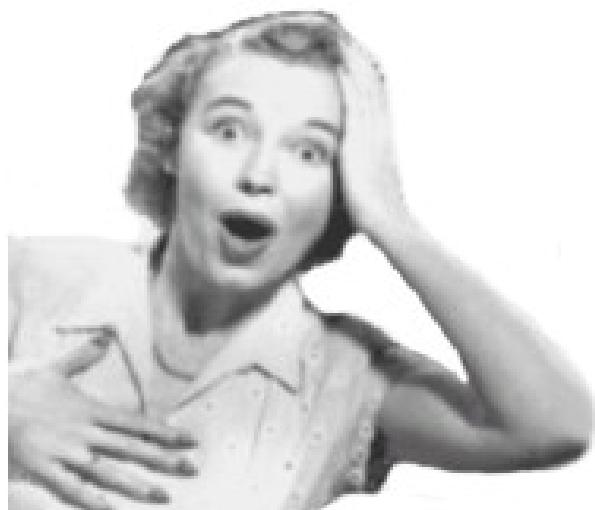
<http://www.phidelis.com.br>

Tecnologia em Gestão Acadêmica
ao alcance de todos



Referências Ágeis

Veja aqui, alguns sites e blogs indispensáveis para o mundo ágil



A screenshot of a Mozilla Firefox browser window showing Danilo Sato's personal website. The title bar reads "Danilo Sato - Mozilla Firefox". The address bar shows "http://www.dtsato.com". The main content area features a large header "Danilo Sato" and "Personal Website". Below the header is a navigation menu with links to "Home", "Studying", "Business", "Book Reviews", "Blog", "Diversity", and "Contact Me". To the right of the menu is the date "Wednesday, November 7, 2007". The left sidebar contains a section titled "Currently Reading:" with a thumbnail image of the book "Best of Ruby Quiz" by James Edward Gray. It also includes social media links for "WS*", "RSS", "Flickr", and "My shared items...". The main content area has a heading "DANILO TOSHIAKI SATO" and a bio about his education and interests in Agile software development. It lists various Agile methodologies and includes a photo of Danilo Sato smiling. Below this, there is a statement about maintaining book reviews and a blog. At the bottom, there is a link to "Inscrições abertas para o Rio on Rails 2007". The footer of the page is visible at the bottom of the browser window.

Blog de Danilo Sato, O cara é fera,
e em seu blog têm muito material sobre XP,
Scrum, Arquiteturas, Java e Ruby.
URL: www.dsato.com

XProgramming.com - an Agile Software Development Resource - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

<http://www.xprogramming.com/>

[RSS Feed](#)

Sections:

- Welcome
- What is XP?
- Articles
- Software
- Community
- Archives
- Search

The Human Interface
New Directions in Computer Design
Jef Raskin

Ron Jeffries

Welcome, from Ron Jeffries 04/28/2006

For RSS Feed see top left menu item.

Printing CSS allows full-page printouts of articles.

[Ron's blog](#), [Hot Needle of Inquiry](#), [Ron Jeffries bio](#).

Your [email](#), with site feedback or proposed articles, will be very welcome.

Finishing Story One [Ron Jeffries](#) 11/19/2007

We review cards and letters, and we finish our first elementary story. We think we will start over. Oh, and my computer is now fast again. Read on ...

Rails: The First Story [Ron Jeffries](#) 11/13/2007

We start on the first story, but get into some trouble with my machine. And you're probably wondering what we're actually up to anyway ... take a look!

Planning the Project [Ron Jeffries](#) 11/12/2007

We take a few moments to write stories and talk about what should be done. Who says we don't plan?

Recent Articles:

- Getting Off Our Tails and Onto the Rails
- Further Report Refactoring
- Ljutic Mono Gun
- Calling the Shot
- End to End, or, Another Run on the Hamster Wheel of Life
- Reloading the Shotgun Project

Extreme Programming Refactored: The Case Against XP
Matt Stephens, Doug Rosenberg

How good do you want to be, and when?
Let Ron Jeffries, experienced author, trainer, practitioner, coach, help your Agile team get to the next level! Bring your project in on time, on budget, fit for purpose!

Ron's books & publications

Calendar

2007 outlook
New Book in Process w/
"What You Need"
Limited Client Time
Available: Inquire
ronjeffries@hem.org

First Quarter 2007
Deep Agile - Boston;
West Coast Client;
Midwest Client;
Public Course
Style and Grace!

3Q 2007
Agile Conference
Boston, MA

Site do **XProgramming.com**,
mantido pelo grande
Ron Jetries, um verdadeiro
Gurú que se transformou
em uma forte
referência sobre **Extreme
Programming**
no mundo inteiro.

URL: www.xprogramming.com

Utilizando Metodologias Ágeis para atingir MPS.BR nível F na Powerlogic

**Autora: Isabella de Araujo Fonseca**

É Certified Scrum Master, atua desde 1999 como consultora em e-Business para grandes empresas utilizando JEE. Atualmente é gerente da equipe de desenvolvimento e de projetos do eCompany Portal - primeira solução de EIP (Enterprise Information Portal) – do país e do eCompany Process - solução de definição e controle de Processos Corporativos integrada ao Gerenciamento de Projetos (EPM e APM), de Requisitos e de Produtos (ALM).

**Autora: Ana Cláudia Grossi**

Oito anos de experiência profissional na área de tecnologia da informação, atuando em qualidade, configuração, validação, verificação, coletas de medições e auditorias no processo de desenvolvimento de projetos web JEE. Forte atuação na gerência de Quality Assurance, como garantia de qualidade de produtos, gerenciando equipe de Analistas de Testes, utilizando metodologias como Agile SCRUM e XP.

**Autora: Fernanda Alves**

Consultora em Gerenciamento de Projetos (APM e EPM) e Processos. Atua em levantamento, análise, desenho e descrição de processos corporativos para modelos de melhoria de processos utilizando metodologias diversas.

Este artigo apresenta o trabalho realizado na Powerlogic Consultoria e Sistemas para o programa de Melhoria Contínua de Processos MPS.BR Pioneira em métodos ágeis, a Powerlogic procurou combinar a agilidade natural destas metodologias e a maturidade adquirida por meio do MPS.BR, constituindo uma experiência concreta na busca de resultados positivos para processos de TI.

1. Introdução

A Powerlogic Consultoria e Sistemas é uma empresa brasileira, com sede em **Belo Horizonte**, com 12 anos de existência atuando no segmento de desenvolvimento de produtos e soluções e de serviços de fábrica de software. Como projetos piloto para implantação do processo de melhoria contínua da Powerlogic, a área a ser avaliada, primeiramente, foi a de Produtos (P&D).

Como principal objetivo desta iniciativa, a Powerlogic teve seu foco voltado para a melhoria da qualidade do desenvolvimento de seu processo e produtos, consequente satisfação do resultado apresentado a seus clientes e, também, melhores condições de trabalho de seus colaboradores. A alta gerência, em uma decisão estratégica, optou por estudar e adotar modelos de qualidade, como **MPS.BR**. Contribuindo também para esta decisão figuram o crescimento da empresa nos últimos anos, o aumento no quadro de funcionários e diversos projetos sendo executados em paralelo. Desse modo, a Powerlogic procurou padronizar e institucionalizar seu processo visando alcançar patamares mais altos de qualidade para manter o crescimento em escala organizada. A primeira meta alcançada foi a avaliação MPS.BR nível F. Para tal, um grupo de análise e estudo dos processos da empresa - **Software Engineering Process Group (SEPG)** – foi formado e designado para levantamento do contexto atual. A instituição implementadora escolhida foi a ASR Consultoria e Assessoria em Qualidade, de São Paulo, conhecida por seu trabalho frente à implantação e manutenção de sistemas de Gestão da Qualidade. Em junho de 2007, o processo da Powerlogic foi considerado aderente às áreas de processo **nível F**, tendo sido avaliado em 5 projetos. A próxima meta é alcançar o **nível C** de maturidade. Os trabalhos serão iniciados em dezembro de 2007 visando aprimorar diversas áreas do processo, como por exemplo, **Engenharia de Software**, de **Reuso**, **Controle de Qualidade**, **Gerência de Riscos** e, ainda, a melhoria contínua do processo propriamente dito. Este artigo tem por objetivo relatar a experiência da empresa, que iniciou seus trabalhos em meados do

ano de 2006, estudando o modelo de qualidade, verificando a viabilidade de implantação e adequação às necessidades da empresa, analisando o processo atual, sugerindo melhorias e buscando a redução de conflitos e melhor retorno de investimento. A seção 2 trata da utilização de metodologias ágeis como **SCRUM** e **XP** para alcance de **certificação nível F do MPS.BR**. A seção 3 apresenta o processo **MPS.PL**. A seção 4 trata das melhorias percebidas e impacto nas áreas envolvidas e a seção 5 apresenta as considerações finais.

2. Desenvolvimento

Antes de pleitearmos esta avaliação, ainda era possível encontrar processos pouco formais, mal documentados e planejados. Quase sempre estávamos reagindo, e não planejando. Ao nos depararmos com o método ágil SCRUM, houve uma grande identificação. Alguns pontos foram importantes para que decidíssemos aplicar em nossa empresa: a dissociação com os processos da Engenharia Civil, o desenvolvimento iterativo e incremental e a premissa de que software funcionando é medida de progresso e que isso nos diz mais sobre o andamento projeto do que documentação comprehensiva.

Problemas como **imprevisibilidade**, “**ruídos de comunicação**”, **documentações extensas** que pouco ajudam no processo, por exemplo, precisam ser encarados de forma diferente e exigem que não sigamos passos semelhantes entre os diversos projetos existentes, que possuem peculiaridades que apenas com a implementação propriamente dita afloram. Além disso, tarefas do processo de desenvolvimento de software fazem parte de um processo criativo, não linear e não palpável, fazendo com que modelos incrementais e iterativos se apresentem como a melhor solução diante da realidade dos projetos Powerlogic.

Novos modelos de processos e métodos estão sendo estudados e propostos para tentar suprir essas necessidades, tentando organizar o caos existente na maioria dos casos e garantir o sucesso dos projetos, fatos que nem sempre são fáceis de alcançar. Métodos ágeis como **SCRUM** e **XP** e ainda **MPS.BR** são exemplos destas novas propostas.

O **SCRUM** é uma das técnicas ágeis que mais crescem no mundo atualmente e pode ser facilmente acoplado a outros *frameworks* de processo (tais como **Processo Unificado**), com facilidade, quando necessário. Não há restrições quanto à adição de formalidades no SCRUM, desde que realizados à luz dos valores e princípios do “**Manifesto da Agilidade**”.

O **SCRUM** possui três principais papéis: **Product Owner**, **Scrum Master** e **Scrum Team**. O primeiro tem a responsabilidade de definir quais serão as diretrizes principais do projeto (**Release**) e requisitos de alto nível (**Product Backlog**), priorizar de acordo com as necessidades do mercado e aprovar ou reprovar as entregas feitas. É ele quem define o **Sprint** e **Release Goal**. O **Scrum Master** é o líder do SCRUM e é responsável por garantir que o **Scrum Team** trabalhe em condições adequadas, removendo impedimentos, solucionando dúvidas e assegurando a produtividade de cada membro. Além disso, deve garantir que o processo definido está sendo seguido e sugerir à área de Gerência de Processos possíveis alterações no processo. Já o **Scrum Team**, é uma equipe auto-organizada responsável pelo produto gerado ao final de cada ciclo (**Sprint**), que estima o tamanho de cada requisito (**Selected Backlog**) a ser implementado e se compromete em atingir o **Sprint** e **Release Goal** definidos.

O MPS.BR é um programa para Melhoria de Processo do Software Brasileiro, que visa aprimorar a qualidade no desenvolvimento de software das empresas brasileiras.

Buscando associar estes conceitos, os princípios e valores de métodos ágeis foram relacionados e mapeados em todo o ciclo de vida do software a um ou mais de um princípio do Manifesto da Agilidade verificando sua aplicabilidade. Estes princípios abordam aspectos de desenvolvimento como análise, projeto, implementação, testes, além de outros, como gerenciamento de projetos e de métricas de satisfação do produto final entregue. As áreas de processo do MPS.BR foram cuidadosamente relacionadas com as características dos métodos ágeis. Desse modo, foram discutidos gerência de requisitos, planejamento e acompanhamento de projetos, gerência de configuração, garantia de qualidade de software e medição e análise, presentes no nível F do MPS.BR e sua aplicabilidade ao SCRUM e XP.

Merece destaque a constatação de que a adoção de modelos de qualidade de software, como o MPS.BR, traz um maior formalismo ao processo, agregando valor com a qualidade de software desenvolvido. Dessa maneira, tem-se que, apesar dos modelos de qualidade e metodologias ágeis possuírem fundamentos inicialmente pensados como divergentes, eles puderam facilmente complementar um ao outro e restaurar o equilíbrio em nosso processo de desenvolvimento de software.

O trabalho inicial executado foi o de levantamento e desenho do processo existente. Descrevemos uma primeira versão do processo, que está sendo melhorado a cada dia e aplicado em projetos para verificação da adequação do mesmo. Melhorias significativas foram percebidas com o processo inteiramente institucionalizado e funcionando em um fluxo contínuo e de fácil entendimento. Atualmente, o processo se encontra na versão **PDS_P&D_v18**.

O produto *eCompany Process*, desenvolvido internamente em JEE, nos apóia no gerenciamento dos projetos e também no suporte e análise de nosso processo para a área. Toda a documentação do processo se encontra a um clique de distância para todos os envolvidos. Reuniões diárias de 15 minutos são executadas (*Daily Scrum*) para acompanhamento do projeto. Utilizamos o *Agile Radiator*, um quadro branco que exibe requisitos (*Selected Backlogs*) pendentes, em andamento e finalizados para *Sprint* corrente. Além disso, ele também evidencia os impedimentos e riscos identificados durante o . O plano de projeto (*Release Plan*) é afixado neste mesmo quadro para informações adicionais do *Release* e para melhor comunicação entre os envolvidos. O *eCompany Process* está integrado a um produto de portal corporativo, também da Powerlogic, chamado *eCompany Portal*, formando uma rede de conhecimento de todos participantes do ecossistema corporativo.

Paralelamente, nos deparamos com o modelo MPS.BR, que analisado inicialmente, se apresentava muito formal e pouco aderente à nossa realidade. Por outro lado, entendemos seus conceitos consistentes, com resultados comprovados e com grande aceitação por toda a comunidade. Poderíamos então continuar com os benefícios do SCRUM e XP e ainda conseguir um nível de maturidade MPS.BR tão sonhado? Como nosso objetivo é alçar vôos mais altos, não poderíamos deixar de tentar fazer esta fusão tornar-se realidade. MPS.BR diz “**o que fazer**”, mas não impõe o “**como fazer**”. Metodologias ágeis são um conjunto de melhores práticas que contém informações específicas de “**como fazer**” e estamos, desta forma, pesquisando, trabalhando e colocando em prática o que tanto acreditamos.

3. Processo Powerlogic

O processo subdivide-se inicialmente em três grandes fases, cada uma delas iterativas e com produção de resultado em espiral, acrescentamos também algumas atividades no item Monitoramento e Controle que acontecem durante todo o *Release* desenvolvido.

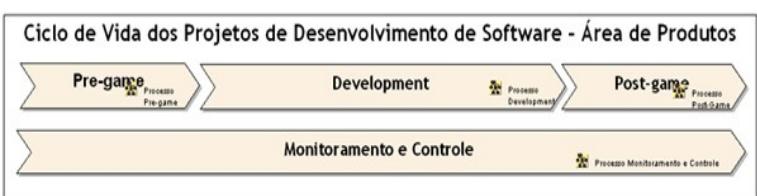


Figura 1 Processo Powerlogic para área de P&D

3.1 - *Pre-game*

Esta etapa tem o objetivo de definir a estratégia de atuação do *Scrum Team* e o planejamento das etapas do trabalho. A partir do planejamento aprovado, poderão ser feitos o acompanhamento e o controle do resultado. Dentro do *Pre-game*, há a definição de um novo *Release* de um produto baseado na lista de *Product backlogs* cadastrados para o mesmo, seguido de uma estimativa preliminar de custo e prazo. Os riscos são levantados, identificados e ações apropriadas para minimizar estes são cadastradas. A matriz de restrição é utilizada neste momento a fim de que acordos de ajuste sejam estabelecidos, em consenso entre os *stakeholders*, definindo-se as dimensões que serão priorizadas (**Tempo / Custo / Qualidade / Escopo**). O *Scrum Team* é definido através da identificação das habilidades requeridas para o projeto e posterior consulta à matriz de competência que mapeará os recursos adequados.

3.2 - Development

Esta etapa tem o objetivo de pôr em prática o plano de ação (*Release Plan*) estabelecido pelo *Scrum Team*, *Scrum Master* e *Product Owner*. É realizada de forma iterativa, através de ciclos de "tempo fechado" (*Time-Boxed*) em **30 dias corridos**, que são *Sprints*, seguidos de 15 dias de testes da equipe de controle de qualidade (*QA Team*), chamado de *Post-Sprint*. Em um *Sprint*, portanto, são executadas atividades de refinamento de requisitos, análise, projeto, desenvolvimento e testes, devendo resultar em um incremento do produto funcionando para o *Product Owner* ao final do , na reunião de *Sprint Review*.

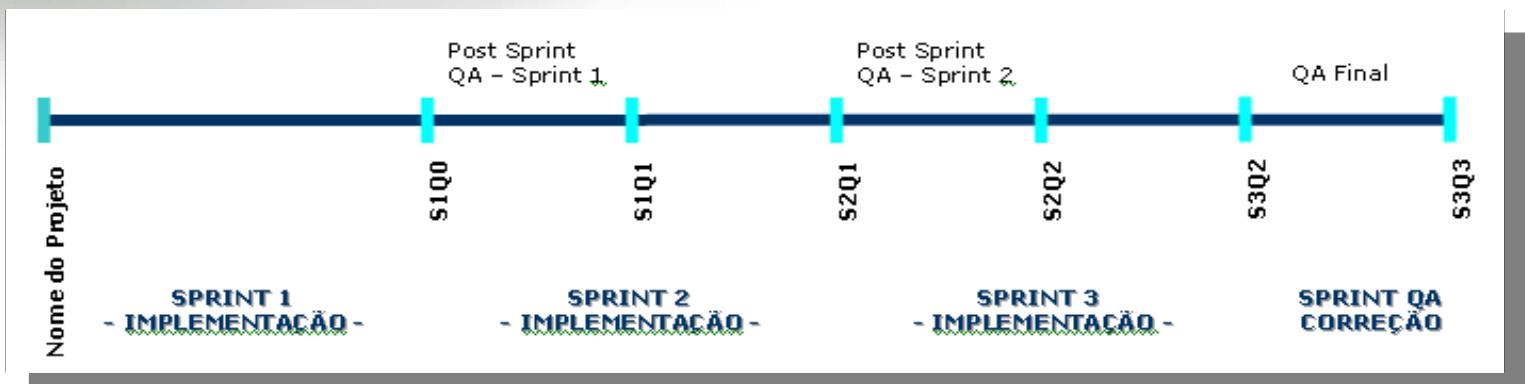


Figura 2 Fase Development

Durante os *Sprints*, as reuniões de *Daily Scrum* ocorrem diariamente, onde se discutem o trabalho realizado no dia, as tarefas para o dia seguinte e os impedimentos que surgiram. Durante esta reunião, são realizados a redistribuição, revisão e ajustes. Ela visa a atender o acompanhamento e monitoramento do planejado para o projeto. O *Scrum Master* é incumbido de **remover impedimentos** no prazo máximo de 24 horas corridas. Os riscos são continuamente monitorados e planos de contingência são definidos.

A reunião de revisão do ciclo, chamada *Sprint Review*, é realizada ao final de todo *Sprint*, com a condução do *Scrum Master* e participação de *Product Owner* e *Stakeholders*. Uma apresentação do produto com as novas funcionalidades é feita de modo a demonstrar sua aderência com o *Sprint Goal* estabelecido. Este processo se inicia novamente, **para cada novo Sprint**.

3.3 - Post-game

Esta etapa tem o objetivo de realizar um fechamento e avaliação do *Release* como um todo, inclusive liberando produto para clientes e refletindo-se acerca das práticas empregadas, indicadores finais e aprendizado em geral.

Nesta etapa, ocorre a reunião de *Release Review*. Esta reunião consiste em uma demonstração do produto final do ponto de vista do cliente, discussão com os *stakeholders* e levantamento de lições aprendidas. A documentação final, CD com o e funcionalidades já testadas são o resultado final desta fase.

3.4 - Monitoramento e Controle

O **Monitoramento e Controle do Release** têm o objetivo de garantir o bom andamento planejado nas reuniões de planejamento. Este acompanhamento é executado pelas reuniões de: *Daily Scrum* (reunião de acompanhamento do *status report* do *Sprint*),

Sprint Review (reunião de encerramento do *Sprint*), **Release Review** (reunião de encerramento do *Release*) **Retrospective Meeting** (reunião de retrospectiva do Sprint/Release).

Diversos apontamentos, tais como percentual de andamento real, apropriações de horas trabalhadas, novas versões dos artefatos de código-fonte, etc., são continuamente coletados (**diariamente** ou **várias vezes** ao dia). E informações do tipo "previsto x realizado" são disponibilizadas para o *Scrum Master*, *Product Owner* e *Stakeholders*. Quadros brancos mantêm as informações mais importantes disponíveis, afixados em local visível para toda a organização.

4. Melhorias Percebidas

Até o momento, melhorias significativas foram percebidas em relação ao desempenho do **Scrum Team** e na qualidade do produto final. Os indicadores criados para as áreas de processo **Gerência de Projetos** e **Gerência de Requisitos**, principalmente, proveram importante *feedback* para a equipe e criou metas a serem alcançadas por cada um. Eles apóiam na tomada de decisão e criam atmosfera de busca de melhoria contínua. Indicadores de **Gerência de Configuração** asseguraram que as práticas determinadas foram seguidas provendo maior controle das versões geradas e integração contínua. Práticas de Gerência de **Qualidade para Processo** garantem a institucionalização e desempenho do processo e produtos de trabalho, através das nas áreas envolvidas. Ao identificar problemas, a **Gerência de Qualidade de Processo**, deve apresentar proposições de soluções e melhorias, e acompanhar suas deliberações até sua finalização.

Para a fase de **Pre-game**, algumas melhorias se destacaram:

- **O planejamento da disponibilidade** dos recursos levando em consideração impedimentos e horas de retrabalho já identificados, horas gastas em reuniões, atendimentos externos, férias, etc, foi executado. Dessa maneira, foi garantida a participação real de cada membro envolvido. Planejar o planejamento foi um grande benefício resultante. Identificação e categorização dos riscos antecipando possíveis problemas. Esta prática proporcionou uma visão mais ampla sob todos os aspectos do projeto (custo, prazo, etc);
- **Alinhamento de metas e planejamento.** O *Release* e *Sprint Goals* são definidos com o consenso de todos os envolvidos promovendo comprometimento e visibilidade. As estimativas de tamanho dos requisitos são efetuadas pela própria equipe de desenvolvimento durante as reuniões de planejamento. Esta prática faz com que o planejamento se aproxime da realidade, uma vez que o profissional que executa é o mesmo que estima;
- **Definição de indicadores** do *Scrum Team* e individuais estimularam o alcance de melhores resultados. (Indicadores de produtividade, percentual de retrabalho e percentual de desvio previsto x real).

Melhorias percebidas para a fase de *Development*:

- **Gestão a vista via Agile Radiator**, provendo *feedback* real e imediato e reuniões de inspeção contínua;

- A **integração contínua** trouxe resultados importantes e informações essenciais para o planejamento e acompanhamento do projeto. O *commit* código fonte, associado ao requisito que deu origem, propicia rastreabilidade em ambos os lados. Uma matriz de rastreabilidade foi construída para se obter estas informações facilitando a análise de impacto;
- **Menos interrupções** para o *Scrum Team*, visto que o *Scrum Master* tem a responsabilidade de assegurar que a equipe trabalhe orientada ao *Sprint/Release Goal*. A equipe fica focada e consegue entregar o combinado. Conceito *Impediment Backlogs* proporcionando identificação e gerência de interrupções e apropriação correta nestes itens pelo *Scrum Team*;
- **Retrabalhos são abertos pelo *QA Team*** mediante aprovação do *Scrum Master*. São considerados requisitos emergenciais garantindo a estabilidade do produto;
- **Reuniões de reestimativa (*Estimating Meetings*)** e de elucidação de requisitos (*Follow-Up Meetings*) ajudam na execução dos requisitos pelo *Scrum Team*;
- **Integração de equipe:** conceito de pilha entre requisitos estimulando a troca de conhecimento, uma vez que o requisito não possui “dono”. Cada membro pode executar atividades de módulos diferentes, utilizando *Pair Programming* quando necessário. O conceito de código compartilhado também é importante, dado que o mesmo será modificado/revisado por diferentes recursos;

- **Sabe-se onde está e aonde quer chegar.** A definição em consenso do *Sprint/Release Goal* possibilita melhor visibilidade e norteia o caminho a seguir para cada envolvido.

Melhorias percebidas durante a fase de *Development*, mas após o fim de cada Sprint:

- **Reunião de *Retrospective Meeting*,** onde ocorre a coleta das lições aprendidas do projeto. Avalia-se o que deu certo (What went well – WWW) e o que deu errado (What can be improved) alimentando o projeto e também o processo;
- **Reunião *Retrospective Meeting 2*,** onde se avalia se algo está sendo feito com relação ao que foi levantado durante o primeiro *Retrospective Meeting*.

Melhorias organizacionais percebidas:

- **Gerência de Qualidade de Processo:** esta nova área criou uma zona de desconforto “sadia”, fazendo com que as pessoas dêem o melhor de si e concretize ações para o objetivo maior organizacional. Ela provê apoio total à condução dos projetos, executa a manutenção do processo, aceitando sugestões e inserindo melhorias necessárias e executa auditorias para garantia da execução do processo;
- **Gerência de Configuração:** esta área garante a integridade dos itens de configuração do Release, apóia a geração de *baselines* e integração contínua;

5. Considerações Finais

Este artigo apresentou parte da **experiência vivida pela Powerlogic** no processo de implantação do programa de qualidade de desenvolvimento de software MPS.BR, inspirado pelas escolas ágeis, em especial o processo SCRUM. Foi apresentada parte do processo e evidenciadas as melhorias percebidas para as áreas envolvidas após a institucionalização do mesmo.

Situações pouco previsíveis e surpresas desagradáveis durante o processo de desenvolvimento de software são, infelizmente, comuns. Entretanto, não há como evitar os casos fortuitos. A empresa busca o aprendizado também com os erros, com poucos desvios. Apresentar resultados continuará sendo importante, mas espera-se estar ligado também à qualidade, que também produz resultados e traz consigo trabalho bem feito, maior produtividade e organização. Dessa forma, como próximo passo, está a avaliação para obtenção do nível C do MPS.BR a ser realizada em um prazo

Referências Bibliográficas

- Agile Alliance, 2002. Agile Manifesto, <http://www.agilealliance.org>
- Boehm, B., "Get Ready for Agile Methods, with Care", Computer, 35, 1 (January 2002) 64-69.
- Paulk, M. C., "Extreme Programming from a CMM Perspective", IEEE Software, 18, 6 (November/December 2001) 1-8.
- Schwaber, K., Scrum Development Process, in OOPSLA Business Object Design and Implementation Workshop, J. Sutherland, et al., Editors. 1997, Springer: London.
- Schwaber, K; Beedle, M, Agile Software Development with Scrum, Prentice Hall, (October 2001).
- Takeuchi, H. and I. Nonaka, The New New Product Development Game.Harvard Business Review, 1986(January-February).

Fratech
SOLUÇÕES EM SISTEMAS
Treinamentos | Mentoring | OutSourcing
www.fratech.net

Métricas e o Desenvolvimento Ágil



Autor: Victor Hugo Germano

Bacharel em ciência da computação pela Universidade federal de Santa Catarina, especializado em Gestão estratégica de TI. Atualmente trabalhando na empresa Audaces auxiliando no processo de adoção de metodologias Ágeis.

Introdução

Devido ao custo elevado em se manter um departamento de TI (profissionais caros e equipamentos mais caros), são necessárias formas de se garantir que o investimento feito esteja realmente gerando o resultado esperado. Infelizmente, à medida que nos aprofundamos no assunto, seguindo modelos tradicionais de desenvolvimento e desempenho, verificamos que falta clareza na utilização de ferramentas de mensuração, e que estamos a todo momento medindo aquilo que não é importante, deixando de lado o que realmente justifica a utilização da TI: **a agregação de valor ao negócio**. Isto é: garantir que a organização que suporta uma equipe de TI - seja através de um projeto ou estrutura empresarial - consiga obter vantagens competitivas através dos insumos gerados pela área tecnológica

Ao invés disso, o foco das métricas tradicionais está em atributos de software e consumo orçamentário. Podemos citar: *pontos de função criados, horas trabalhadas, número de linhas de código por desenvolvedor* ou ainda *orçamento gasto*. Tais métricas são reflexo da dificuldade que as empresas possuem em determinar o que deve ser medido na área de TI, e superestimando o valor de tais itens cria-se uma visão míope de um projeto.

Pontos de Função são a tentativa de encontrar uma medida universal de complexidade para projetos. Isto é, interpretando os requisitos de um sistema poderíamos encontrar o seu tamanho real, assim como pode-se determinar com exatidão a distância entre São Paulo e Porto Alegre em quilômetros, por exemplo. Esta proposta assume que se saiba de antemão todo o escopo de um projeto, e exige que se estipule *tudo* que o sistema pode fazer antes de determinar sua quantidade de Pontos de Função. Isto vem de encontro real ao Desenvolvimento Ágil, que assume escopo de um sistema como sendo uma entidade mutável, e incentiva a adaptação à mudança em detrimento da previsibilidade e dos problemas relacionados à tentativa de definir claramente um escopo.

Medidas como linhas de código por desenvolvedor ou horas trabalhadas sempre levam a problemas maiores. Um programador, quando cobrado em linhas de código, evitará qualquer tipo de **reutilização** ou **refactoring** para passar a impressão de ser produtivo quando na verdade ele está ampliando ainda mais a complexidade dos sistemas em que trabalha, dificultando sua manutenção. Ou ainda ele poderá degradar-se com-

pletamente ao trabalhar mais do que o necessário, criando dificuldades de relacionamento e de participação no projeto, bastante contrária à prática *Semana de 40 Horas do XP*, que faz alusão à tentativa de encontrar a causa raiz dos problemas antes de trabalhar além do horário, evitando cometer um erro para corrigir outro erro. Perceba a complexidade que é criar e fomentar tais medidas.

Seguindo o modelo **taylorista** de gerência, assumimos a necessidade de mensurar individualmente performance, e utilizar indicadores individuais para cobrar produtividade de indivíduos de uma equipe, comparando-os com aqueles com as melhores marcas. Infelizmente os resultados podem ser desastrosos. Uma equipe deve possuir uma identidade única, que é projetada ao mundo externo como reflexo de seu comprometimento com o projeto, e não como uma série de indivíduos que trabalham por si em detrimento do grupo. É comum nestes casos que problemas de relacionamento ocorram dentro da própria equipe por causa da competição gerada para saber "quem é o melhor".

As Ferramentas Ágeis

A definição de sucesso, particularmente nos dias de hoje, é em função da habilidade da equipe ao reagir à mudança, não a capacidade de seguir um plano. A forma com que as equipes são medidas influencia diretamente em sua adaptabilidade. Adaptação – ao invés de aderência – traz sucesso.

Jim Highsmith

Tradicionalmente encaramos projetos através da trindade ***custo - escopo - prazo***. E realmente, por muito tempo tais indicadores foram os principais medidores de performance. Não existia a necessidade de mensurar o valor de negócio entregue ao término de um projeto, nem mesmo o quanto saudável sua execução foi para a equipe. Entretanto, se quisermos alcançar os maiores benefícios do Desenvolvimento Ágil, é necessário ampliar este modelo encontrando formas mais adequadas de indicar sucesso e incentivar nossas equipes a se tornarem mais produtivas. Por isso o foco concreto na comunicação aberta e adaptação através do ***feedback***. Além, obviamente, das métricas de produtividade.

Baseado no **Efeito Hawthorne**, que demonstrou a mudança de produtividade de trabalhadores através do simples fato deles terem conhecimento de tal medição, **Mike Griffths** apresenta a idéia de que métricas efetivas não devem, em nenhum momento, atrapalhar a qualidade do projeto (medir linhas de código não é o caso!), e deve estar direcionado a elementos chave para o desenvolvimento. **Donald Reinertsen** propõe a seguinte reflexão:

Primeiro, deve ser simples, métricas ideais são geradas automaticamente no sentido de que não é necessário esforço extra para serem produzidas. Segundo, devem ser relevantes. Isto é, o controle deve ser feito pelas mesmas pessoas que estão sendo medidas. Psicólogos já descobririram que quando as pessoas acreditam que podem controlar alguma coisa, elas estarão mais motivadas para controlá-la. Mensurar uma equipe por algo que ela não tem controle apenas causará estresse, insatisfação e alienação. Terceiro, deve ser focado em indicadores de condução.

Meça olhando para o futuro, e utilizando tais dados para relatar o passado. É mais fácil medir o tamanho de uma fila de testes do que é medir o tempo de processamento dos testes individuais pelo fato de que o tamanho da fila é um indicador de condução para futuros atrasos no processamento do teste.

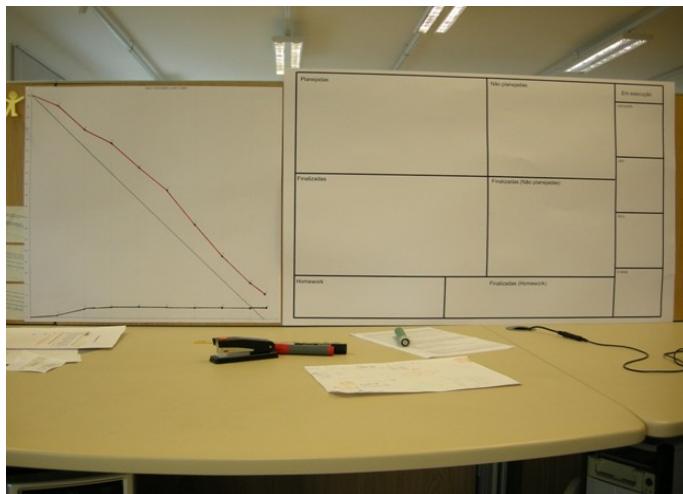
Modelos ágeis em sua concepção já promovem um jogo completo de métricas que podem ser coletadas de forma transparente. Testes unitários, análise da qualidade de código e outras medidas técnicas podem ser inclusive automatizadas no processo de **Integração Contínua**. A unidade comum de trabalho - a história - é o fundamento para medidas de gerenciamento de projetos e qualidade de alinhamento de negócio. É importante lembrar que todo o ferramental existe para dar suporte à tomada de decisão, identificando o mais rápido possível desvios e corrigindo o curso de um projeto.

Não escreva, mostre!!

Fato: **somos serem visuais**. A compreensão que temos do mundo está associada à forma que o enxergamos à medida que ele evolui. Nos esquecemos, em nossos projetos, da importância de apresentar informações de forma visual a todos os envolvidos. Ao invés disso, restringe-se o acesso a informações vitais de progresso, num modelo de alienação claro, que apenas inibe o comprometimento. Na tentativa de controlar todos os procedimentos de um projeto, nos afogamos em diagramas complexos, documentações que muitas vezes servem como justificativas, caso fracassemos.

Enquanto isso, não é incomum encontrar dentro de um projeto expressões do tipo: "Mas agora já é tarde demais! Não podemos refazer o **cronograma e o orçamento...**". Os Métodos Ágeis tentam transformar a avaliação de um projeto em uma atividade diária, realizada por toda a equipe, evitando que riscos e problemas sejam ignorados por meses de desenvolvimento. Muito mais difícil esconder problemas dentro de um projeto Ágil, uma vez que o fluxo contínuo de valor deve ser estabelecido.

Como forma de ampliar a comunicação e a colaboração da equipe, propomos um ambiente de trabalho que possa refletir de maneira clara, objetiva e aberta, os resultados e progressos de um projeto. Mais do que frequentemente você perceberá em ambientes Ágeis inúmeros quadros e **post-its** na parede. Existe muita relevância nesse modelo de apresentação, e uma vez que a equipe é responsável pelo sucesso de um projeto, cabe a ela a função de avaliar problemas e tomar as devidas decisões corretivas mitigando problemas.



A imagem acima foi retirada do ambiente de desenvolvimento da empresa **Audaces Automação Industrial**. Após alguns ciclos, de forma adaptativa a equipe construiu um quadro que correspondesse a suas necessidades de exposição de resultados.

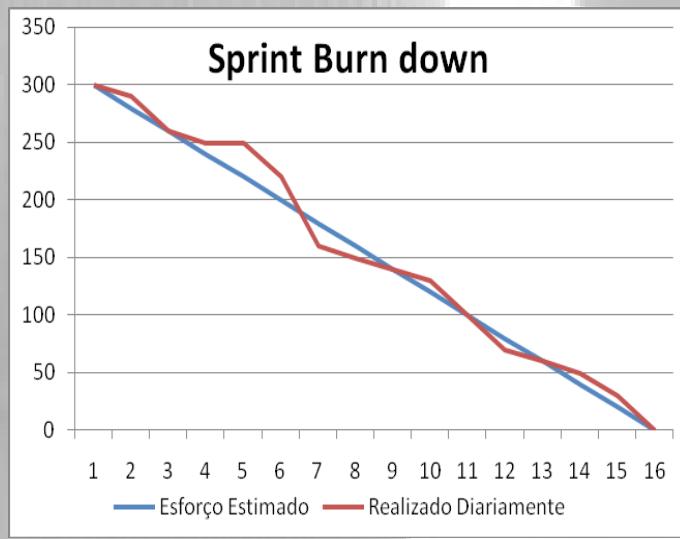
Além do **gráfico Burndown** que trataremos abaixo, existe ainda uma curva indicando tarefas não planejadas. Do lado direito do quadro, existem os espaços para divulgação das tarefas a serem realizadas em uma iteração. São subdivididos em atividades planejadas e não planejadas para a iteração, e a indicação de quais destas estão finalizadas. No canto extremo direito, há uma pequena tabela com as atribuições de cada integrante da equipe, que é atualizada através das reuniões diárias.

Entenda como funcionam gráficos no Desenvolvimento Ágil:

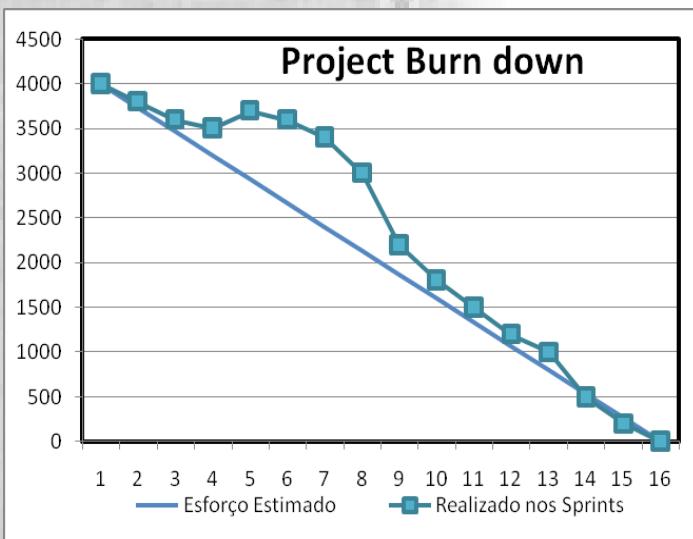
Burndown

O gráfico Burndown é bastante associado ao Scrum, e o impacto causado nas equipes nas primeiras utilizações é bastante motivador. Simples e direto, o modelo é uma relação entre **Esforço Estimado** para uma iteração e sua completude através do tempo. No gráfico abaixo, são apresentadas duas linhas: a azul apresenta a quantidade de esforço que é necessário para finalizar todas as funcionalidades, servindo como uma linha mestra à equipe.

Diariamente o gráfico será atualizado e as funcionalidades finalizadas terão seu valor desenhado no gráfico, apontando o real estado do desenvolvimento em comparação à estimativa original.



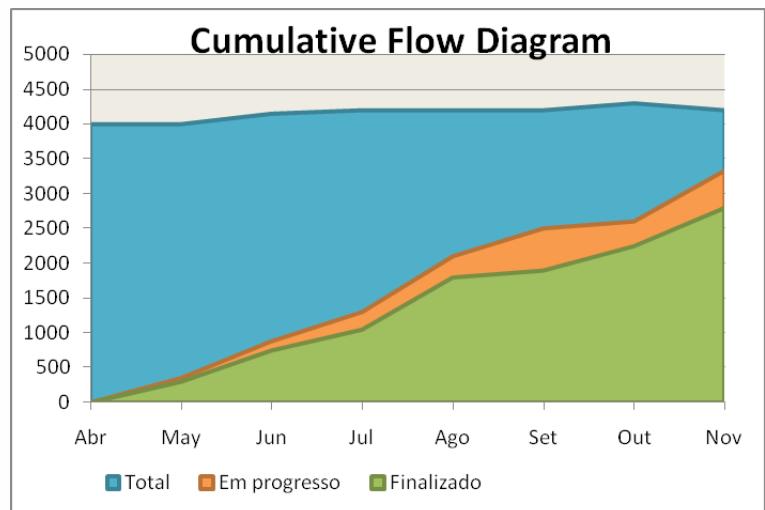
Ainda num projeto Scrum, a equipe acompanha seu progresso sobre o planejamento inicial do projeto ao término de cada Sprint. O eixo horizontal do Projeto ou Release do gráfico abaixo apresenta as Sprints; o eixo vertical nos mostra a quantidade de trabalho restante ao início de cada ciclo. Tal trabalho restante pode ser apresentado em qualquer unidade de medida que a equipe desejar: story points, dias ideais, complexidade,etc.



CFD - Cumulative Flow Diagrams

Funcionalidades completas versus funcionalidades restantes é uma métrica vital. Ao término de um dia de trabalho, é a quantidade de funcionalidades geradas e não o número de linhas de código ou pontos de função que geram valor para o cliente. **"Funcionalidades Completas"** significam aceitas pelo cliente (ou seu proxy) - não apenas codificadas e testadas - já que não agregam nenhum valor até o momento que estiverem completamente finalizadas (pronto igual a pronto mesmo!). Na verdade, código escrito e não aceito pelo cliente é considerado inventário, e seguindo os princípios do **Lean Thinking**, isto é puramente desperdício.

Complementar ao BurnDown Chart do Scrum, trazendo a relação de requisitos completos, requisitos restantes e requisitos em desenvolvimento, segue abaixo um modelo do gráfico chamado **Cumulative Flow Diagram**



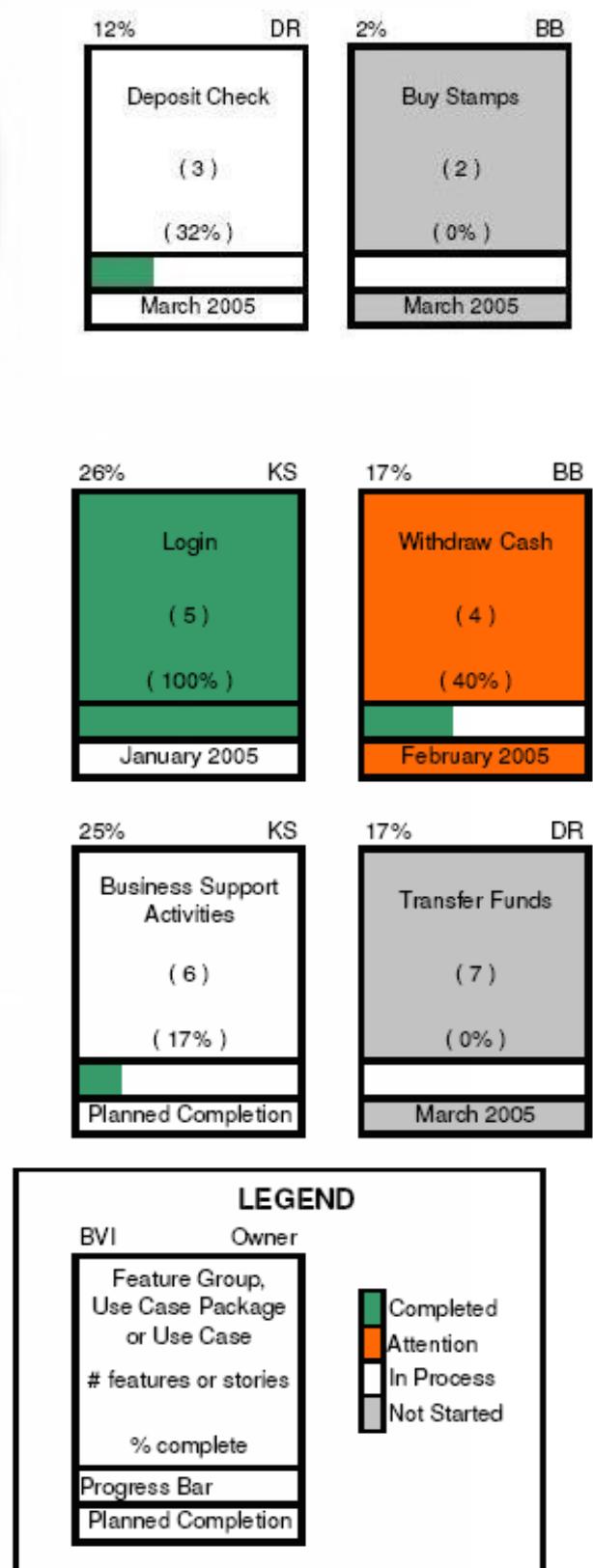
No exemplo acima, o progresso é indicado pela completude da faixa verde no decorrer do projeto. Importante notar a mudança de proporção relacionada à faixa superior, indicando aumento de User Stories no decorrer do projeto. Esta mudança pode indicar um problema relacionado à comunicação entre produto e desenvolvedores.

Parking-Lot Chart

A idéia é utilizar agrupadores para reunir requisitos com o mesmo domínio e montar um gráfico contendo Domínio, Número de histórias em cada domínio, número de pontos de história e a completude. Teve sua origem no modelo de desenvolvimento Feature Driven Development. As cores do gráfico apresentam onde o progresso está sendo feito, áreas de preocupação e itens não iniciados.

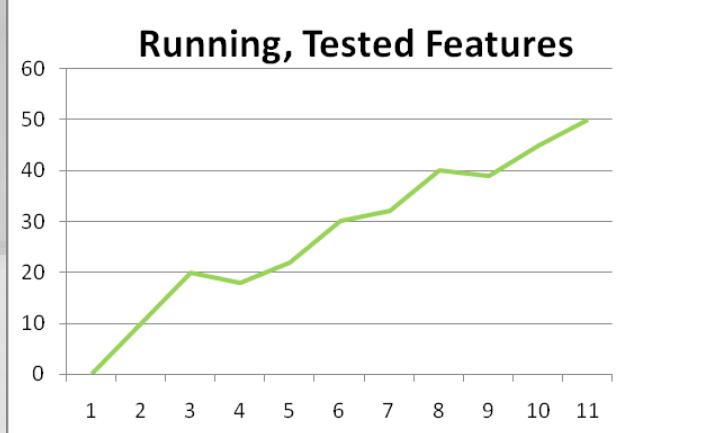
Pode-se ter uma visão bastante real do estado atual do projeto, sendo eficiente para apresentações em reuniões de status, mostrando à diretoria uma fotografia dos principais pontos do projeto, facilmente organizado em uma folha A4, assim como Carlos Slim, homem mais rico do mundo, acredita ser o imprescindível para a tomada de decisão.

Brent Barton, Dan Rawsthorne e Ken Schwaber apresentam um modelo bastante ilustrativo de como um gráfico desse tipo deve ser confeccionado. As cores são importantes pois indicam posições em que se deve tomar mais atenção ou ainda a finalização de um domínio todo.



TF: Running, Tested Features

Representa a quantidade de requisitos que estão passando em todos os testes de aceitação. É facilmente gerado utilizando ferramentas automatizadas de teste. O raciocínio é simples, e idealmente deve parecer com a imagem abaixo:



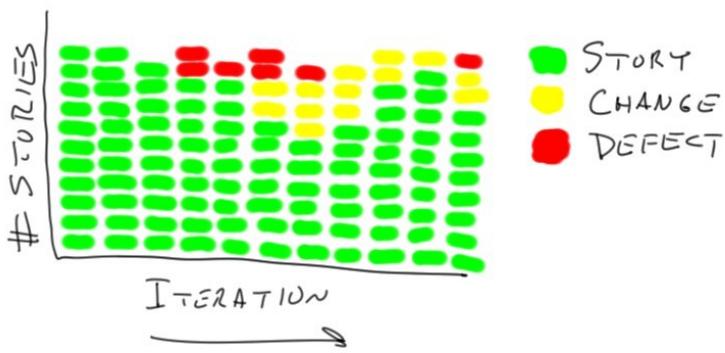
Também conhecido como *Burn-Up chart*. Qualquer anomalia no comportamento desta imagem deve ser encarada como um problema, já que, teoricamente, os requisitos implementados devem passar em todos os testes.

Evolução, Defeitos, Mudanças

No decorrer de um projeto alguns defeitos podem ser encontrados. Mesmo utilizando todas as técnicas Ágeis que favorecem a produção de sistemas com melhor qualidade, seria ingenuidade acreditar que Bugs não aparecerão. Sendo assim, é importante aferir o quanto de trabalho está sendo despendido na correção de problemas no desenvolvimento.

Bugs podem significar que a aceitação de testes pelo usuário está em falta ou mesmo o processo de engenharia de software necessita ser revisto. Talvez seja necessário utilizar mais a fundo o TDD.

Um dos princípios do desenvolvimento Ágil é justamente garantir que software em funcionamento seja entregue ao cliente, ampliando a agregação de valor. Podemos encarar o fato de construirmos softwares com bugs também como desperdício, já que problemas no desenvolvimento não agregarão valor ao cliente e causarão retrabalho da equipe, evitando novamente que sejam entregues funcionalidades nas próximas iterações. Além disso, mudanças nos requisitos ou histórias podem ter um significado positivo ou negativo. Positivo quando cliente está mais experiente e está aperfeiçoando as estórias, garantindo que a equipe possa ampliar o conhecimento do processo de negócio. Mas muitas mudanças podem significar que o projeto está instável, que se deve gastar um pouco de mais tempo na hora de descrever uma estória e elucidar requisitos. Com a intenção de identificar algum possível erro na entrega de valor pela equipe, apresento do gráfico abaixo, intencionalmente desenhado manualmente por Ron Jeffreis.



Conclusões

Métricas são necessárias, sejam para cobrar ou avaliar o progresso dos projetos. O principal benefício de utilizá-las em projetos Ágeis é ter informações o mais rápido possível, de maneira objetiva e clara, ajudando a tomada de decisão. Logo nas primeiras iterações de um projeto já é possível ter visivelmente idéia de quais são os problemas, e não haveria assim espaço para justificar fracassos e riscos desconhecidos. Entretanto, é preciso mudar conceitos sobre o que deve ser medido, e de que maneira esta mensuração não afetará o trabalho da equipe. Valorize acima de tudo a comunicação direta entre equipe e gerência, tirando o melhor proveito da utilização do Desenvolvimento Ágil.

OpenUP: Uma Introdução



Autor:José Papo



José Papo é engenheiro de software e mestre em engenharia de computação pelo IPT. Professor da graduação e pós-graduação na Universidade São Judas Tadeu. Possui as certificações IBM Certified Solution Designer Rational Unified Process V7.0, IBM Certified Specialist for Requirements Management w/Use Cases, IBM Certified Solution Designer – Object Oriented Analysis and Design vUML 2, IBM Rational Solution Sales Professional, Certified Scrum Master, ITIL Foundation Certified Professional e OMG UML 2.0 Certified Professional.

OpenUP – Uma Introdução

OpenUP é um processo enxuto, baseado no “*Unified Process*”, que possui um ciclo de vida iterativo e incremental. O OpenUP também foi elaborado como uma filosofia ágil, pragmática e que foca na natureza colaborativa do desenvolvimento de software. É um processo de baixa cerimônia e que não indica nenhum tipo de ferramenta específica. Uma das características visíveis do OpenUP é que a disciplina de gestão de projetos é uma adaptação do Scrum (processo ágil empírico com foco nos aspectos de gestão de projetos, sem entrar em detalhes da engenharia de software).

O OpenUP possui princípios (isto é, se você não seguir um dos princípios você não está na realidade adotando o processo como se deve) semelhantes à versão 7 do RUP. Eles são:

- Balancear prioridades competidoras para maximizar o valor aos envolvidos do projeto.
- Colaborar para alinhar interesses e compartilhar entendimento.
- Focar cedo na arquitetura para minimizar riscos e organizar o desenvolvimento.
- Evoluir para continuamente obter feedback e melhorar.

O OpenUP pode ser mais facilmente entendido através dos seus ciclos de visibilidade, representados na figura 1. Vamos falar sobre cada um deles separadamente, na seqüência do ciclos mais curto até o ciclos mais longo, para dar uma visão clara de como é a vida em um projeto OpenUP.

Micro-Incrementos

O esforço de cada membro da equipe é organizado em micro-incrementos. Estes representam pequenas quantidades de trabalho que produzem uma continuidade mensurável de progresso do projeto. Um micro-incremento pode representar o resultado de algumas horas ou de alguns poucos dias de trabalho para uma pessoa, ou para um grupo de pessoas trabalhando colaborativamente.

Cada micro-incremento deve ser bem definido para que a equipe possa monitorar e controlar o progresso diário. Cada micro-incremento é especificado em um item de trabalho (*work item*). Essa é uma das diferenças de outros processos como o RUP e o Scrum. Todos os requisitos e atividades do projeto OpenUP serão associados a um item de trabalho. Não existe divisão entre fluxos como caso de uso, cenário, mudança e defeito. Como esses requisitos e atividades podem ser grandes, é possível dividir um item de trabalho em vários itens de trabalho menores, até ficar com um nível de granularidade aceitável (de preferência não ultrapassar três dias).

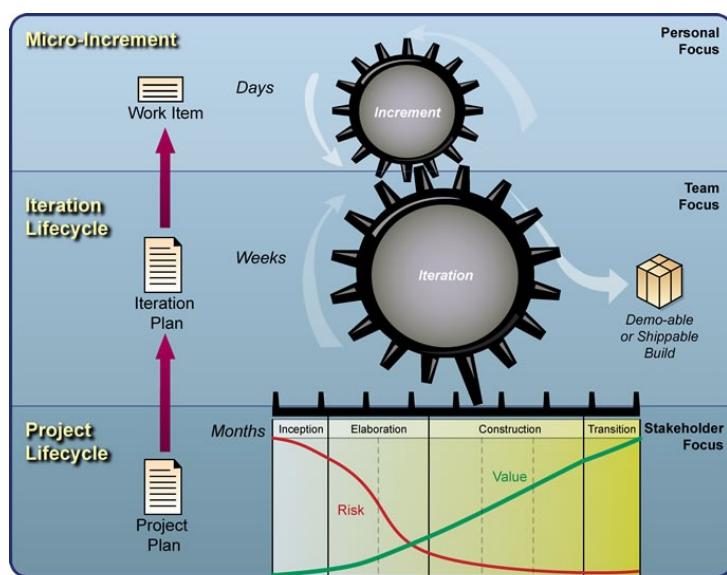


Figura 1 – Ciclo de Vida do OpenUP (Fonte: OpenUP versão 1.0)

Vamos a dois exemplos concretos:

- Definir a Visão do Produto é uma atividade que pode levar várias semanas. Para assegurar a monitoração do progresso diário você deve dividir esse item de trabalho em itens de trabalho menores. “Identificar os Envolvidos do projeto” e “Identificar as restrições” seriam dois desses itens de trabalho que representam bons micro-incrementos.

- Implementar a solução é uma atividade que pode levar muito tempo se for baseada em casos de uso ou mesmo cenários de caso de uso. Para assegurar o progresso contínuo é mais interessante dividir esse item de trabalho em vários micro-incrementos. Uma forma poderia ser criar um item de trabalho menor para projetar e implementar um subfluxo ou apenas um passo de um caso de uso ou cenário.

A aplicação evolui através da execução de vários itens de trabalho simultaneamente. A equipe utiliza técnicas de reuniões diárias e também ferramentas de colaboração para conseguir transparência no trabalho de equipe. Ao mesmo tempo, progresso contínuo é demonstrado através da integração contínua de micro-incrementos durante cada dia do projeto.

Ciclo de Vida da Iteração

As iterações no OpenUP têm como objetivo primordial focar a equipe na entrega de resultados de valor aos clientes, a cada ciclo curto de algumas semanas (o tempo de iteração é variável, mas o OpenUP recomenda iterações de 2 a 6 semanas). No final de cada iteração a equipe, obrigatoriamente, deve entregar um *build* executável do software, já implementado e testado. Este *build* conterá um incremento do produto.

Os processos de planejamento, estimativa e monitoramento do progresso de cada iteração se baseiam nos itens de trabalho. O plano de iteração corrente costuma incluir sempre os itens de trabalho de maior prioridade.

Cada iteração deve iniciar com uma reunião de planejamento de iteração. Toda a equipe deve participar. Esse período deve durar de um a dois dias e já elabora também detalhes arquiteturais de alto nível, bem como a ordenação e o detalhamento dos itens de trabalho em micro-incrementos. Depois desse período a equipe executa a iteração. É importante lembrar que o time deve utilizar a integração contínua e builds diários.

Para fornecer ainda mais disciplina, um build estável semanal deve ser produzido. Uma atenção maior dos analistas de testes deve se concentrar nos builds semanais. Nos últimos dias de uma iteração o foco deve ser na correção de defeitos do incremento executável. A iteração termina com uma avaliação do que foi construído versus o objetivo planejado no início da iteração. A equipe também realiza uma retrospectiva para obter lições aprendidas da iteração e melhorar aquilo que não funcionou tão bem durante a iteração.

O OpenUP, como outros processos ágeis, valoriza muito times auto-organizados. Isto significa que a equipe como um todo é responsável por organizar o trabalho e determinar como melhor atingir seus compromissos. Transparência, comunicação aberta e comprometimento pessoal são fundamentais para a auto-organização funcionar. O OpenUP assume que o gerente de projetos atue mais como um líder educador, um líder servidor e um *coach*. Ele, portanto, deve eliminar o estilo “chefe-subordinado” de seu paradigma.

Ciclo de Vida do Projeto

O OpenUP organiza um conjunto de iterações em fases. Essas fases possuem o mesmo nome das fases contidas no Processo Unificado (Unified Process) e no Rational Unified Process (RUP). Cada fase termina com um marco, que possui como objetivo mitigar um determinado tipo de risco tipicamente importante para os envolvidos do projeto:

- Iniciação: eliminar riscos relativos ao caso de negócios do projeto.
- Elaboração: eliminar riscos arquiteturais e técnicos do projeto.
- Construção: eliminar riscos de não construir as funcionalidades prioritárias do ponto de vista dos envolvidos.
- Transição: eliminar riscos de homologação, testes beta, implantação do sistema em produção e migrações de sistemas legados.

Portanto, as fases do OpenUP nada mais são que marcos para redução de riscos. Conforme mostra a figura 2, o OpenUP busca reduzir o risco logo no início do projeto ao mesmo tempo que agrega valor.

OpenUP – Futuro

O OpenUP é apenas o processo mais famoso dentro de uma família de processos sendo desenvolvida. Já existem disponíveis as extensões OpenUP/XP, OpenUP/Scrum e OpenUP/DSDM.

Além disso, será possível adicionar plug-ins ao OpenUP para endereçar questões como arquitetura orientada a serviços (SOA), arquitetura dirigida a modelos (MDA), desenvolvimento de sistemas embutidos, J2EE, .NET, entre outros. A versão futura do RUP será nada mais que um empacotamento do OpenUP e toda uma série de plug-ins adicionais que permitirão ao engenheiro de processos customizar o processo de sua empresa de acordo as necessidade específicas de cada projeto.

Experimente o OpenUP. Faça o download (é gratuito) e aproveite a oportunidade de usar um processo ágil que utiliza os conceitos do processo unificado, já muito difundido no mercado brasileiro.

Link para o Eclipse Process Framework:
<http://www.eclipse.org/epf/>

Link direto para o OpenUP:
http://www.eclipse.org/epf/downloads/openup/openup_dow

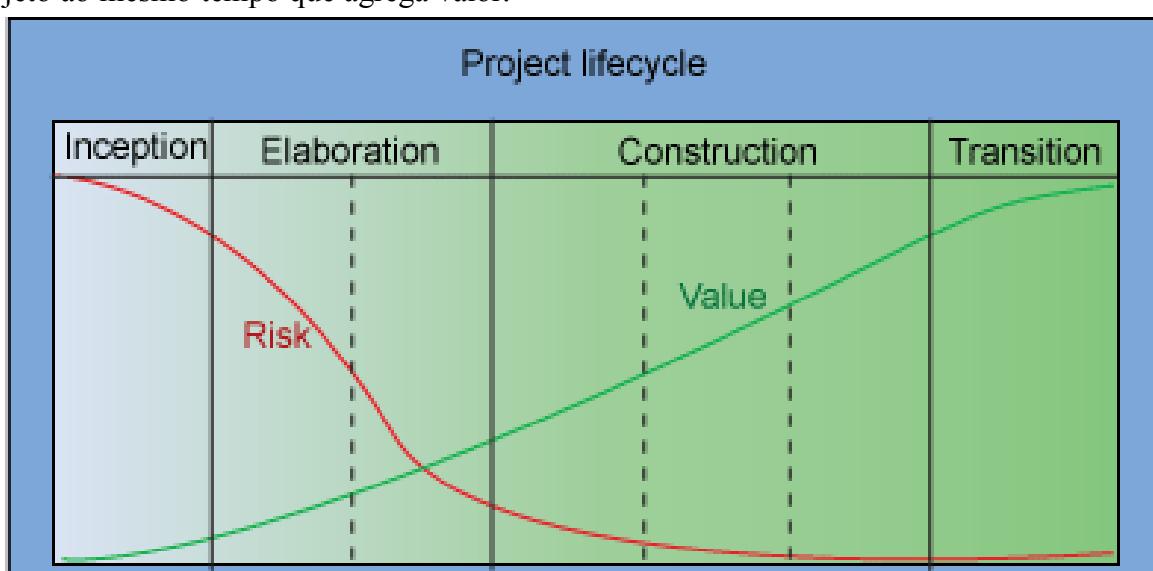


Figura 2 – Redução de risco e geração de valor durante o ciclo de vida do projeto OpenUP
(Fonte: OpenUP versão 1.0)

Aperfeiçoamento de Projetos Ágeis

Parte I: Uma Visão Sistêmica



Autor: Alisson Vale

Tem mais de 15 anos de experiência com desenvolvimento de software e a mais de 6 anos lidera projetos de software dos mais variados tipos e tamanhos. Hoje é Líder de Projeto e Diretor da Phidelis Tecnologia, onde lidera a equipe de desenvolvimento da solução acadêmica oferecida pela empresa.

E-mail: alisson.vale@phidelis.com.br

Weblog: <http://www.phidelis.com.br/blogs/alissonvale>

A essência do funcionamento de um projeto ágil está na sua capacidade de aprender e de se adaptar com esse aprendizado. A maioria das metodologias ágeis estabelece práticas que nos ajudam a ampliar essa capacidade em nossos projetos. A prática mais comum é o uso de sessões conhecidas como “Retrospectivas”. Mas será que estamos preparados para ir além e entender os princípios que estão por trás dessa prática? Nesse artigo, escrito em duas partes, eu convido o leitor a pensar sobre melhoria contínua e sobre os vários aspectos que cercam o tema.

Na teoria darwinista, a adaptação evolutiva é o instrumento de sobrevivência das espécies em ambientes altamente competitivos. Ela lhes atribui a capacidade de se adequar aos mais complexos cenários para preservação da vida. O processo de seleção natural, se analisado sob o ponto de vista sistêmico, resulta no aperfeiçoamento contínuo das espécies por meio da execução de um incomensurável número de ciclos de vida e morte, onde características adquiridas ao longo do tempo são postas à prova. Dentre tais características, as que influenciarem positivamente a sua capacidade de sobrevivência são naturalmente absorvidas, enquanto que as que impactarem negativamente nessa capacidade serão descartadas. A espécie adquire, assim, conhecimento, habilidades, memórias e comportamentos, influenciada por um longo processo de aperfeiçoamento e adaptação contínua. Certamente, o mais fantástico processo de auto-aperfeiçoamento conhecido hoje pela ciência.

A adaptação e o aperfeiçoamento contínuo geram um processo evolutivo que emerge das relações sistêmicas dos organismos com o seu meio ambiente. Em projetos ágeis, seus processos também emergem das relações sistêmicas estabelecidas entre as pessoas envolvidas no projeto (*stakeholders*) e o seu meio de negócio. A capacidade de adaptação é elemento essencial de diferenciação do modelo ágil em relação aos processos tradicionais para desenvolvimento de software. Os pequenos ciclos de avaliação e aprendizado que o compõe desafiam equipe e cliente a evoluírem seu processo por meio da detecção, análise, correção e absorção sistemática de conhecimento sobre erros, ineficiências ou falsas suposições encontrados durante seu tempo de vida.

Aliás, correções sistemáticas de pequenos erros cometidos devido a falsas suposições são, para o método ágil conhecido como *Adaptive Software Development* (Highsmith, 2000), a chave para o funcionamento dos seus ciclos de especulação, colaboração e aprendizado. A palavra “Especulação”, nesse método, já dá o tom necessário para a compreensão de que um ciclo se inicia com base em algo pouco conhecido (Especulação); que é trabalhado colaborativamente (colaboração); e que se tornará muito bem conhecido ao seu final (Aprendizado). No contexto do aperfeiçoamento contínuo, a percepção adequada desse entendimento gira em torno de saber que para ter um processo que seja melhor hoje do que ontem, é necessário (1) especular sobre algo que precisa ser melhorado (2) colaborar para que tal melhoria seja alcançada e (3) aprender e incorporar a melhoria conquistada.

O Manifesto Ágil (AgileAlliance, 2001) endereça a questão do auto-aperfeiçoamento ao afirmar que um time de projeto deve “refletir, em intervalos regulares, sobre como ser mais efetivo, e assim afinar e ajustar seu comportamento apropriadamente”. Todo incremento, iteração ou liberação de release cria, para a equipe de projeto, uma grande oportunidade de recomeço.

Recomeço leva ao aprendizado, e aprendizado leva à melhoria. A maioria dos métodos, metodologias e frameworks ágeis também estará estabelecendo suas práticas e definindo suas idéias sobre melhoria contínua. Processos baseados em Extreme Programming ou Scrum, por exemplo, utilizam as chamadas “Reuniões de Retrospectiva” para discutir e deliberar sobre o quê funcionou e o quê pode melhorar. Alistair Cockburn sugere “Workshops de Reflexão”(Cockburn, 2001) como prática para o estímulo à melhoria contínua em sua família de metodologias conhecida como Crystal.

Outras propostas ágeis como *DSDM*(DSDM Consortium, 2007), *OpenUP* (The Eclipse Foundation, 2007) e *MSF for Agile Software Development* (Microsoft Corporation, 2006) explicitam como princípio básico de trabalho a melhoria contínua. O *OpenUP* recomenda como prática “fazer perguntas e verificar suposições sobre o projeto regularmente”. O *Agile MSF* estabelece como princípio a idéia de “aprender com todas as experiências”. A aplicação desse princípio leva a projetos que “permitem a seus membros se beneficiarem de lições aprendidas por meio do cometimento de erros, e por meio da repetição de práticas de sucesso”. Seja qual for a técnica utilizada, na prática o processo de melhoria contínua gira em torno de, dentre outras coisas, buscar melhores maneiras de (1) aumentar a *quantidade* do tempo em que a equipe está envolvida com atividades que geram valor para o cliente por meio de software funcional; e (2) de aumentar a *qualidade* do tempo em que a equipe está envolvida com atividades que visem garantir as condições de segurança, estabilidade funcional, compreensibilidade e manutenibilidade do software que está sendo desenvolvido.

As Origens

A idéia do aperfeiçoamento contínuo remonta a meados da década de 1950 quando Edward Deming, estatístico americano, levou seu modelo de produção orientado pela qualidade para a indústria japonesa(Deming, 2003).

Deming acreditava que o aperfeiçoamento contínuo era um mecanismo-chave para gerar a necessária vigilância constante não só na qualidade do que está sendo produzido, mas também no rendimento do sistema produtivo. Ele também inovou ao propor que os mecanismos de melhoria sobre o processo de trabalho deveriam estar nas mãos de quem executa o trabalho, não somente de quem o gerencia ou o supervisiona. Assim, é responsabilidade de toda a equipe participar dos processos de obtenção de métricas, identificação e análise de problemas e do planejamento e execução das ações de melhoria planejadas.

Deming também foi um dos primeiros a propor o Ciclo de Schewart (mais conhecido como o ciclo PDCA) como instrumento prático para viabilizar o modelo de aperfeiçoamento contínuo. Em um projeto ágil, o ciclo *Plan-Do-Check-Act* se manifesta de várias maneiras diferentes.

Nesse tipo de projeto, aperfeiçoamento e adaptação são uma necessidade. O PDCA é o instrumento base para a condução desse processo. No PDCA, a equipe precisa: (1) *Planejar*: identificar e analisar os problemas, planejar ações para resolvê-los (2) *Fazer*: executar ações para resolvê-los (3) *Checar*: medir e verificar a eficácia das ações executadas (4) *Agir*: adaptar o plano de melhorias, incorporando não só as melhorias obtidas, mas também o novo aprendizado obtido.

Retrospectivas: PDCA na Prática

A prática de definir uma reunião cíclica, ao fim de cada iteração do projeto, para discutir estruturadamente sobre como melhorar o processo ilustra bem o PDCA.

Em uma Reunião de Retrospectiva de um projeto ágil, é comum o uso de técnicas para identificar, priorizar, analisar, definir problemas e sugerir ações que gerem melhorias para o processo. A técnica mais utilizada consiste em se usar uma folha de flip-chart ou um quadro branco onde se desenha duas grandes áreas: uma para identificação do quê funcionou bem, onde a equipe cola post-its que identificam práticas, ferramentas ou atividades que geraram bons resultados durante a iteração; e uma outra para identificação dos pontos que poderiam ser melhorados, onde outros post-its indicarão problemas ou dificuldades percebidos pelos vários membros da equipe durante a iteração. Dentre os problemas elencados, a equipe pontua, por meio de votos e discussão, aqueles problemas que cada um acha terem o maior impacto para o rendimento do projeto como um todo. Isso é importante, pois os membros da equipe devem votar no problema que eles entendem afetar mais o projeto naquele momento, não naqueles problemas que afetam mais o seu próprio trabalho.

Os problemas mais votados, serão aqueles que obterão uma maior atenção da equipe no próximo ciclo. Por fim, a equipe discute e planeja as ações que serão tomadas para resolver os problemas encontrados.

O PDCA com instrumento de aperfeiçoamento contínuo pode ser visto claramente nessa técnica: o *Plan* aparece no momento em que a equipe levanta os problemas do ciclo anterior (post-its), realiza a devida análise e priorização (votação discutida) e estabelece as ações de melhoria.

A fase *Do* ocorre no ciclo seguinte quando a equipe tenta realizar as ações planejadas. O *Check* se dará na próxima reunião de retrospectiva, quando as coisas boas e os problemas serão novamente discutidos. Mesmo que o problema não tenha sido solucionado, ele agora revela-se mais bem conhecido.

O elemento *Act* se manifesta quando a equipe absorve esse aprendizado, discutindo novas soluções, mudando sistematicamente sua maneira de enxergar o mesmo problema.

Em projetos ágeis, é o padrão sistêmico de adaptação e aperfeiçoamento que faz a equipe controlar o processo, evitando que o processo controle a equipe. Entender o ciclo PDCA e identificar seu funcionamento dentro do projeto é fundamental, pois ele estabelece esse padrão sistêmico cujo resultado é a incorporação de pequenos elementos de controle sobre a complexidade do projeto. É o padrão que garantirá a sustentabilidade do projeto a longo prazo.

Entendimento sistêmico

Um engano comum em projetos ágeis é a visão de que a melhoria contínua visa apenas aumentar a produtividade do time como um todo. Eu diria que isso é apenas parte de uma abordagem mais ampla. A Toyota, por exemplo, enxerga o rendimento de um sistema (no nosso caso de um projeto) segundo uma rede complexa de influência dos chamados “elementos de avaliação primários”: qualidade, produtividade, custo, segurança e atendimento ao cliente (Liker & Meier, 2007). Em seu modelo de melhoria contínua (o kaizen), toda e qualquer ação de resolução de problemas é executada por meio do entendimento de suas influências nos elementos de avaliação primários.

Um defeito, por exemplo. Que elemento de avaliação primária você acha que será mais afetado por uma grande incidência de defeitos no seu produto? Se você respondeu “qualidade” talvez você ainda não esteja pensando no problema sob a ótica do pensamento sistêmico (Senge, 1990). Uma análise sistêmica sobre esse tipo de problema indicaria que todos os elementos de avaliação primária seriam duramente afetados. Mais defeitos significa menos qualidade e mais insatisfação do cliente. Para resolver a insatisfação a equipe aumenta a demanda por inspeções manuais. Para resolver os problemas de qualidade gerados pelos defeitos haverá mais retrabalho, o que diminuirá a produtividade e aumentará o custo do projeto.

Quando todas essas relações não são corretamente compreendidas sob o ponto de vista do sistema produtivo, o caminho mais comum é estabelecer causas superficiais ou ações pontuais para resolver o problema. Veja como é fácil para uma equipe, em uma típica reunião de retrospectiva, estabelecer causas e ações pontuais para resolução de problemas sistêmicos :

Líder de Projeto: Pessoal, chegamos à conclusão de que o nosso principal problema hoje é o aumento no número de bugs. O que podemos fazer para resolver isso?

Desenvolvedor 1: Acho que deveríamos gastar mais tempo testando antes de fazer a entrega.

Líder de Projeto: Mas isso vai significar diminuir o ritmo de entregas! Deve haver outro jeito.

Desenvolvedor 2: Acho que é por aí também. No fundo, estamos sacrificando nossa qualidade para produzir mais. Temos que rever nossas estimativas.

Líder de Projeto: Porquê nossos testes unitários não detectam os problemas? Não poderíamos tentar aumentar sua abrangência?

Desenvolvedor 1: Não sabemos. Temos 80% de cobertura, mas mesmo assim os bugs aparecem. Há muito código de interface gráfica não coberto. Alguns bugs vem daí. Outros surgem de situações que realmente não tínhamos previstos.

Líder de Projeto: OK. Vamos tentar conversar com nosso cliente para diminuir o número de entregas na próxima iteração e verificar se isso vai ajudar a ganharmos mais tempo para os testes.

A princípio não há nada de errado nessa reunião sob o ponto de vista ágil. A equipe elencou seus problemas, votou o que acha mais importante, discutiu as suas causas e estabeleceu algumas ações para

resolver aquilo que é mais importante no momento. O grande problema com essa reunião é que ela é conduzida sob falsos pressupostos. O líder inicia com a pergunta: “O quê podemos fazer para resolver isso?”, ou seja, parte-se direto para a solução sem primeiro tentar entender quais são as reais causas do problema. A pergunta mais indicada talvez fosse “Porquê temos tantos defeitos?”. Partir direto para a solução pode levá-lo a dar o tiro certo no alvo errado. Como a equipe está acostumada a usar o pensamento linear para identificar o problema, ela se concentra em tentar resolver um sintoma, não a doença em si. O pouco tempo para testes não é a causa do problema, é apenas um sintoma. O desenvolvedor 1 dá uma solução que resolve o problema do desenvolvedor (seu código será mais testado), mas não resolve o problema do projeto. Ele colocou apenas dois dos quatro elementos básicos de avaliação em sua análise, ignorando que o custo também aumentaria e que a satisfação do cliente seria afetada, pois ele começaria a receber menos funcionalidades.

O desenvolvedor 2 parece ser influenciado pelo desenvolvedor 1 e o ratifica mantendo o foco qualidade versus produtividade, como se fossem elementos antagônicos.

O pensamento linear os leva a achar que a única maneira de aumentar a qualidade é diminuindo a produtividade. Hoje a administração moderna já sabe que o aumento da qualidade é um dos principais fatores de influência no aumento da produtividade. Ou seja, aumentar a qualidade não vai diminuir a produtividade, muito pelo contrário, vai aumentá-la! O problema aparece quando você tenta separar as atividades de qualidade das atividades de produção.

Na verdade, agindo dessa maneira, tenta-se aumentar o controle do que está sendo produzido, não necessariamente sua qualidade. Os defeitos continuarão a serem introduzidos no produto. O aumento no controle atua apenas aumentando a capacidade do projeto de encontrar defeitos, não de evitá-los. Isso é feito, por exemplo, adotando a prática de inspeções intensas ao fim de um ciclo de desenvolvimento (também comuns em processos tradicionais). As inspeções, se utilizada como único elemento de quality assurance, são ineficientes pois permitem que o processo seja conduzido sem qualidade até que o setor de qualidade receba o produto para testes. Nessa abordagem, a qualidade não é garantida, ela é delegada para um outro time. Qualquer investimento em qualidade, realizado dessa maneira, pode sim significar redução de produtividade e de tempo de entrega.

De fato, não estaríamos ajudando a aumentar a qualidade nesse caso, apenas tentando ganhar proteção e segurança contra os nossos próprios erros.

Depois de ouvir a opinião dos desenvolvedores, o líder de projeto faz sua primeira incursão na direção de tentar entender o problema. Mas, repare que mais uma vez o foco é desviado da causa (“Porquê os testes unitários não detectam o problema?”) para a solução (“E se aumentarmos sua abrangência?”).

Esse é um comportamento comum nesse tipo de reunião. As pessoas ficam muito ansiosas para chegar a uma solução e se esquecem de tentar entender o problema.

No momento seguinte a reunião toma uma direção inusitada. Começa-se a discutir sobre o “porquê os testes unitários não detectam os bugs?”, ao invés da pergunta inicial que estava tentando ser respondida “como reduzir o número de defeitos?”. Testes unitários são ótimos instrumentos para evitar defeitos, mas eles não são os únicos “responsáveis” por evitá-los. Não se pode apostar todas as fichas de qualidade do seu projeto em apenas uma prática. A qualidade é como um fluido que deve ser colocado para lubrificar cada engrenagem do processo. Uma engrenagem sem o fluido será um potencial ponto gerador de defeitos.

A reunião termina com um plano de ação delineado sem nenhuma compreensão das reais causas do problema. No cenário em questão, é bem provável que as ações tomadas gerem o efeito desejado a curto prazo. Porém, no longo prazo, quando a equipe se acomodar ao novo nível de produtividade, ou que é pior, quando ela for pressionada para a re-estabilização dos níveis de produtividade, o problema dos defeitos surgirá ainda com mais intensidade. A causa-raiz não foi resolvida.

Na parte 2 deste artigo:

A solução da Toyota para o aperfeiçoamento contínuo;

- *Identificação de problemas*

- Análise ;

- Análise da causa-raiz;

- *A técnica dos 5 porquês*

Estratégias para otimização de processos por meio de Reuniões de Retrospectiva;

Referências

AgileAlliance. (13 de Fev de 2001). *Agile Software Development Manifesto*. Fonte: <http://www.agilemanifesto.org>

Cockburn, A. (2001). *Agile Software Development*. Addison-Wesley.

Deming, W. E. (2003). *Saia da Crise*. (M. A. Mendes, Trad.) São Paulo: Editora Futura.

DSDMConsortium. (2007). Acesso em 18 de 11 de 2007, disponível em DSDM.org: <http://www.dsdm.org>

Highsmith, J. (2000). *Adaptive Software Development*. Dorset House Publishing.

Liker, J. K., & Meier, D. (2007). *O Modelo Toyota: manual de aplicação*. (L. B. Ribeiro, Trad.) Porto Alegre: Bookman.

Microsoft Corporation. (2006). *Process Templates - MSF For Agile Software Development*. Acesso em 18 de 11 de 2007, disponível em Microsoft Web Site: <http://msdn2.microsoft.com/en-us/teamsystem/aa718801.aspx>

Senge, P. (1990). *The Fifth Discipline*. New York: Doubleday.

The Eclipse Foundation. (2007). *OpenUp*. Acesso em 18 de 11 de 2007, disponível em <http://www.epfwiki.net/wikis/openup/>



Autor: Felipe Rodrigues

Felipe Rodrigues de Almeida (felipero@gmail.com) é Arquiteto Java com experiência de 5 anos em desenvolvimento de sistemas distribuídos. Atualmente trabalha na arquitetura de sistemas de GeoProcessamento e, conduz projetos e eventos na Fratech TI. Participa ativamente do desenvolvimento do framework Struts2 e mantém o projeto open-source BoxSQL (<http://boxsql.dev.java.net/>). Passa o tempo livre curtindo e cuidando de seus 4 cães.

Nos dias 2, 3, 4 e 5 de Novembro aconteceu o evento **Java Brasil 2007**, que possuía o seguinte slogan: “Conferência ágil para profissionais Java”.

Imagino que ao ler esse slogan logo abaixo de um nome tão forte como Java Brasil 2007, fica a impressão de que isso é muito mais marketing do que qualquer outra coisa. Algumas pessoas poderiam criar uma certa expectativa de que seria um evento comum, como tantos outros, falando unicamente sobre java. Outras poderiam pensar que o evento não tinha foco de assunto e que isso poderia tornar o conteúdo muito superficial.

A verdade sobre isso é que, do ponto de vista da **Fratech**, um evento sobre Java não deve ser apenas um evento sobre tecnologia, mas sim um evento sobre desenvolvimento de software e, dessa forma percebemos a necessidade de falar sobre as formas de se desenvolver software hoje em dia e consequentemente sobre desenvolvimento ágil.

O Java Brasil em 2007 começou a partir de uma idéia informal, minha e do Manoel Pimentel, sobre organização de um evento diferente no Brasil, e principalmente um evento que fosse até o público e não ao contrário. O nome inicial seria Visão Java, pois a idéia da revista Visão Ágil era muito recente. Com o tempo a idéia foi ganhando força e devido ao foco de cada um, o evento ganhou o nome de Java Brasil, pois a imagem que tínhamos era de que seria um evento itinerante.

Mas como começar? Nunca havíamos organizado um evento e não tínhamos a mínima idéia de como fazer isso. Decidi envolver a Fratech na organização pois dessa forma o evento teria uma infraestrutura para sua organização.

Mesmo com o apoio da Fratech, tivemos várias “surpresas” ao longo da nossa jornada. Dificuldades e falta de tempo prejudicaram e muito o marketing inicial do evento. Problemas de infraestrutura nos servidores de nosso antigo provedor também impediram um pouco o acesso aos site do evento. Também tivemos boas surpresas, como o interesse de várias empresas na iniciativa e a efetivação de algumas parcerias.

A parte mais fácil foi conseguir palestrantes. Como vocês já devem saber, a comunidade Java é extremamente unida e disposta a ajudar. Conseguimos alguns dos melhores profissionais do Brasil para falar de assuntos de seus respectivos domínios. Referências nacionais e internacionais sobre assuntos já comuns, fato que aumenta o mérito profissional.

Após 5 meses entre a idéia inicial e o acontecimento do evento, o resultado que conseguimos foi um evento de alta qualidade técnica, com uma organização extremamente flexível e com habilidade para as mais diversas situações ocorridas. Isso permitiu a realização de um evento orientado ao público, possibilitando um feedback imediato de boa parte dos congressistas e palestrantes. O feedback se resume nas seguintes palavras: **“Valeu muito a pena!”**.



Workshop de Struts 2 – Com Ian Roughley(Struts2 Team Leader) e Felipe Rodrigues(Fratech)

No evento foi anunciado uma estratégia de continuar a produzir eventos desse tipo no ano de 2008, e com apoio de algumas empresas, pretendemos realizar um série de eventos com formatos variados ao longo do país.

O que segue agora é um review de tudo que aconteceu nos 4 dias de evento do Java Brasil 2007.

1º Dia – Workshops

A idéia inicial deste dia era realizar 4 horas de workshop de **Struts2** e em seqüência 6 horas de **Agile Immersion**. Mas após algumas conversas com os participantes, percebemos que nem todos tinham interesse em participar do workshop de struts2. O resultado foi a realização em paralelo dos dois workshops. Dessa forma, ouvindo a opinião de cada participante, pudemos aprender sobre o foco de cada um e direcionar o evento para a satisfação geral.

Os workshops aconteceram como esperado e atenderam todas as expectativas.

Struts 2

No workshop de Struts2, que eu ministrei ao lado de **Ian Roughley**, começamos falando dos princípios básicos e comparações entre a versão antiga e a versão nova do framework. Ao terminar de falar sobre o básico, os participantes escreveram sua primeira aplicação do dia, uma aplicação de login que foi evoluída para um **CRUD** usando conceitos mais avançados.

Na seqüência introduzimos conceitos de **Model Driven Development** e **Model Driven Actions**, falando sobre best practices e design patterns. O exercício foi a migração de nosso exemplo para Model Driven.

Neste ponto passamos a utilizar conceitos de validação por XML e posteriormente por Annotations. Terminando com um formulário complexo de edição de vários objetos ao mesmo tempo.

O ponto forte do workshop foi a interação entre os participantes e os facilitadores. Resolução de customizações sugeridas por cada participante e auxílio na execução dos exercícios possibilitaram que todos os terminassem com um aplicação completa e funcional, incluindo aqueles que nunca haviam desenvolvido com o framework.

Agile Immersion

Com uma abordagem extremamente prática e dinâmica, **Manoel Pimentel** conduziu os participantes ao longo de 8 horas de workshop. Com uma seqüência de explicação e aplicação de conceitos, o workshop passou por temas como,

- **Modelagem Ágil, M3, UML em Cores, FDD, SCRUM** além de dinâmicas focadas em exposição de conceitos do dia-a-dia de um time ágil. O tópicos abordados foram:
 - Modelos de produção
 - Ciclos de vidas
 - Os 12 princípios ágeis
 - O Manifiesto Ágil
 - Conhecendo o Scrum
 - Os Papéis no Scrum
 - Product Backlog
 - Sprint Planning Meeting
 - Planning Pocker
 - Scrum Daily Meeting
 - Sprint Burndown
 - Sprint Review
 - Sprint Retrospective
 - Melhorando a engenharia através da FDD
 - M3- Modelagem Baseada em Mapas Mentais
 - UML em Cores
 - Estimativas
 - Métricas
 - Um projeto do "zero" - Dinâmica Prática de construção de um revista usando os processos ágeis.
 - Scrum com outras metodologias como XP, FDD e TOC



Dinâmica no WorkShop Agile Immersion – Com Manoel Pimentel(Visão Ágil)

Durante o workshop os participantes foram divididos em equipes onde cada equipe tinha a responsabilidade de produzir uma revista. Podendo experimentar um processo ágil com direito a todas as best practices possíveis.

Dentre os participantes encontramos pessoas altamente capacitadas e com boa experiência em projetos, utilizando metodologias ágeis constantemente, fato que só veio a reforçar a qualidade do workshop realizado. Todos os participantes afirmaram que tiveram oportunidade de visualizar erros e acertos em seus processos, assim como aprender novas abordagens para situações críticas.

Durante as pausas, entre uma atividade e outra, o comentário que se ouvia era sobre a animação do workshop. A energia dispensada pelo facilitador foi vital para o sucesso do workshop, terminando com praticamente 2 horas a mais do que o planejado, devido a iteração com o público.

Acredito que todos tenham aprendido muito e que a troca de experiências foi muito maior do que o esperado.

2º Dia – Início da Conferência

No segundo dia aconteceu uma abertura, introduzindo todos aqueles que apoiaram e tornaram possível a realização do evento. Contamos com a presença de **Ari Dias Neto** (IBM), **Manoel Pimentel** (Visão Ágil), **Ian Roughley** (Struts2 Project Leader), **Rubens** (Orolix), **Eduardo Guerra** (Mundo Java) e **Justino** (Powerlogic)

Logo após a abertura começaram as palestras, todas com alto nível de qualidade. Para manter a idéia do slogan sempre real, havia sempre opções de conteúdo ágil em todos os horários.

Houveram alguns problemas com a falta de alguns palestrantes, mas tudo foi contornado de forma sustentável. Na opinião de todos, faltou um painel com as mudanças nos horários das palestras atualizado, o que facilitaria muito a escolha da sala



Abertura da Conferência – Da esquerda para direira: Eduardo Guerra(Mundo Java), Ari Dias Neto(IBM), Felipe Rodrigues(Fratech), Manoel Pimentel(Visão Ágil), Rubens(Orolix) e Justino(PoweLogic)

Do início ao final do dia contamos com a presença de 113 pessoas participando do evento. No final do dia participamos de uma descontração, com música ao vivo e um Pub bar no hall do evento. Os palestrantes e os participantes se misturaram em conversas dos mais variados temas.

Nesta hora todos tiveram oportunidade de encontrar e fazer amigos. Foi muito interessante a participação de todos os palestrantes do dia, que ao se reunirem puderam mostrar uma outra aptidão além da técnica. Mas só quem participou pra saber. O ponto forte da noite foi a troca de conversa fiada entre participantes, organizadores e palestrantes.

3º Dia – Quase no fim

Domingo chuvoso, depois de um sábado longo, o dia começou com poucas pessoas nas salas, porém isso foi se modificando conforme o pessoal ia acordando. :-)

Como no primeiro dia, 2 palestrantes que não puderam vir, e o ilustre **Alisson Vale** nos agraciou com suas idéias propondo-se a realizar 2 palestras no lugar de Adail Muniz que não pôde comparecer por motivos familiares.

Neste dia também tivemos a palestra de **Alexandre Magno**, que explicou o **Scrum** de forma brilhante e tivemos a realização de um **workshop sobre Scrum**, ministrado pelo **Rodrigo Yoshima**. Dessa vez um workshop um pouco mais específico, visando consolidar os conceitos do desenvolvimento ágil com Scrum.



Palestra sobre Scrum – Com Rodrigo Yoshima(ASPERCOM)

O mini curso de Design Patterns também aconteceu com participação de **Felipe Rodrigues** e **Manoel Pimentel**, falando sobre modelagem ágil e simplicidade na informação a ser passada. Os participantes modelaram uma pequena aplicação usando os conceitos de UML em cores e Modelagem Ágil, conceitos explicados no Mini-Curso.

Foram apresentados conceitos sobre a separação entre as camadas de uma aplicação, mostrando como solucionar problemas específicos de implementação com alguns Design Patterns. No final do Mini-curso tivemos a participação do Ian Roughley com uma boa discussão sobre arquitetura de sistemas que utilizam componentes distribuídos como EJB3.



Palestra sobre Metodologias Ágeis – Com Allison Vale(Phidelis)

4º Dia – Seminário Powerlogic

No quarto dia tivemos a realização do III Seminário Técnico Professional Java EE Open-Source - Etapa de Campinas, onde contamos com a presença de Paulo Alvim, Diretor de Tecnologia da Powerlogic, que foi responsável por apresentar as soluções de produtividade da powerlogic que agora estão disponíveis na região de Campinas através da Fratech. Dentre os participantes tivemos a presença da Nortel, Embrapa e TRT.

Meu sentimento pessoal é de que o evento foi abrillantado pela qualidade do público presente e pela participação efetiva deste público, assim como o empenho dos profissionais que conduziram os temas. Eventos como esse trazem network profissional, possibilitando novas oportunidades na carreira, além de nos dar a oportunidade de aprender e reafirmar conceitos que diferenciam os excelentes dos bons. A frase mais ouvida no feedback dos participantes foi “Java Brasil, quem não foi, perdeu!!!”. E que venha 2008!!!

Brincando com a UML em Cores

Conheça nesse artigo, uma forma divertida e eficaz para modelagem de software.



Autor: Manoel Pimentel Medeiros

É Engenheiro de Software, com mais de 15 anos na área de TI, atualmente trabalha com projetos Java pela Rhealeza(SP), também é Diretor Editorial da Revista Visão Ágil, Colunista da Revista Java Magazine, Membro da Agile Alliance, participa do time de Desenvolvimento do NetBeans, Líder do projeto BoxSQL, Fundador do XPNorte, do NUG-BR e do PróPatterns e frequentemente palestra em eventos sobre processos e tecnologias.
Maiores informações em: <http://manoelpimentel.blogspot.com>

Introdução:

Na grande maioria dos projetos, existe uma enorme lacuna nas atividades relacionadas a engenharia de software, principalmente na disciplina de modelagem. E como essa é uma atividade primária para o bom e claro entendimento acerca do escopo de um produto, grande parte dos fracassos desses projetos, se originam exatamente nessa restrição.

É importante observar que existe um enorme paradoxo na relação de modelagem x projeto, pois quando não se peca pela ausência de técnicas de modelagem, se peca pelo excesso de artefatos e técnicas aplicados em projeto. Por isso, já existem várias soluções que visam simplificar esse processo de criação e leitura de modelos.

Também já existem uma gama de técnicas que estimulam boas práticas de modelagem em projetos, dentre algumas, podemos citar: a **FDD**(Feature Driven Development), **MDA**(Model Driven Architecture), **DDD**(Domain Driven Design) além claro, de práticas mais essenciais como a própria **Orientação a Objetos**.

Portanto, uma ótima proposta para facilitar a modelagem de software, é a UML em cores e iremos entender um pouco mais sobre seu funcionamento nos tópicos abaixo.

Histórico da UML em Cores

Resumidamente, a UML em cores foi inicialmente proposta por Peter Coad(www.petercoad.com) e teve grandes contribuições de Eric Lefebvre e Jeff De Luca , que após vários anos de experiência em modelagem de aplicações para diversos segmentos e de variados tamanhos, chegaram a conclusão que, todas as entidades encontradas nas aplicações que já haviam sido criadas, seguiam um certo padrão de existência e comportamento, com base nessas observações, foi criado um agrupamento em grandes categorias, afim de melhor identificar essas tais entidades. Essas categorias, foram chamadas de **arquétipos**, e para realmente criar uma boa separação visual entre eles, foi atribuído cores diferentes para cada um, e essas por sua vez, adicionam um fator semântico ao modelo, ajudam a diminuir a variação no processo de modelagem e padronizam o entendimento entre a equipe de negócio e a equipe de TI.

Foco no domínio

Um grande conceito relacionado a UML em cores, é o **Domain Neutral Component**, ou em bom português Domínio Neutro de Componentes, que é uma forma de criar modelos baseados no domínio da aplicação (objetos e regras de negócio, como por exemplo: Cliente, Produto, Status, Venda, Etc.), possibilitando que esses modelos, sejam totalmente independentes das outras camadas da aplicação como a de **apresentação** e de **integração**, como você pode observar na figura 01 .

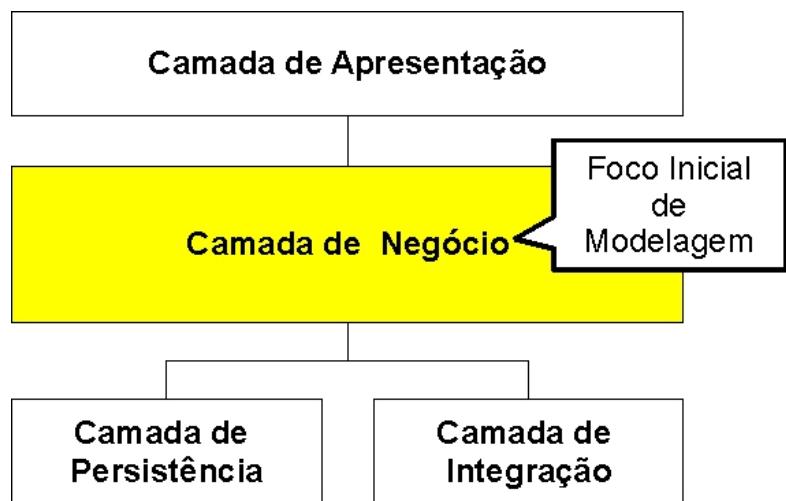


Figura 01 - Visão Arquitetural da Modelagem

Conhecendo os Quatro Arquétipos:

Observe também que conforme você pode ver na figura 02, a UML em Cores sugere o uso de **04 arquétipos**, representados por cores diferentes, e nesse tópico, você vai entender os conceitos e as aplicações típicas para cada um.

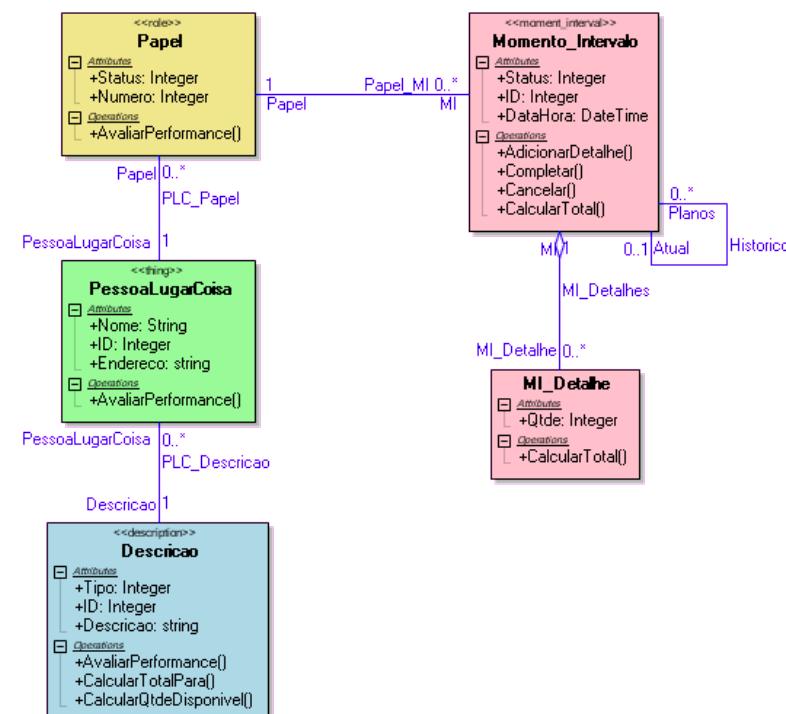


Figura 02 - Exemplos de arquétipos sugeridos pela UML em Cores

Momento-Intervalo

O primeiro arquétipo que iremos analisar, será o **Momento-Intervalo**, que é representado pela cor **vermelha** ou **rosa**, que simboliza entidades que possuem um tempo de vida bem definido, ou seja, ela nasce e deixa de existir em um espaço de tempo bem definido e momentâneo na realidade de negócio da aplicação.

Dessa forma, esse arquétipo normalmente é usado para entidades como **Pedido de Venda**, **Pedido de Compra**, **Processo de Matrícula**, **Processo de Admissão**, etc.

Uma interessante variação desse arquétipo, é o **Momento-Intervalo-Detalhe**, que representa entidades fracas que dependem de outras entidades Momento-Intervalo para sua existência, um bom exemplo disso é: **Itens de uma venda** ou **Itens de uma compras** que dependem de uma entidade do tipo pedido de compra ou de venda.

Pessoa-Lugar-Coisa

Esse arquétipo, que é representado pela cor **verde**, é uma das mais genéricas e abstratas das formas, pois é usado exatamente para representar qualquer entidade que seja uma pessoa(física ou jurídica) ou algum lugar(Departamento, Cidade, País) ou alguma coisa(Centro de Custos, Produtos, Tributos).

Papel

Arquétipo representado pela cor **amarelo**, é normalmente usado para simbolizar uma entidade que têm um **papel** muito bem definido na aplicação, exemplo: Cliente, Fornecedor, Funcionário.

Outra aplicação desse arquétipo, é para representar uma especialização de uma entidade **Pessoa-Lugar-Coisa**, por exemplo, em um software acadêmico, uma mesma **pessoa**, pode desempenhar os **papeis** de **Funcionário** e de um **Aluno**.

Descrição

Representado pela cor **azul**, o arquétipo **Descrição**, é usado para entidades fracas que servem como **meta-informações** sobre outra entidades, ou seja, é usado para definir características de uma determinada coisa.

Existem bons exemplos para aplicação de arquétipo, pois vejamos: Podemos usá-lo para entidades como **Status**(Ativo, Inativo, Cancelado) **Turno** (Manhã, Tarde, Noite) **Estado Civil** (Solteiro, Casado, Divorciado, Viúvo).

Uma boa regra que tenho adotado par usar esse arquétipo, é procurar por **entidades fracas** que tendem a não ter um grande volume de dados e sua existência seja mais estática na aplicação, ou seja, baixa ocorrência de inserções e alterações.

Exemplo Prático:

Vamos agora, montar um exemplo prático para demonstrar o uso da UML em cores para modelagem de domínio de uma aplicação de venda.

Observe que estamos usando um modelo extremamente simples, que pode ser criado usando editores de textos ou ferramentas gráficas ou ferramentas cases e o que é mais legal, é que a UML em cores, favorece o uso de post-its, o que é uma ferramenta muito prática e eficaz para a atividade de modelagem. Portanto, veja na **figura 02** abaixo, que temos algumas entidades básicas de uma aplicação de venda. Veja que nesse exemplo, criamos um modelo abrangente, sem muitos detalhes acerca dos atributos, método e dos relacionamentos de cada entidade, vale ressaltar que essa visão, segue a proposta feita pela FDD(Feature Driven Development), onde nos ciclos de concepção e planejamento, têm uma atividade chamada "Desenvolver um Modelo Abrangente", que consiste em elaborar um modelo com uma idéia inicial acerca da aplicação e a cada ciclo do projeto, esse modelo vai ganhando mais detalhes e mais aderência ao negócio da empresa, porém, como o foco desse artigo não é se aprofundar em FDD, sugiro que olhe as referências no final do mesmo, e pesquise maiores informações sobre esse tema.

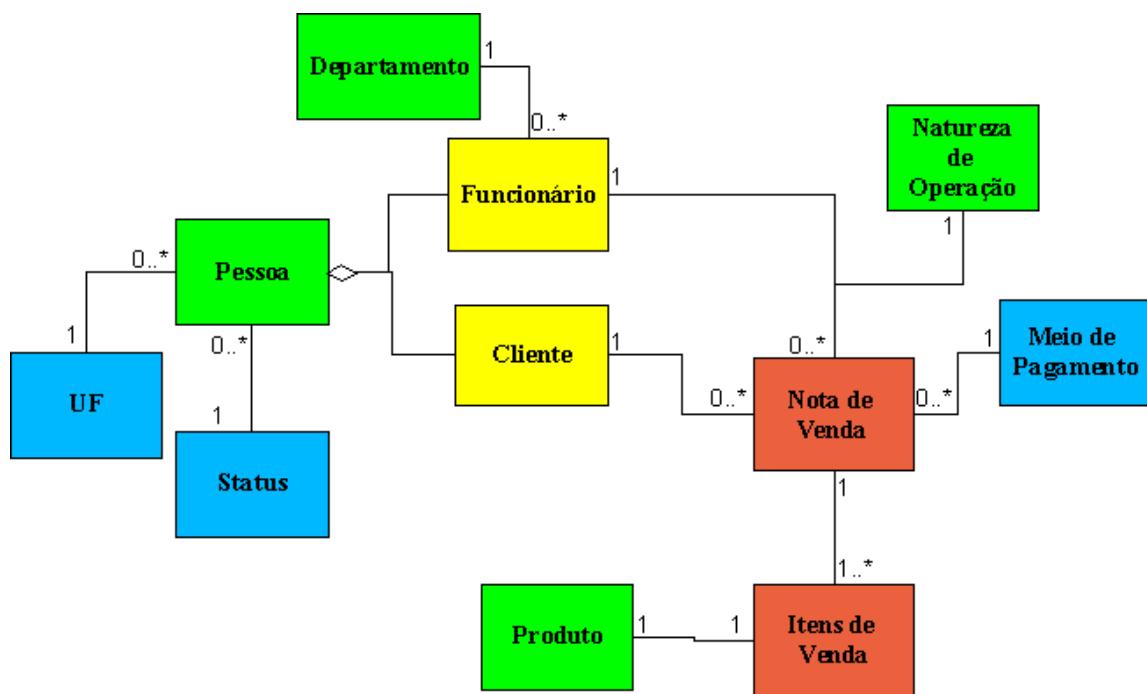


Figura 03 - Exemplo prático de um modelo usando as técnicas de UML em Cores.

Conclusão

Espero que esse artigo tenha alcançado seu objetivo principal, que é de despertar seu interesse sobre o tema, e principalmente sua curiosidade em aplicar essa técnica em seus projetos e avaliar os benefícios oferecidos por ela.

Referências

- Livro em inglês sobre UML em Cores: Coad Peter, Eric Le Febvre & Jeff De Luca, *Java Modeling in Color with UML - Enterprise Components and Process*, Prentice Hall, Upper Saddle River, NJ, 1999
- Materiais em português sobre FDD em:
<http://www.heptagon.com.br>

London Scrum Gathering

Cobertura em Londres do mais importante evento internacional sobre Scrum



Autor: Alexandre Magno

Alexandre Magno é líder de projetos de software onde utiliza principalmente metodologias e processos ágeis. Atua na área de software há mais de 15 anos, já tendo participado de projetos de variadas dimensões de lead time, escopo e investimento. Foi o primeiro Certified Scrum Practitioner da América Latina, sendo hoje membro do comitê da Scrum Alliance para avaliação de novos CSPs. Magno é o fundador do grupo Scrum-Brasil e atualmente é responsável pelos serviços de agile da Caelum (www.caelum.com.br).
Contato: axmagno@gmail.com / <http://amagno.blogspot.com>

Duas vezes por ano a Scrum Alliance(www.scrumalliance.org), organização que representa a comunidade Scrum ao redor do mundo, realiza o Scrum Gathering. Este evento, que normalmente conta com edições de primavera e outono, se propõe a ser uma grande reunião dos praticantes do Scrum espalhados pelo mundo.

A última edição deste evento foi realizada em novembro passado na capital inglesa e contou com uma média de 200 (duzentos) participantes nos 5 (cinco) dias de evento. Por mais que o predominante fossem participantes ingleses, franceses e norte-americanos, tivemos um grande número de praticantes do Egito, Alemanha, Austrália, Rússia, Turquia, Marrocos e muitos, muitos mesmo, da Índia. Infelizmente, mas como eu já imaginava, eu era o único brasileiro no evento, até porque além da distância territorial entre Brasil e Inglaterra, a distância de câmbio entre o Real e a Libra fazem com eventos nas terras da rainha sejam realmente de alto custo para nós brasileiros.

O Evento

O London Scrum Gathering foi realizado na Dexter House, uma casa de eventos ao lado da London Bridge, entre os dias 12 e 16 de novembro de 2007. Os dois primeiros dias de eventos estavam reservados para a realização do treinamento Certified Scrum Master a ser ministrado por Ken Schwaber e Mike Cohn. Para os três dias seguintes, montar uma agenda particular para o evento era um desafio para qualquer profissional interessado em Scrum. Eram muitos nomes importantes da comunidade e muitas palestras com assuntos interessantíssimos acontecendo paralelamente. O evento estava dividido em 5 (cinco) trilhas:

- Scrum in the large
- Human Side of Scrum
- Scrum Skills
- Product Owner issues
- Scrum Experiences

Além disso, um desafiador Open Space estava reservado para o último dia de evento e seria facilitado por ninguém menos que Rachel Davies, da AgileExperience. Sabine Canditt, Ken Schwaber, Kenny Rubin, Nigel Baker, Mike Cohn, Henrik Kniberg, Geoff Watts e Roman Pichler eram outros grandes nomes que tornavam mais desafiadora ainda a tarefa de escolher a qual próxima palestra assistir.

14 de novembro, o primeiro dia

O evento foi aberto através de um Keynote do Bruce Radloff da TeleAtlas, mas infelizmente não pude acompanhar visto que a minha palestra era logo na sequência, e então – além do nervosismo – eu estava ocupado deixando tudo pronto na minha sala de apresentação.



Cocktail no estilo “tea break.”

Após um tradicional “tea break” britânico – chegou a hora da verdade! A minha apresentação tinha como propósito mostrar os porquês da grande resistência sofrida pelas abordagens ágeis (com foco em Scrum) nas empresas ao redor do mundo.

Logo após o almoço, assisti a palestra “Why Scrum Projects Fail?” apresentada por Joseph Pelrini da Metaprogramming e Jiri Lundak da Löwenfels Partner (Suiça). Foi uma palestra muito legal, que tocou muito no lado humano que Scrum envolve e forneceu grandes e importantes conselhos para as Scrum Retrospectives. Na seqüência, para última palestra do dia, escolhi assistir ao Mike Cohn da Mountain Goat (USA) falando sobre o processo de transição para agile, uma palestra também recheada de dicas valiosíssimas.

15 de novembro, o segundo dia

O Keynote deste dia foi apresentado por Pirkka Palomäki da F-Secure (Finlândia) que falou sobre o desafio de espalhar Scrum por todos os cantos da empresa. Na seqüência a colega Sabine Canditt (CSP da Siemens) executou uma das mais brilhantes palestras do evento, fiquei impressionado em ver como a Siemens vem usando Scrum pra “quase” tudo.

Depois disso, sala lotada para assistir “ninguém mais – ninguém menos” que ele: Mr. Schwaber, com sua palestra “The Enterprise and Scrum”. Para quem já havia lido o seu último livro (com o mesmo título) nenhuma surpresa, mas de qualquer forma ouvir os detalhes do dia-a-dia da implantação de Scrum em uma empresa por um de seus criadores foi uma experiência ímpar. O Keynote seguinte (da British Telecom) - que



Palestra do Mike Cohn

foi excelente e divertidíssimo por sinal – era pra ter sido a última apresentação do dia, mas Ken Schwaber presenteou-nos com uma sessão “extra” de 4 horas intitulada “Scrum Update”...MAGNÍFICO! O mais legal aqui foi ver todos aqueles grandes nomes que citei anteriormente sentados no papel de aluno de Ken Schwaber, lado-a-lado com você, discutindo sobre: como aplicar, ensinar e tudo mais com Scrum.

16 de novembro, o último dia

Como eu já mencionei, o último dia do evento foi reservado para a execução de um Open Space no qual cada grupo tinha que conduzir durante um dia a difícil tarefa de organizar uma edição do evento Scrum Gathering. Os desafios eram muitos...a cada conjunto de minutos, Product Owners ilustres tais como Ken Schwaber e Kenny

PRECISANDO TREINAR A SUA EQUIPE ?

Consulte nossos pacotes corporativos!

FJ-11
Java e Orientação a Objetos



FJ-19
Preparatório para Certificação Java



FJ-21

Java para Desenvolvimento Web



FJ-26

Laboratório de MVC com Hibernate e JSF para a Web

FJ-31
Enterprise Java Beans



FJ-55
Java para pequenos dispositivos (Java ME)



FJ-28

Desenvolvimento ágil para Web 2.0 com VRaptor, Hibernate e AJAX



RR-11

Desenvolvimento ágil para Web 2.0 com Ruby on Rails

PM-81
Gerenciamento de Projetos de Software com Scrum



FJ-91

Arquitetura e Design de Projetos Java



Caelum
Ensino e Soluções em Java
www.caelum.com.br



Open Session

Rubin apresentavam obstáculos a serem conduzidos pelo Scrum Master de cada time (papel rotativo). O aprendizado nestas sessões de Open Space foi enorme e o saudosismo no final do dia foi grande.

Conclusão

Participar deste meu primeiro Gathering foi uma experiência fantástica. Por mais que meu investimento tenha sido reduzido pelo fato de ter sido palestrante, acho que mesmo se eu tivesse ido apenas como participante, ou seja, fazendo todo o investimento necessário, o evento teria me retornado cada centavo gasto (e olha que não são poucos). Mais que as palestras, que foram de alto nível mesmo, o networking e o turismo (que querendo ou não sempre é bem vindo), acho que a principal lição e o que mais valeu desta viagem foi ter visto a postura da comunidade internacional de Scrum. Fiquei impressionado com a simplicidade e “fome” de conhecimento de cada pessoa que ali conheci, e também em ver grandes nomes se colocando durante o evento inteiro no papel de aluno, assistindo palestras, perguntando...Ken Schwaber assistindo Mike Cohn, Kenny Rubin assistindo Roman Pichler,

e por aí vai. Até na minha palestra (coitado de mim) vi dois admiráveis praticantes de Scrum assistindo, perguntando e se interessando pelo tema. Além disso, e não menos importante, gostei do evento pois ali se reuniam profissionais que realmente estavam interessados na prática de Scrum, e não em discursos extremamente filosóficos de gente que nunca nem sequer se preocupou em utilizar Scrum de verdade no seu dia-a-dia. Scrum, mais do que para ensinar ou discutir em listas, é – PRINCIPALMENTE – para se praticar. Como disse Nigel Baker da BT “**Praticar mais que tagarelar!**”;

Por fim, aconselho qualquer membro da nossa comunidade a participar da próxima edição do Scrum Gathering, que acontecerá em Chicago de 14 a 16 de abril de 2008.

Mais detalhes sobre o London Scrum Gathering

<http://www.scrumalliance.org/events/2--london-scrum-gathering>

Download das palestras do London Scrum Gathering

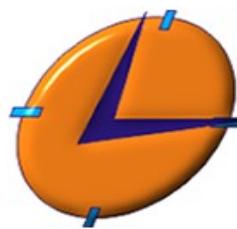
<http://www.scrumalliance.org/resources>

Chicago, a próxima parada

<http://www.scrumalliance.org/events/5--scrum-gathering>



Agenda dos Open Sessions



Apoios



www.tc4digital.com

