

Revista

Visão Ágil

O seu canal sobre processos ágeis

Integração Contínua

Álbum da Visão Ágil
no FISL 9.0

As 5 Doenças do
Gerenciamento de Projetos

Causa n° 4: Dependência Entre Tarefas

Álbum do Encontro da comunidade
de Metodologias Ágeis do RS



ScrumMaster por ele mesmo

Avalie o comportamento ideal de um
ScrumMaster e suas habilidades vitais

Nove Pecados Mortais no
Planejamento de Projetos

Aperfeiçoamento
de Projetos Ágeis - Parte II

Cobertura da QConference
2008 em Londres

Gerenciamento de resultados
com o Product Storage Chart

News e Referências Ágeis

Sprint Backlog

Conheça os temas selecionados para essa edição(iteração).



As 5 Doenças do Gerenciamento de Projetos
Causa nº 4: Dependência Entre Tarefas
Página 05 [Ir direto...](#)



Integração Contínua
Página 10 [Ir direto...](#)



Aperfeiçoamento de Projetos Ágeis Parte II
Página 18 [Ir direto...](#)



Nove Pecados Mortais no Planejamento de Projetos
Página 27 [Ir direto...](#)



ScrumMaster por ele mesmo
Página 32 [Ir direto...](#)

News

Página 04 [Ir direto...](#)

Referências Ágeis

Página 09 [Ir direto...](#)

Álbum do Encontro da comunidade de Metodologias Ágeis do RS

Página 17 [Ir direto...](#)

Álbum da Visão Ágil no FISL 9.0

Página 52 [Ir direto...](#)



Cobertura da QConference 2008 em Londres
Página 44 [Ir direto...](#)



Gerenciamento de resultados com o Product Storage Chart
Página 47 [Ir direto...](#)

Editorial

Agile como vantagem competitiva

Equipe Visão Ágil

Diretor Editorial:

Manoel Pimentel Medeiros, CSP

Diretor de Marketing:

Alexandre Magno Figueiredo, CSP

Assessoria Administrativa:

Manuella Bulcão Medeiros

Atendimento ao leitor

e-mail: manoel@visaoagil.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 ágeis publicados em nossa revista, por favor envie-os para manoel@visaoagil.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 amagno@visaoagil.com, e teremos o maior prazer em trabalharmos juntos.

No mundo inteiro, estamos vivendo um momento interessante no cenário Agile, pois cada vez mais temos novas empresas adotando algum processo ágil, um número maior de profissionais está se dedicando a consultoria especializada sobre o assunto, muitos blogs têm surgido para trazer temas relacionados aos processos ágeis, as comunidades estão ficando cada vez maiores e mais participativas.

Mas ao contrário do que se pensa com esse cenário, Agile ainda não é "bola da vez", pois ainda falta muita coisa acontecer para que isso seja uma verdade absoluta e isso dependerá de um grande esforço conjunto de toda a comunidade. Porém, nos últimos anos fui consolidando uma visão particular sobre o mercado de desenvolvimento de software, pois vejo que a mecânica desse mercado se assemelha ao conceito "**Yin & Yang**", ou seja, "**só há Agilidade se houver Cascata**".

Traduzindo: NÃO acredito que a maioria das empresas irá adotar Agile em seus processos, e na verdade acho isso MUITO BOM, pois dessa forma, as empresas que realmente estão adotando, continuarão a ter uma vantagem competitiva sobre as que não adotam.

Na verdade ficaria muito chato se todas as empresas tivessem a mesma produtividade e trabalhassem seguindo as mesmas idéias, pois a beleza de usar agile é exatamente essa: "mostrar que é possível ser mais produtivo que seus concorrentes".

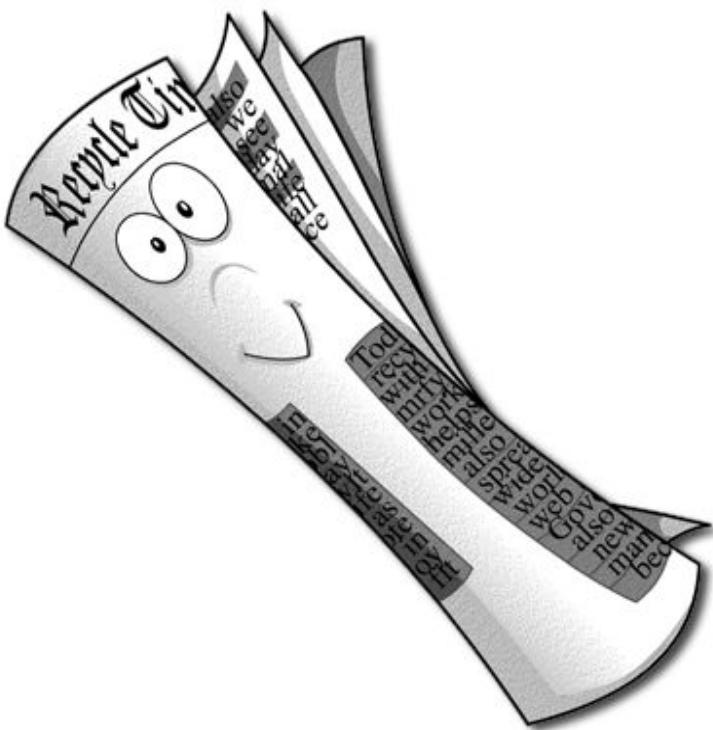
Porém, caso você também queira que sua organização mude a forma de trabalhar e passe ter benefícios reais com a implantação de agile, nessa quarta edição da Revista Visão Ágil, estamos com uma coleção de artigos bem especiais para "**turbinar**" a caixa de ferramentas de quem está implantando alguma metodologia ágil em uma organização, portanto, seja bem-vindo e boa leitura.

Manoel Pimentel, CSP

Diretor Editorial



**Revista
Visão Ágil**



News

Veja aqui as novidades
do mundo ágil

InfoQ no Brasil

A **Fratech Tecnologia** está em negociação para firmar uma parceria estratégica com a **InfoQ** aqui no Brasil.

Maiores informações: www.fratech.net

Agile 2008

Em agosto acontecerá em Toronto mais uma edição do principal evento ágil do mundo, o Agile 2008.

www.agile2008.org

Visão Ágil 2008

A Revista Visão Ágil junto aos grupos de usuários, realizará nos dias 19 e 20 de Setembro a conferência sobre processos ágeis com profissionais do cenário nacional e internacional.
Participe: www.visaoagil.com

Workshop Scrum com Rodrigo Yoshima

31 de maio em Belo Horizonte/MG

www.aspercom.com.br

Palestra “Was Socrates a ScrumMaster?” com Alexandre Magno

NYC Scrum Group - New York, USA
20 de julho

amagno.blogspot.com

Ciclo de treinamentos de Gestão Ágil daTeamware

Novas turmas disponíveis em várias cidades do Brasil

www.teamware.com.br

As 5 Doenças do Gerenciamento de Projetos

Causa n° 4: Dependência Entre Tarefas



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.

Em projetos, todas as tarefas dependem de outras. Em seu livro "The Mythical Man Month" ("O Mito do Homem-Mês"), Fred Brooks respondeu à pergunta sobre como os projetos atrasam: "Um dia por vez". Você já notou que se a data final do seu projeto desliza, é quase impossível recuperá-la? Você também já notou o quanto fácil é atrasar, mas o quanto é difícil adiantar? Se já, então você entende os problemas criados pela dependência entre tarefas.

Um efeito negativo causado pela dependência entre tarefas é explicado no seguinte exemplo. Você tem uma tarefa que foi estimada em 5 dias, incluindo a segurança, a inicia imediatamente e a completa "mais cedo". A pessoa que recebe sua saída está pronta para usá-la *imediatamente*? Geralmente não. Portanto, se você entregar os resultados em 3 dias, a próxima pessoa não vai tocá-los por 2 dias adicionais, porque ela não está agendada para começar a tarefa dela até aquele dia. Assim, a segurança embutida é perdida, mesmo com a tarefa sendo terminada antes. Para superar esse problema **você precisa ter um sistema de projetos que garanta que todas as tarefas começem, não quando elas estão agendadas para começar, mas quando as entradas necessárias estiverem disponíveis**. Isso é especialmente vital com as tarefas no caminho crítico (ou na corrente crítica).

Outro efeito negativo causado pela dependência entre tarefas é bem conhecido na teoria da probabilidade, chamado de "probabilidade de eventos dependentes" (também conhecido por outros nomes). Essa teoria afirma que o tempo total requerido para eventos dependentes, em termos de probabilidade, é o produto da probabilidade de todos os eventos dependentes. Veja como isso impacta você (Figura 7). Se você tem três tarefas que são dependentes uma da outra, e cada uma tem uma chance de 90% de ser terminada no prazo, qual é a probabilidade de todas as três serem completadas no prazo? Cerca de 73%! Devemos calcular a probabilidade de terminar a Tarefa-1 (90%) e depois calcular a probabilidade de terminar a Tarefa-2, dada sua dependência com a Tarefa-1 ($90\% \times 90\% = 81\%$). Como pode ver, a probabilidade de terminar a Tarefa-1 e a Tarefa-2 no prazo é de apenas 81%. Podemos então calcular a probabilidade de completar a Tarefa-3, dada sua dependência com a Tarefa-1 e a Tarefa-2 serem completadas no prazo ($90\% \times 90\% \times 90\% = 73\%$). Com apenas três tarefas, cada uma com 90% de chance de terminar como prometido, há apenas uma chance de 73% de terminarmos, de fato, todas as três como prometido. Não precisamos de muitas tarefas para alcançar uma probabilidade zero de terminar o projeto no prazo.

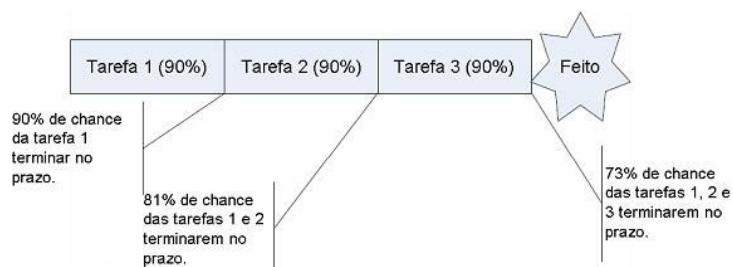


Figura 7



Figura 8

Você pode estar pensando: "Eu não tenho esse problema, porque eu realizo as tarefas em paralelo, em vez de em série." Vamos examinar esta solução (Figura 8). Se cada tarefa possui uma chance de 90% de ser feita como planejado, qual é a chance do projeto todo terminar no prazo? Sua primeira reação pode ser 90%. Porém, já que o término do projeto depende do término de todas as tarefas, devemos usar o produto de cada evento dependente para determinar o tempo provável de término. Neste caso, vamos multiplicar 90% de cada tarefa em paralelo, obtendo a mesma chance de 73% de término no prazo. Agora você pode ver porque tentar fazer com que cada tarefa termine no prazo não significa que o projeto terminará no prazo. Será exigido que toda tarefa dependente termine como planejado, para que o projeto termine no prazo. Esse efeito fez com que os gerentes de projetos concluíssem que a única forma de terminar um projeto no prazo é garantir que todas as tarefas, realmente, terminem no prazo. Tal solução exigiria que todas as tarefas sejam estimadas com 100% de precisão. Mesmo se isso fosse possível, faria com que os tempos das tarefas tivessem uma duração inimaginável.

Para entender melhor como os atrasos são passados adiante, mas os adiantamentos não, examine a Figura 9. Este projeto foi planejado para durar 17 dias.

O quanto adiantado ele poderia terminar se a primeira tarefa estivesse cinco dias adiantada? Os mesmos 17 dias. A razão é que nós somos dependentes do término de todas as cinco tarefas. Portanto, mesmo se uma tarefa estiver adiantada, o projeto não terminará adiantado em nada. E se as primeiras quatro tarefas terminassem 5 dias antes? A duração do projeto ainda seria de 17 dias. Agora pense na duração do projeto se apenas uma tarefa atrasar 5 dias. O projeto levaria 22 dias. Como podemos ver, adiantamentos não são passados adiante, atrasos sempre são.

Uma dependência adicional não discutida, e geralmente proibida na metodologia do caminho crítico, é a dependência de recursos. Na imagem seguinte (Figura 10), temos um projeto que inclui dependências de tarefas e de recursos. Note que as tarefas A e D ambas requerem o uso do recurso "programador". Para completar as tarefas C e D temos que completar A e B. Para iniciar a tarefa D também precisamos completar a tarefa A, devido à dependência de recurso. Dado que cada tarefa possui uma probabilidade de terminar no prazo de 90%, qual é a probabilidade deste projeto terminar no prazo? Apenas 73%! Embora não haja uma seta ligando a tarefa A à tarefa D, a dependência permanece. Com essa dependência encontramos todos os problemas mencionados previamente. Entretanto, dependências de recursos não são calculadas em redes de projetos tradicionais.

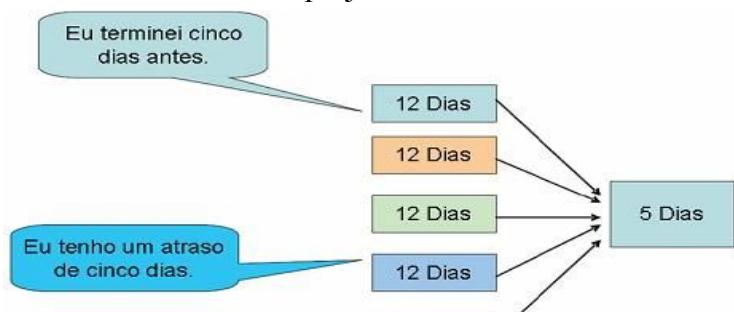


Figura 10

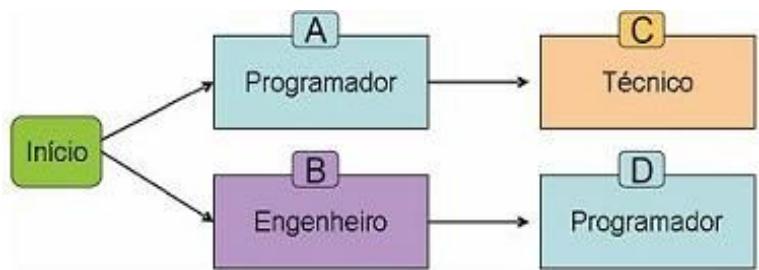


Figura 10

Vamos examinar o ciclo vicioso de lógica que ocorre devido ao fato de fazer as estimativas se tornarem compromissos. Lembre-se que as pessoas querem ser vistas como confiáveis e, portanto, tentam completar as tarefas o mais próximo da data compromissada quanto possível. Isto impede que a proteção seja removida nas estimativas futuras e as ajuda a serem vistas como confiáveis.

Quando um projeto está atrasado, qual é a resposta típica? Conduza uma atividade de lições aprendidas e identifique as tarefas que atrasaram o projeto. Qual é o comportamento criado por essa descoberta? Da próxima vez que criarmos um plano de projeto, adicionaremos mais segurança e detalhes a esses tipos de tarefas. Porém, se adicionarmos mais segurança e não mudarmos como medimos as pessoas, para permitir que os términos antecipados e atrasados se compensem entre si, então o projeto não terminará "no prazo" da próxima vez. Igualmente, se nas tarefas culpadas por atrasar o projeto adicionarem mais segurança, esse projeto será planejado para durar ainda mais. Se os ganhos e os términos antecipados não se compensarem e o projeto ainda não terminar no prazo, nas lições aprendidas descobriremos quais tarefas atrasaram o projeto.

Como você pode ver, isso torna-se um laço negativo. Estamos atrasados, culpamos tarefas, adicionamos mais segurança, ficamos atrasados de novo, culpamos tarefas, adicionamos segurança, e assim por diante. Eventualmente, seus clientes procurarão seu concorrente, porque seus projetos demoram demais. Seus clientes determinam o quanto você pode aumentar o cronograma por conta do gerenciamento deficiente.

Uma última questão relacionada com tarefas que se integram umas com as outras. Qualquer lugar em que há integração de tarefas há riscos adicionais. As coisas nem sempre se encaixam como esperamos. O que acontece quando tarefas que se integram têm problemas? O projeto é atrasado. Sim, você poderia colocar mais segurança nesses pontos, mas ela seria desperdiçada como já descrito. Esse problema pode ser superado apenas por um método de agendamento que considere a variação e a incerteza nas durações das tarefas. Ele não pode ser superado simplesmente declarando que as pessoas "devem" terminar no prazo. O método de agendamento que gerencia essas questões é chamado Corrente Crítica. Encorajo os leitores a lerem "Corrente Crítica", de Eliyahu Goldratt, para mais informação sobre o assunto.

Workshop sobre Modelagem Ágil e DDD

- Um passeio pela MDA (Model Driven Architecture)
- Conhecendo a DDD (Domain Driven Design)
 - O que é DDD
 - Para que usar DDD
- Arquitetura em Camadas (Layered Architecture)
 - Domain Objects
- Linguagem Onipresente (Ubiquitous Language)
 - Design Flexível(Supple Design)
- Design Estratégico (Strategic Design)
 - O Manifesto Ágil
 - Engenharia de requisitos com Scrum, XP e FDD
 - Testes Ágeis
 - Documentação Ágil
 - Explorando a visão arquitetural
- M3 - Modelagem Baseada em Mapas Mentais
 - UML em Cores
 - Uso de prototipação
 - AgileDraw

Organização:

- Um projeto do "zero" - Dinâmicas Práticas



Informações pelo site:

www.fratech.net

Carga Horária: 16h -

Data: 13 a 14 de Junho/2008

Local: Universidade Anhembi Morumbi

São Paulo(SP)



O seu canal
sobre processos ágeis



Referências Ágeis

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

The screenshot shows a Mozilla Firefox browser window with the URL <http://www.phidelis.com.br/blogs/alissonvale>. The page title is "Artigo Publicado na Revista Visão Ágil". The main content area features a large image of Alisson Vale and text about his article in the magazine. To the right, there is a sidebar with a "Sobre o Autor" section containing a photo and bio, a search bar, and a calendar for April 2008.

Blog de Alisson Vale, Um dos pioneiros na utilização e divulgação de métodos ágeis aqui no Brasil. Devido sua grande experiência como líder e diretor de projetos da empresa Phidelis, tornou-se um dos mais respeitados autores nacionais sobre agile e afins.

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

The screenshot shows a Mozilla Firefox browser window with the URL <http://blog.mountaingoatsoftware.com/>. The page title is "Mike Cohn's Blog - Succeeding With Agile™". The main content area features a large image of Mike Cohn and a post titled "Prioritizing Tasks Within a Sprint". The sidebar includes a search bar, a logo for Mountain Goat Software, and an "ARCHIVES" section with links to various months from 2007 to 2008.

Blog do renomado **Mike Cohn**, um dos personagens internacionais mais bem conceituado quando o assunto é a aplicação da metodologia Scrum em um projeto de software.

URL: blog.mountaingoatsoftware.com

Integração Contínua

Seu sistema ao alcance de um clique



Autor: Victor Hugo Germano

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

Normalmente subestimado pelas equipes, o período de integração do código e dos subsistemas dentro de um projeto pode facilmente ser o principal vilão na liberação de um release. Esta falta de preocupação em criar, o quanto antes, um produto finalizado levará a atrasos em entregas e desculpas comuns como "*mas funcionava na minha máquina*".

Em uma equipe com vários desenvolvedores, qual a melhor forma de unificar as diversas alterações feitas na base de código? Processos Ágeis utilizam a prática conhecida como **Integração Contínua** para solucionar essa questão.

Integração Contínua consiste em integrar o trabalho diversas vezes ao dia, assegurando que a base de código permaneça consistente ao final de cada integração. Nesse artigo, você conhecerá os passos necessários para usar essa prática em seu dia-a-dia, e quais os benefícios para a sua saúde e a saúde de seu projeto.

O que é Integração Contínua?

O processo de integrar um software não é, nem de longe, um problema novo, ainda que não seja um problema real em um projeto *solo*, com poucas dependências de sistemas externos. Entretanto, à medida que a complexidade do projeto aumenta (adicionando mais uma pessoa, por exemplo), é preciso garantir que os componentes de software funcionem juntos - precoce e **frequentemente**. Esperar até o final de um projeto para integrar seu código pode levar a toda sorte de problemas de qualidade, que são extremamente caros e normalmente levam os projetos ao atraso. A Integração Contínua (IC) trata estes riscos em pequenos incrementos.

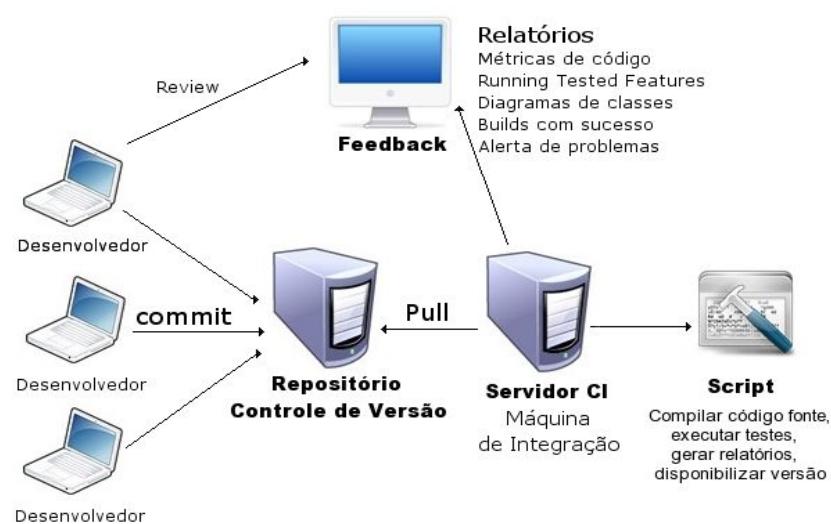
Em seu famoso artigo "**Continuous Integration**", Martin Fowler descreve a IC como:

"... uma prática de desenvolvimento de software onde os membros de uma equipe integram seu trabalho frequentemente, normalmente ao menos uma vez ao dia para cada pessoa - levando a múltiplas integrações diárias. Cada uma destas integrações é verificada por um processo de build automatizado (incluindo testes) para detectar erros de integração o mais rápido possível. Muitas equipes já identificaram que esta abordagem leva à redução significativa de problemas de integração e permite à equipe desenvolver um software coeso mais rapidamente..."

Isto significa que, segundo Paul Duval:

- Todos os desenvolvedores executam a construção do sistema **antes** de enviar qualquer código ao repositório de arquivos para assegurarem-se que suas mudanças não quebraram o build de integração.
- Desenvolvedores sincronizarão seus código com o repositório de arquivo *ao menos uma vez ao dia*.
- *Builds de Integração* ocorrem várias vezes ao dia num servidor em separado .
- 100% dos testes devem passar.
- Um produto será gerado (ex: WAR, executável, JAR, etc.) que poderá ser funcionalmente testado.
- Qualquer problema no processo de construção automática **possui prioridade máxima**.
- Alguns dos desenvolvedores revisam os relatórios gerados no processo de construção, como métricas para padrões de código e relatórios de análise de dependência, buscando áreas para melhoria.

O processo



Paul Duval, em seu livro **Continuous Integration**, descreve o cenário acima da seguinte maneira:

- O desenvolvedor envia o código para o controle de versão (SVN, CVS, etc). Enquanto isso a máquina de integração (servidor de Integração Contínua) está "ouvindo" o repositório buscando por modificações.
- Logo após um commit ser efetuado, o servidor identificará a mudança no código. Inicia-se o processo de build baixando os arquivos do Servidor de Controle de Versão. Assim, o script de build é executado, testes são realizados, relatórios gerados, e todo o projeto é integrado.
- O Servidor de Integração envia por e-mail ou outros dispositivos o feedback sobre o build para usuários específicos do projeto.
- O servidor volta ao estado de Pull buscando por mudanças no repositório.
- Um comportamento importantíssimo: Esta arquitetura força a equipe a manter sempre em funcionamento todo o código fonte que existir no repositório, adicionando mais controle a possíveis problemas de implementação ou integração.

Identificamos cinco fases principais:

Compilação

Praticamente o sinônimo de Integração Contínua, já que é o elemento mais básico de todo o processo de construção e integração. A geração de código binário é requisito para o sucesso das outras fases. Obviamente, é normal que em linguagens dinâmicas como Python, PHP ou Ruby esta fase se torne nebulosa, e o processo de criação de um binário possa ser substituído por uma atenção mais minuciosa à fase de teste e inspeção.

Teste

Muitos consideram Integração Contínua sem testes automatizados uma farsa. E não poderia deixar de concordar. Sem a validação automatizada de requisitos de negócio, funcionalidades e correções de bugs, não há como garantir a confiabilidade do software. A maior parte dos desenvolvedores em um ambiente de IC utiliza ferramentas como JUnit, NUnit ou qualquer outra xUnit para executar testes. Entretanto, é possível que esta fase da construção do software se encarregue de validar vários outros aspectos, não apenas testes unitários. Podemos citar: testes de carga/performance, segurança, sistema e requisitos de negócio.

Inspeção

Automatizar a inspeção de código, por exemplo análise dinâmica ou estática pode ser utilizado para auxiliar na melhoria da qualidade do software através do estabelecimento de regras. Por exemplo, um projeto pode determinar que nenhuma classe deverá possuir mais de 300 linhas de código não comentado, ou que os testes devem cobrir ao menos 90% das linhas de código. A intenção é remover a intervenção humana neste processo, reduzindo custos - atualmente atribuídos à equipe de **Quality Assurance** -, e principalmente evitando o desgaste natural que pode acontecer. A única intervenção humana neste caso é feita para a configuração de regras.

Implantação

É a forma de garantir, a qualquer momento, a disponibilidade de uma versão compilada, testada e pronta para o uso. Este é o ponto principal de todo o processo de Integração Contínua. Este é o foco: criar, várias vezes ao dia, uma versão estável, inspecionada por métricas de qualidade e testada, pronta para se avaliada pelo cliente. Que tal, útil?

Feedback Contínuo e Documentação

A dificuldade em criar documentações por parte dos desenvolvedores pode ser diminuída - seja o problema em *escrever* ao invés de *codificar*, ou o custo em manter diagramas e documentações de código em tempo real. JavaDoc, NDoc, Maven e doxygen são exemplos muito úteis para agilizar a produção de documentações.

A geração de relatórios e métricas automatizadas será o ponto principal na melhoria da qualidade de software. Esqueça pontos de função finalizados ou linhas de código produzido. Vá além! De forma automatizada é possível visualizar o acoplamento de seu sistema, dependência das classes, número de linha de código por método, conformidade com padrões de formatação e design, quantidade de códigos com problemas no repositório ou todas as métricas propostas por Bob Martin no livro *Agile Software Development*. Estes podem ser os principais insumos para uma análise mais profunda e realmente relacionada à produção de softwares com qualidade.

O Valor da Integração Contínua

Redução de Riscos

Integrando seu código várias vezes ao dia, você pode reduzir alguns dos riscos de seu projeto, facilitando detecção de defeitos, a mensuração da saúde de seu código e reduzindo o números de incertezas.

Erros podem ser encontrados *no momento que eles são introduzidos ao sistema*, ao invés de esperar algumas semanas por um ciclo de uma área de testes. Incorporando testes e inspeções contínuas em seu sistema, a saúde de muitos atributos de software, como complexidade, pode ser rastreada no decorrer do tempo. Assim, diminua incertezas sobre ambientes e casos específicos: implemente-os em seu servidor de build e obtenha o resultado imediato!

Redução de Processos Repetitivos

Processos repetitivos executados por seres humanos, em nosso ambiente, devem ser considerados desperdício! O Custo, esforço e tempo empregados poderiam ser utilizados para o processo criativo de produção de software. Parece óbvio, certo? Mas pense bem sobre suas atividades diárias. Quais delas poderiam ser automatizadas? Quanto tempo você dispõe preparando um ambiente para produção? Quanto tempo demora para apresentar o seu sistema para um novo cliente?

Gerar software funcionando

Integração Contínua permitirá que você entregue software em funcionamento *a qualquer momento*. Do ponto de vista do usuário ou cliente, este é o maior benefício da IC. Assim, adicionar pequenos incrementos de software a um sistema pode ser facilmente incorporado numa política de apresentação de funcionalidades. E mais ainda, implantados em produção imediatamente! Projetos que não utilizam essa prática podem sofrer de muitos atrasos, pois será sempre necessário preparar um ambiente para validar a produção do software, sempre ao final de um ciclo de desenvolvimento.

Ampliar a Visibilidade do Projeto

A IC ampliará suas possibilidades de inovações e experimentação. Projetos podem sofrer por não possuírem dados concretos para a tomada de decisão, já que normalmente decisões sobre código e arquitetura são tomadas baseadas no sentimento dos desenvolvedores, não em métricas. Tentar conseguir informações manualmente pode facilmente levar à inutilização de qualquer documentação no tempo, devido à dificuldade de aferir as métricas propostas neste artigo.

Assim, uma equipe poderá partir para uma implementação arriscada, ou ainda tentar a utilização de novas tecnologias sem que isso comprometa tudo o que havia sido produzido.

Ampliar a Confiança no Produto

Poucos foram os projetos em que vi desenvolvedores com certeza sobre: a robustez de suas soluções, sua capacidade de estimativa de esforço ou a confiabilidade em entregar um sistema que funcione quanto em um ambiente com IC. Experimente!

Por onde começar?

Controle de Versão

Pode parecer mentira, mas existem muitas empresas que ainda não possuem um servidor de controle de versão para o código em desenvolvimento. Irresponsabilidade? Ignorância? Espero que não.

Mantenha seu código organizado e em um único lugar e gerencie suas versões através de um Servidor de Controle de Versões. Assim, os desenvolvedores poderão trabalhar em paralelo, e uma parte do trabalho de integração de código poderá ser realizado pela ferramenta. Existem muitas aplicações no mercado, mas vou citar duas: CVS e sua evolução SVN, ambos de código aberto. **Cuidado:** se você ainda não possui uma ferramenta dessas em seu ambiente de trabalho, deveria estar preocupado.

Servidor de Build

Entre as funções de um servidor de build, o mínimo que você deve se preocupar:

- Recuperar as últimas alterações no repositório de arquivos em algum intervalo de tempo.
- Executar ações baseadas num cronograma, como diariamente ou a cada hora.
- Suporte para diferentes ferramentas de *scripting*, incluindo as executadas em linha de comando: Rake, make, Ant ou NAnt.
- Enviar emails ao interessados do projeto com informações sobre o build.
- Armazenar um histórico dos builds
- Possuir uma interface web acessível a qualquer um.

Informações sobre qual ferramenta utilizar, e as funcionalidades de cada uma podem ser encontrados na Tabela de Funcionalidade de IC (CI Feature Matrix), disponível na sessão de links. CruiseControl, Anthill, Gump e CI Factory são alguns exemplos de ferramentas.

Crie um desafio em seu projeto: ***Poderíamos automatizar a compilação de um de nossos sistemas?***

Invista tempo em aprender sobre ferramentas de script e como utilizá-las em seu ambiente. Assim, a escolha de um servidor de build será bastante simples.

Testes

Sou adepto da idéia de Bob Martin sobre profissionalismo em desenvolvimento de software:

"Nos dias de hoje, é irresponsabilidade um desenvolvedor entregar qualquer linha de código sem que exista um teste unitário para comprovar o seu funcionamento"

Mesmo sendo extremista, acredito que este deve ser o direcionamento. Mas como reunir seus testes em um ambiente de Integração Contínua? A família de ferramentas xUnit pode ser bastante útil na criação da IC. Entretanto, testes unitários podem não ser suficientes, já que também é possível verificar aderência de regras de negócio e carga, por exemplo. O Selenium e o JWebUnit podem suprir a necessidade de testar interfaces web, servindo muito bem para validação de regras de negócio. Todas essas ferramentas devem ser integradas juntamente com o processo de construção via script.

Inspeção de Código

Decida o que você quer medir antes de mais nada. Cobertura de código, complexidade e duplicações podem ser úteis para sua equipe. Checkstyle é uma ferramenta que permite verificar a aderência de seu código a padrões de desenvolvimento, podendo também encontrar redundâncias e fazer inspeções em nível de design e complexidade de código.

Você pode remover a complexidade de seu código ficando atento a:

- O número de classes, métodos, linhas de código não comentadas e variação do estilo de documentação entre os pacotes (um padrão JavaDoc pode ajudar).
- Complexidade ciclomática.
- Dependências.
- Acoplamento.

Crie a cultura de avaliação contínua sobre a arquitetura de seu código, assim os insumos gerados por uma ferramenta como o Checkstyle ou NDepend podem contribuir e muito para a saúde de seu sistema.

Feedback

Não desperdice seu tempo criando relatórios manualmente. Encontre a documentação exata de seu código, extraia a arquitetura em que o sistema está realmente construído. Tome decisões de desenvolvimento baseadas por fatos e números, não suposições.

"Como regra geral, o homem mais bem sucedido em vida é aqueles que possuí as melhores informações"

Benjamin Disraeli

Receba informações sobre problemas no momento em que eles surgirem. Deixe visível para toda a equipe o momento em que o seu software deixou de ser confiável. Tome medidas rápidas para corrigir o problema. Este é um exemplo bastante nítido do princípio *Stop the line!* do Lean.

Utilize com cautela Emails, Instant Messagers, Sirenes e SMS para retirar o melhor proveito da infraestrutura de feedback criada pela IC.

Conclusões

Mesmo não sendo a panacéia para todos os problemas de software, o ambiente proposto neste artigo espera ajudar a melhoria da qualidade dos projetos através de métricas e números reais sobre o estado de seu sistema. Encontrar a resposta sobre "Onde estamos" é o primeiro passo para ampliar a sua capacidade de produzir sistemas com melhor qualidade.

Focar-se na produção de software é uma mudança muito grande na cultura de nossas empresas. Transformar os desenvolvedores em avaliadores de qualidade trará benefícios muito além da construção de produtos mais confiáveis.

Um ambiente de Integração Contínua modificará a forma com que seu código é produzido e a forma com que os envolvidos no projeto lidarão com problemas de desenvolvimento.

Ao aventurar-se neste campo, esteja preparado para questionar todas as suas atividades e todo o processo de condução de um projeto de software. Boa sorte!

Referências e Links

- O artigo mais clássico sobre o tema:
<http://martinfowler.com/articles/continuousIntegration.html>
- Livro: Continuous Integration - Improving Software Quality and Reducing Risk - Paul Duvall, Steve Matyas e Andrew Glover
- Continuous Integration Feature Matrix -
<http://confluence.public.thoughtworks.org/display/C/CI+Feature+Matrix>
- Blog do Autor: <http://malditacomedia.blogspot.com>
- Foto por Manu Contreras



Revista Visão Ágil

Artigos

Cases

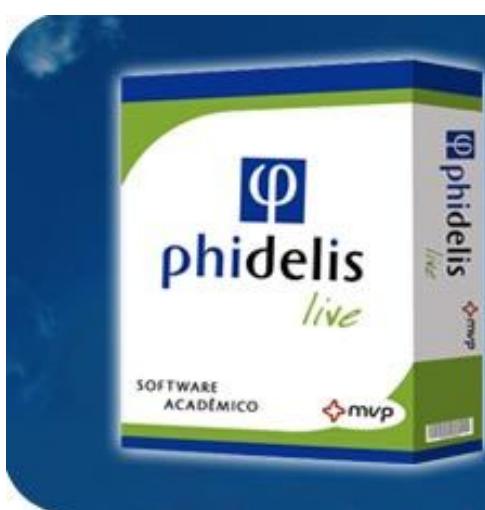
Comunidade

Biblioteca Pública

Eventos

Tudo sobre Agile

www.visaoagil.com

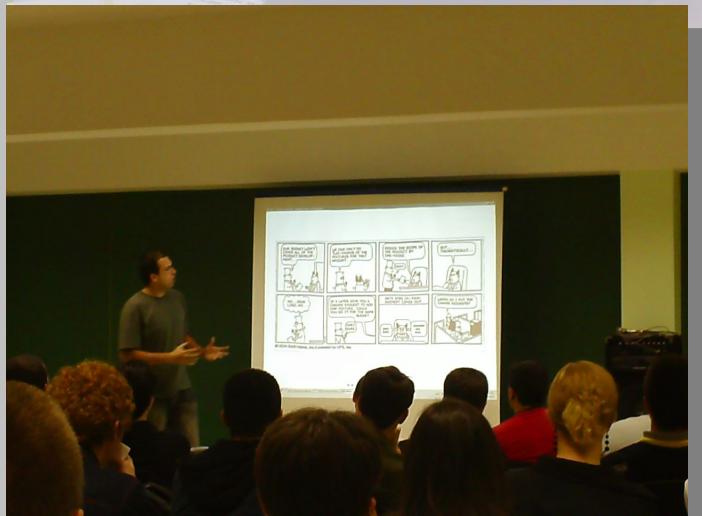


<http://www.phidelis.com.br>

Tecnologia em Gestão Acadêmica
ao alcance de todos

Encontro da comunidade de Metodologias Ágeis do RS

Em Porto Alegre(RS), Na noite de 17 de Abril de 2008, nas dependências da Faculdade Dom Bosco, foi organizado pelo pessoal do XP-RS(Grupo de Usuários Extreme Programming do Rio Grande do Sul), um encontro para trazer palestras e debates acerca de Metodologias Ágeis. Nós da Visão Ágil, aproveitando nossa estadia em Porto Alegre em função do FISL, participamos do evento e cobrimos os melhores momentos para você amigo leitor.



Daniel Wildt na palestra inicial sobre eXtreme Programming: princípios, valores e práticas.



O público compareceu em peso ao evento, com bastante interesse, muitas perguntas e bastante participativo.



Manoel Pimentel fazendo uma breve apresentação institucional sobre a Revista Visão Ágil.



Vinícius Manhães Teles, No início de sua apresentação no estilo “Tropa de elite” sobre eXperiências Ágeis

Aperfeiçoamento de Projetos Ágeis

Parte II: As Três Perspectivas



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>

Não existe processo perfeito. Isso é fato. No entanto, buscar a perfeição, mesmo sabendo que nunca iremos alcançá-la faz parte da essência do ser humano. Todos nós lutamos para melhorar nossas condições de trabalho, nossa eficiência e nossos resultados. O que muda de uma pessoa para outra (ou de um projeto para outro) é a freqüência com que paramos para analisar o que fazemos, buscando formas de sermos melhores hoje do que éramos ontem. Alguns o fazem todos os dias, outros todas as semanas ou todos os anos. A grande questão é: como podemos fazer isso de modo estruturado? Nesse artigo, eu descrevo formas de abordar e aplicar modelos de aperfeiçoamento contínuo em projetos de software baseados em práticas complementares, capazes de atuar sob o prisma de três perspectivas diferentes.

Inspeção e Adaptação são duas faces de uma mesma moeda em projetos ágeis. A Inspeção direciona nossa atenção para questões relacionadas com o nosso modo de trabalhar que de alguma forma interferem nos resultados que obtemos . Quando fazemos inspeção olhamos de dentro para fora, ou seja, algo dentro do dia-a-dia do projeto precisa ser melhorado para que quem esteja do lado de fora perceba melhores resultados. Quando falamos em adaptação, no entanto, nossa atenção é direcionada para fora do controle do nosso projeto. Ela precisa aparecer quando eventos ou situações originadas fora do âmbito do projeto acabam interferindo em seus resultados.

Nesse caso precisamos nos adaptar a um cenário diferente daquele que estamos habituados. É comum não percebemos essa diferença pois, além de ser útil, ela acaba se tornando irrelevante já que o instrumento que possibilita a inspeção e a adaptação é o mesmo: o aperfeiçoamento contínuo. São as técnicas de aperfeiçoamento contínuo que darão a capacidade necessária para que um time de projeto tenha condições de auto-inspecionar seu próprio processo de trabalho. E serão as mesmas técnicas que darão a capacidade para que esse mesmo grupo consiga se adaptar às questões externas que inevitavelmente surgirão no decorrer do projeto.

Neste artigo, eu apresento algumas técnicas e as organizo no que eu costumo chamar de “Perspectivas”. Práticas de aperfeiçoamento contínuo terão melhor eficácia quando utilizadas para endereçar problemas de acordo com o seu nível de abrangência. Por exemplo, problemas de natureza sistêmica serão resolvidos de forma mais eficaz quando forem tratados adotando-se uma Perspectiva Estratégica. Um problema sistêmico certamente envolverá conhecer muito profundamente suas várias causas – dificilmente esse tipo de problema terá uma única causa -, e, mais do que isso, envolverá uma análise profunda para se chegar à causa-raiz primária que efetivamente gera o problema. Quando se tenta resolver um problema utilizando a Perspectiva Estratégica, espera-se causar um grande impacto na forma em que o projeto está estruturado ou até nos conceitos sobre os quais ele se baseia. A análise de causa-raiz utilizada pela Toyota é um ótimo instrumento para endereçar problemas sob o ponto de vista estratégico.

A Perspectiva Estratégica não é a única que podemos utilizar para criarmos as condições para o aperfeiçoamento contínuo. O dia-a-dia em um ambiente de trabalho é tomado por inúmeros pequenos problemas que quando se acumulam podem gerar um efeito devastador sobre sua eficácia. É para resolver esse tipo de problema que precisamos adotar instrumentos que atuem equilibrando o processo na medida em que as práticas adotadas não geram o resultado apropriado. Quando intervimos no modelo operacional de trabalho estamos atuando segundo uma Perspectiva Tática. São esses instrumentos que nos ajudarão a responder à pergunta: “Como podemos fazer isso melhor?”.

Mesmo depois de sermos capazes de auto-aperfeiçoar nossos próprios processos e práticas de trabalho, ainda temos mais um desafio a resolver: Como nos auto-aperfeiçoarmos como pessoas ou como profissionais na direção de fazer essa máquina de auto-aperfeiçoamento girar? Essa é a Perspectiva Individual e é nela que está a base de tudo, pois, no final das contas, é o esforço individual das pessoas que será somado na direção de aperfeiçoar toda a organização.

A Perspectiva Estratégica: O Kaizen

“Perguntamos por que cinco vezes”. É o que diz Yuichi Okamoto, ex-presidente do Toyota Technical Center, quando perguntado sobre o segredo do sucesso da Toyota e do seu sistema de produção, mais conhecido como Toyota Production System (Liker, 2005). Um dos elementos mais importantes do sistema de produção utilizado pela Toyota é o seu modelo para melhoria contínua e solução de problemas, também conhecido como *kaizen*. O princípio básico desse modelo reside na consciência de que todo o problema, antes de ser um mero inconveniente, é acima de tudo uma oportunidade de melhoria; e que a solução para tais problemas só serão permanentes e efetivas se identificarmos a sua “causa-raiz”.

[Voltar para o Sprint Backlog](#)

O primeiro desafio desse modelo é deixar de procurar pela “fonte” de um problema e começar a tentar identificar sua “causa-raiz”. Um dos principais instrumentos que a Toyota utiliza para isso, é o que ela chama de “Análise dos 5 porquês”. Ao identificar um problema, pergunte porquê ele ocorre. Após encontrar a primeira resposta, pergunte porquê novamente. Repita esse processo repetidamente por cinco vezes e você chegará a uma razão onde o nível de distanciamento da causa para o efeito será o mais adequado para encontrar a solução definitiva para o problema. Por exemplo, imagine que uma equipe utilizando uma metodologia ágil no desenvolvimento de um produto não esteja conseguindo entregar uma versão funcional com qualidade de uso ao final de uma iteração. Seguindo técnicas do Scrum, a equipe consegue fazer demonstrações do seu produto para o cliente ao final da iteração. Mas a versão desenvolvida na iteração n só consegue ser disponibilizada para uso em meados do meio da iteração n+1.

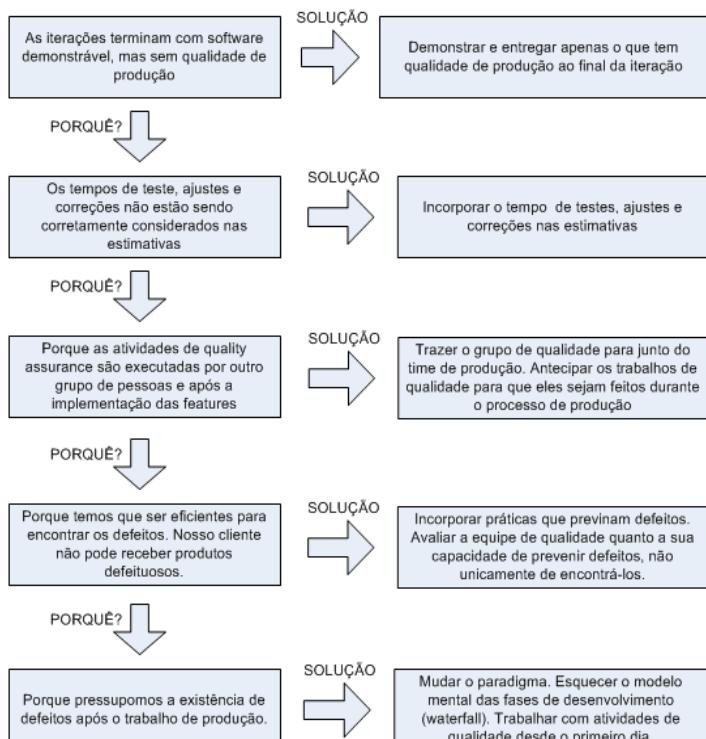


Figura 1: Exemplo de uso da técnica dos 5 porquês da Toyota.

A equipe identifica isso como um problema, pois as atividades que envolvem ajustes e correções executadas sobre as *features* da versão n afetam o rendimento da iteração n+1 e seguem afetando as demais iterações de forma progressiva. Veja na Figura 1 um possível uso da técnica dos cinco porquês para entender melhor o problema e, assim, encontrar soluções definitivas.

Normalmente, quando tentamos resolver um problema, nós nos concentramos em resolver seu primeiro nível (o primeiro porquê). Como nossa mente está muito acostumada a estabelecer relações diretas de causa e efeito, esse foco na causa superficial é bastante natural. Mas, infelizmente, essa abordagem não costuma gerar bons resultados. Quando você se concentra em resolver a causa imediata, entra-se em um processo que eu costumo chamar de “se livrar do problema” ao invés de “resolver o problema”. Normalmente, a solução para o primeiro porquê resolve o problema de forma imediata, mas não evita que ele apareça novamente e, o que é pior, colabora para o efeito progressivo causado pelo problema na medida em que ele se mantém ao longo do tempo. No exemplo da Figura 1, quando você decide simplesmente entregar apenas o que tem qualidade de produção no final de uma iteração e deixar o que ficou de fora para a próxima, você “se livrou” do problema. Afinal, o que não ficou pronto será absorvido na iteração seguinte e tudo aparentemente estará de acordo com os princípios metodológicos. Mas, nesse caso, você realmente resolveu o problema? Sob que ponto de vista? O seu ou do seu cliente?

Conforme já exposto na Parte I desse artigo, a Toyota avalia a resolução de problemas sob a ótica dos elementos de avaliação primária. Isso envolve qualidade, custo, produtividade, segurança e atendimento ao cliente (Liker & Meier, 2007).

Quando apenas a causa superficial de um problema é resolvida, normalmente os efeitos sobre esses elementos não são eqüalitativamente distribuídos. Um elemento acaba sendo afetado negativamente para que outro seja afetado positivamente, em uma espécie de lei de compensação. Quando você atinge a causa-raiz de um problema, é comum que todos os elementos sejam afetados positivamente, variando apenas em que momento cada elemento será influenciado (curto, médio ou longo prazo).

Retomando o exemplo da Figura 1, a resposta ao segundo porquê começa a nos levar às origens do problema. Assim, quando a equipe se depara com a questão de incorporar os tempos de testes e ajustes às estimativas, ela perceberá que isso não será uma tarefa fácil, pois uma equipe diferente faz os testes e é difícil prever a complexidade dos ajustes que precisarão ser feitos logo depois que uma funcionalidade é desenvolvida e que um feedback é dado. Aprofundando a análise, chegaremos ao problema conceitual que efetivamente contribui para as dificuldades que a equipe vem enfrentando. Diferentemente do que é incentivado pelas metodologias ágeis, a equipe continua executando pequenos ciclos de fases encadeadas dentro das iterações, deixando uma fase de testes ser executada após a fase de implementação. O que a leva a perceber que ainda é necessária uma reavaliação conceitual do seu método de trabalho. A intervenção sistêmica ocorrerá, assim, por meio de uma mudança no paradigma utilizado, o que causará um impacto profundo no modelo utilizado.

Chegar ao quinto porquê nos leva a um melhor entendimento do problema e, na maioria das vezes, a uma solução de efeito sistêmico permanente, ao invés de uma solução temporária e pouco eficaz.

No entanto, é importante ter em mente que soluções sistêmicas eventualmente são tão traumáticas que os estudiosos do modelo Toyota costumam dizer que, nesses casos, não há como melhorar sem piorar primeiro. É preciso muita coragem e muito envolvimento do grupo de trabalho para adotá-las. No exemplo utilizado anteriormente, a mudança que precisa ser feita certamente causará muitos transtornos no início, mas depois de passado o perfil de turbulência – aquela piora antes da melhora -, o projeto se beneficiará de uma intervenção definitiva e de um efeito de melhoria sistêmica muito maior.

O modelo Kaizen da Toyota é um modelo essencialmente organizacional. Ele se infiltra em cada setor da organização até chegar ao nível do indivíduo, que terá seus próprios instrumentos de melhoria contínua. Quando falamos especificamente da realidade de nossos projetos de software, nem sempre temos a capacidade de influenciar toda uma organização de forma que ela possa adotar um modelo tão engenhoso como é o caso do Kaizen. Mas podemos utilizar as mesmas técnicas para tentar influenciar nossos projetos de forma mais significante. A grande diferença entre essa Perspectiva Estratégica e a Perspectiva Tática que veremos a seguir, é que a primeira é lenta, requer muito tempo de observação, análise e reflexão. Portanto, é ineficiente se feita no escopo de uma reunião de retrospectiva, por exemplo. Soluções que emergem dessa perspectiva também tendem a ser dificeis e custosas para se implementar. Por isso, é importante saber quando e como usá-la.

Perspectiva Tática: As Retrospectivas

Também podemos influenciar nossos próprios projetos adotando práticas de melhoria contínua que atuem no âmbito tático e operacional. Os métodos ágeis de forma geral nos oferecem alguns instrumentos para lidar com o aperfeiçoamento contínuo no âmbito do dia-a-dia do projeto. O instrumento mais utilizado para isso são as Reuniões de Retrospectiva.

[Voltar para o Sprint Backlog](#)

Projeto Ágeis são semelhantes a campeonatos esportivos. Um campeonato é disputado jogo a jogo. Cada jogo terá uma história diferente e apresentará um resultado diferente. A equipe vencerá se conseguir focar em resultados, se adaptar com a realidade de cada jogo e ao mesmo tempo aperfeiçoar sua forma de jogar ao longo do campeonato. Projetos tradicionais esperam o campeonato terminar para refletir e melhorar. O modelo ágil, por outro lado, trata cada uma de suas iterações como as equipes esportivas tratam seus jogos. A primeira atitude de um técnico esportivo depois de um jogo é reunir a equipe para falar sobre os pontos positivos e negativos do último jogo. No contexto do modelo ágil, tais reuniões são chamadas de Reuniões de Retrospectiva.

Usualmente, essas reuniões ocorrem no início ou ao final de uma iteração e tem como objetivo avaliar os pontos positivos e negativos do processo e das práticas adotadas na iteração anterior, bem como propor ações de melhoria para a iteração seguinte. A retrospectiva é o instrumento que endereça de forma mais direta o princípio da inspeção e adaptação presente no modelo ágil. Em uma reunião de retrospectiva, uma equipe ágil reflete e discute sobre como executar seu trabalho de maneira mais eficiente. A seguir descrevo rapidamente como essas reuniões são estruturadas e quais são seus principais elementos conceituais.

A Estrutura da Reunião

Reuniões de retrospectiva têm mais semelhança com workshops do que com reuniões segundo o formato tradicional que conhecemos (aquele em que todos sentam em uma mesa retangular munidos de uma caneta e um bloco de anotações).

As pessoas não precisam fazer anotações em uma retrospectiva (a não ser que realmente queiram). Toda a coleta e registro de informações será feito por meio de quadros brancos, flipcharts e post-its. Além de facilitar o trabalho colaborativo de coleta e análise dos dados, esse tipo de material fará parte da decoração do ambiente de trabalho, criando um “lembrete visual” do que foi discutido e acordado. Utilizar datasheets e softwares durante a reunião pode não ser uma boa idéia, uma vez que muitas vezes contribuem para “quebrar” o clima de colaboração devido ao trabalho operacional enfadonho.

Outros pontos importantes a se considerar para uma Reunião de Retrospectiva:

- Controle de tempo (timeboxed). Definir os tempos de cada etapa da reunião. Isso a tornará mais objetiva e ajudará na manutenção do foco da reunião.
- Participação de todos os membros da equipe. Além da contribuição de todos nas discussões, é importante dar aos membros da equipe a sensação de que eles fazem parte do processo de melhoria do seu próprio trabalho.
- Ambiente confiável e seguro. Deve-se estabelecer que a reunião é um ambiente de livre troca de idéias sem medo ou insegurança. Sem pressões hierárquicas e sem críticas pessoais.

A Diretiva Primária (“The Prime Directive”)

Uma retrospectiva NÃO é uma reunião de cobrança ou de avaliação individual de performance. É uma reunião de avaliação das práticas e do processo, não das pessoas. O instrumento utilizado para tentar evitar que uma coisa se transforme em outra é, antes de começar a reunião, convocar os participantes a aceitarem a Diretiva Primária (Kerth, 2001), que declara:

“Independentemente do que descobrirmos, devemos entender e acreditar que todos fizeram o melhor que podiam, dado o que era conhecido até então, as habilidades de cada um e as condições do momento”.

Quando a equipe aceita essa prerrogativa, a reunião será focada nas causas dos problemas e não nas pessoas que os causaram. De fato, uma reunião onde se discutem problemas pode facilmente ter seu foco desviado para críticas pessoais e para a busca de culpados. Se a reunião atingir este ponto será totalmente ineficaz. Portanto, é preciso ter muito cuidado na condução de tais reuniões, principalmente com equipes com problemas de relacionamento ou que trabalhem em ambientes muito competitivos. No caso de haver um problema individual de performance este deve ser discutido em outro âmbito, não no âmbito da retrospectiva.

Introduzindo a situação atual

Na etapa inicial de uma retrospectiva, é importante estabelecer a situação atual do projeto e o quê foi alcançado até então. Busca-se com isso, criar o foco necessário para o resto da reunião. Se você quer manter a reunião mais formal, pode utilizar essa etapa para apresentar uma agenda, introduzir convidados ou resumir os esforços alcançados até o momento pelas ações de melhoria da equipe. Essa etapa não deve levar mais do que alguns minutos e abrirá espaço para a próxima etapa: a coleta de informações.

Coleta de Dados: Análise quantitativa

Números têm fundamental importância pois evidenciam tendências de melhora ou de piora para determinados aspectos do projeto. A questão-chave é identificar quais os números realmente são relevantes e que merecem ser analisados no seu projeto. Eventualmente eles servem apenas para demonstrar ou apoiar uma análise teórica. Em outros casos, eles podem revelar se os níveis de qualidade do projeto estão melhorando ou piorando.

Você pode aproveitar a retrospectiva para coletar e divulgar números que evidenciem a qualidade do trabalho desempenhado pela equipe. A idéia é registrar esses números em quadros de acompanhamento e monitorá-los entre uma iteração e outra. Na Tabela 1, são apresentados alguns exemplos de métricas que podem ser monitoradas. Veja mais em (Leffingwell, 2006).

Métrica	It.03	It.04
Nº de pontos aceitos	12	12
Nº de pontos entregues	10	14
Nível de Entrega (%)	83%	116%
Nº de pontos adicionados na iteração	0	2
Nº de pontos removidos da iteração	2	0
% de histórias automatizada por testes	77%	63%
% de histórias documentadas	15%	17%
Total de testes unitários	127	145
Total de testes automatizados	22	32
Cobertura de testes	80%	75%
% de Progresso do Projeto	2%	8%

Tabela 1: Exemplos de análises numéricas para acompanhamento das iterações

Um outro tipo de análise quantitativa pode nos ajudar a saber se estamos conseguindo melhorar um outro aspecto importante do projeto: a questão motivacional. Medir o nível de satisfação da equipe pode ser uma boa maneira de acompanhar os níveis de aceitação psicológica que a equipe tem com relação às suas atividades e ao seu ambiente de trabalho. No livro Agile Retrospectives, Making Good Teams Great (Derby & Larsen, 2006), é descrito um instrumento interessante para avaliar satisfação: um histograma de satisfação (Figura 2). Por exemplo, a lista apresentada a seguir apresenta os níveis de satisfação considerados na análise:

5. Extremamente Satisfeito
4. Muito Satisfeito
3. Satisfeito
2. Eventualmente Satisfeito
1. Insatisfeito

Durante a retrospectiva, cada membro da equipe escreve o número que represente seu nível de satisfação em um cartão, post-it ou pedaço de papel e o coloca em uma sacola, na mesa ou o empilha. Depois um voluntário faz a contagem dos números e preenche um histograma como o da Figura 2.

[Voltar para o Sprint Backlog](#)

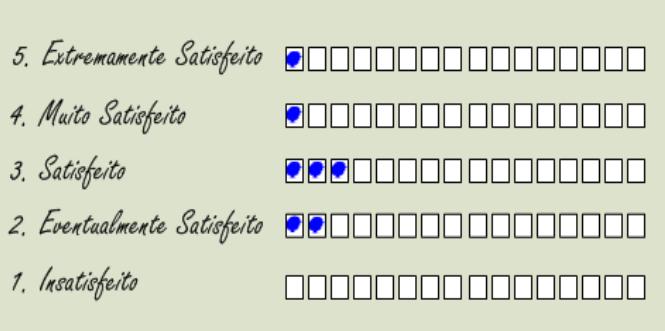


Figura 2: Exemplo de um Histograma de Satisfação

Ao se fazer isso em todas as retrospectivas, obtém-se uma noção do nível de motivação da equipe, podendo-se utilizar essa informação para planejar ações que aumentem a auto-estima do grupo melhorando, dessa maneira, sua coesão e disposição para atingir as metas estabelecidas.

Coleta de Dados: Análise qualitativa

O próximo passo da reunião é a coleta de dados não numéricos: eventos significativos da iteração, ações que funcionaram bem ou aquilo que precisa ser melhorado. Há algumas técnicas que podem ser utilizadas para isso. A primeira é o Timeline. Desenha-se uma linha horizontal em uma folha de flip-chart marcando-se a data de início da iteração no início da linha e a data de fim no final da linha. Distribui-se post-its para a equipe, que então os utiliza para descrever os eventos significativos que aconteceram durante a iteração. Os post-its são colados nas datas em que os eventos ocorreram na linha do tempo de forma que a história da iteração possa ser visualizada graficamente. A função do timeline é principalmente reavivar a memória da equipe para não deixar que um possível evento significativo escape da análise que virá a seguir. Esse trabalho com o timeline também ajudará no processo de nivelamento do conhecimento da equipe sobre o que houve na iteração. A idéia é que a memória coletiva seja utilizada para emergir os eventos que aconteceram durante a iteração, tornando-os visíveis para toda a equipe.

Com o timeline montado será mais fácil levantar o que foi bom e o que foi ruim dentro da iteração. A técnica é fazer com que cada membro da equipe descreva em post-its itens que respondam a algumas perguntas simples que os levem a fazer uma análise qualitativa da iteração de forma rápida e sem formalismos, por exemplo:

- O que foi bom e devemos continuar fazendo na próxima iteração?
- O que poderia ter sido melhor?
- O que devemos tentar evitar para não acontecer novamente na próxima iteração?
- Que ações podemos tomar para melhorar o que não foi bom?

Ao escrever em um post-it e colar na área de análise da reunião (que pode ser um flip-chart ou um quadro branco), o membro da equipe deve descrever aquilo que escreveu em poucas palavras. Nesse momento, é importante não deixar que se comece longas discussões sobre possíveis causas ou solução para os itens apresentados. O momento é mais de elicitação do que propriamente de discussão. É importante também separar o momento em que se fala dos problemas do momento em que se fala das soluções. Misturar as duas coisas pode causar desordem e confusão. O Scrum tem uma boa técnica para resolver esse processo, conforme é descrito a seguir.

Análise qualitativa no Scrum

Equipes que utilizam o Scrum como metodologia de trabalho costumam trabalhar com questões semelhantes, mas utilizando perguntas mais diretas e padronizadas (em inglês): “**What Went Well?**” e “**What can be improved?**” (respectivamente “O que foi bom?” e “O que pode ser melhorado?”).

No Scrum, cada item da lista do que pode ser melhorado é um impedimento. Impedimentos atrapalham ou bloqueiam o trabalho da equipe, por isso precisam ser direcionados para uma solução. Tal solução pode estar no âmbito da própria equipe ou no âmbito da organização. Por isso, há ainda uma terceira pergunta a ser feita para cada impedimento: “**Who is in control of this impediment?**” (“Quem tem o controle sobre o impedimento?”). Há apenas duas respostas possíveis: a equipe ou a organização. O que for considerado como algo que pode ser resolvido no âmbito da equipe é adicionado ao Backlog de Produto. As questões de âmbito organizacional são adicionadas ao Backlog de Impedimentos. O **Scrum Master** acompanhará o Backlog de Impedimentos tentando mediar as soluções junto à organização. A equipe, por sua vez, terá ainda a responsabilidade de explicar a importância dos itens adicionados ao **Backlog** de Produto e de negociar algum tempo com o **Product Owner** para que os impedimentos listados ali possam também entrar no escopo de trabalho das próximas iterações.

A Perspectiva Individual

Um dos aspectos-chave das retrospectivas é o fato de elas canalizarem o esforço de todo um grupo de trabalho para melhorar o trabalho dos indivíduos que a ele pertencem. Retrospectivas freqüentes não trazem apenas resultado para o projeto em si, ou para a organização. Elas trazem benefício direto para cada indivíduo que pertence àquele grupo. Eventualmente, tais benefícios retornam para o indivíduo na forma de melhora na qualidade do trabalho, em melhora da sua capacidade técnica ou até mesmo na melhora da qualidade de vida de cada pessoa.

Tendo consciência disso, será muito mais fácil para as pessoas adotarem a “Perspectiva Individual” da melhoria contínua. Nesse âmbito ocorre o caminho inverso: o indivíduo ajuda o grupo. Nesse modelo, todas as pessoas da equipe podem utilizar sua capacidade criativa e de resolução de problemas para criar e pensar sobre instrumentos que gerem resultado em termos de melhoria para todas os outros membros da equipe. Nada impede, por exemplo, que um membro da equipe pense e proponha um modelo de teste mais eficiente, ou que ele use uma parte do seu tempo para aprender e divulgar o uso de uma ferramenta mais produtiva. Mesmo que isso não seja de sua responsabilidade, todos devem se preocupar em melhorar o que fazem, em serem úteis não só executando suas tarefas diárias, mas também em criar soluções que gerem benefício direto para o grupo em que trabalham. Cada membro da equipe pode sim ver-se beneficiado pela sinergia do processo de melhoria contínua e, assim, ser estimulado a dar a sua parcela de retribuição. Ou pensando em oportunidades de surpreender seus colegas com criatividade e bom senso, ou simplesmente esforçando-se para propor idéias inovadoras para a melhoria do projeto em que trabalham.

Conclusão

Não há dúvida que ambientes de trabalho que consigam unir as Perspectivas Estratégica, Tática e Individual terão um efeito devastador sobre seus resultados. É claro que isso não é uma tarefa fácil. Mas o simples de fato de sabermos que há tantos instrumentos que podemos utilizar para criar processos de melhoria contínua em nossos projetos, já é um grande ponto a nosso favor. Afinal, não há nada mais reconfortante do que descobrirmos que somos melhores hoje do que fomos ontem.

Referências

Derby, E., & Larsen, D. (2006). *Agile Retrospectives: Making Good Teams Great*. The Pragmatic Programmers, LLC.

Kerth, N. L. (2001). *Project Retrospectives: A Handbook for Team Review*. New York: DORSET HOUSE PUBLISHING CO., INC.

Leffingwell, D. (2006). *Iteration and Release Retrospectives: The Natural Rhythm for Agile Measurement*. Agile Journal .

Liker, J. K. (2005). *O Modelo Toyota: 14 Princípios de Gestão do Maior Fabricante do Mundo*. Porto Alegre: Bookman.

Liker, J. K., Meier, D. (2007). *O Modelo Toyota: Manual de Aplicação*. Porto Alegre: Bookman.

Sprint-It & InfoQ. (2006). *InfoQ: Scrum Checklists*. Retrieved Mar 18, 2008, from InfoQ: <http://www.infoq.com/minibooks/scrum-checklists>

Vale, A. (2008). *Aperfeiçoamento de Projetos Ágeis - Parte 1: Visão Sistêmica*. Revista Visão Ágil (Ed. 3 Ano II).



Revista Visão Ágil

Artigos Cases

Comunidade

Biblioteca Pública

Eventos

Tudo sobre Agile

www.visaoagil.com

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

Nove Pecados Mortais no Planejamento de Projetos



Tradutor: Victor Hugo Germano

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

Autor: Steve McConnell é CEO e Engenheiro Chefe na Construx Software. Autor dos livros *Code Complete* (1993, 2004) e *Rapid Development* (1996), ambos premiados pela revista: *Software Development*. Em 1998, publicou *Software Project Survival Guide*, *Professional Software Development* (2004) e em 2006: *Software Estimation: Demystifying the Black Art*.

Introdução

Em um momento em que algumas empresas de software alcançaram resultados próximos à perfeição em entregas, outras continuam a sofrer com resultados medíocres. Pesquisas normalmente indicam que um planejamento de projetos pobre é uma das principais fontes de problemas. Como você pode reconhecer um projeto de software mal planejado? Aqui você encontrará alguns dos pecados mais mortais que pude encontrar em planejamento de projetos.

1. Sem Planejamento algum

De longe o problema mais comum em planejamento é não ter planejamento algum, sendo este pecado facilmente evitável. Uma pessoa não precisa ser um especialista na área para criar um planejamento eficiente. Tenho visto inúmeros projetos planejados por amadores que tem se saído bem simplesmente pelo fato das pessoas no comando terem considerado cuidadosamente as necessidades específicas do projeto. Forçado a escolher entre um especialista em planejamento de projetos que não pensa cuidadosamente através de seu plano ou um amador na área que tenha exaustivamente avaliado suas necessidades, eu apostaria no amador todas as vezes.

2. Falha ao contabilizar todas as atividades do Projeto

Se o **Pecado Mortal #1** é não ter planejamento algum, o **Pecado Mortal #2** é não planejar o suficiente. Alguns planos de projetos são criados assumindo que ninguém no time de desenvolvimento ficará doente, terá treinamentos, férias ou sairá da empresa. As atividades principais são geralmente subestimadas em grande nível. Planejamentos criados assumindo condições irreais como estas levarão projetos ao desastre.

Existem inúmeras variações deste tema. Alguns gestores negligenciam ao contabilizar atividades secundárias como o esforço necessário para configurar sistemas, converter dados de versões anteriores, realizar cortes para novos sistemas, realizar testes de compatibilidade e outros tipos de trabalhos chatos que necessitam de mais tempo em projetos do que gostaríamos de admitir.

Alguns projetos que terminam dentro do planejamento reduzindo o ciclo de testes originalmente planejado; a razão para tal é que provavelmente não existirão muitos defeitos para serem detectados ou corrigidos. (Determinar o porque - se for realmente o caso - não se costuma planejar um ciclo menos de testes no primeiro momento é deixado como um exercício ao leitor.)

3. Falha ao Planejar por riscos

Em *Design Paradigms*², Henry Petroski argumenta que, na arquitetura de pontes, as falhas mais espetaculares foram normalmente precedidas de períodos de sucesso que levaram ao prazer na criação de novos designs. Arquitetos de pontes mal sucedidas estavam tão iludidos em copiar os atributos de pontes bem sucedidas que não prestavam atenção suficiente em cada uma das potencialidades de falha para as novas pontes.

Para projetos de software, esquivar-se constantemente de falhas é tão importante quanto simular o sucesso. Em muitos contextos, a palavra "risco" não é mencionada a menos que o projeto já esteja em grandes problemas. Em software, o gestor do projeto que não utiliza a palavra "risco" todos os dias e não incorpora o gerenciamento de riscos em seu plano provavelmente não está realizando seu trabalho. Como Tom Gilb diz: "Se você constantemente não atacar os riscos de seu projeto, eles irão constantemente atacar você"³

4. Utilizar o mesmo plano para todos os projetos

Algumas empresas crescem envoltas em uma forma particular de conduzir os projetos de software, que é conhecida como "o jeito que fazemos as coisas por aqui". Quando esta abordagem é utilizada, a empresa tende a ir bem enquanto o projeto se parecer com os antigos projetos.

[Voltar para o Sprint Backlog](#)

5. Aplicar Indiscriminadamente Planejamentos Pré-fabricados

Um primo próximo do Pecado Mortal #3é reutilizar um plano genérico criado por outra pessoa sem aplicar seu próprio julgamento ou sem considerar suas necessidades únicas de projeto. "O plano de alguém" normalmente chega na forma de um livro ou uma metodologia que um gestor aplica em teoria. Exemplos mais atuais incluem *Rational Unified Process*,⁴ *Extreme Programming*, em algum ponto (apesar de minhas melhores intenções de fazer o contrário) meu próprio *Software Project Survival Guide* e o *CxOne* da minha empresa. Estes planos pré-fabricados podem ajudar a evitar os Pecados Mortais #1e #2, mas não são um substituto para a tentativa de otimizar o planejamento de um projeto para sua principais necessidades.

Nenhum especialista de fora pode compreender as necessidades específicas de um projeto tão bem quanto as pessoas diretamente envolvidas nele. Gestores deveriam sempre deixar o planejamento "dos especialistas" para suas circunstâncias específicas. Felizmente, tenho visto que gestores que tem a real consciência dos problemas envolvidos em planejamento para lerem livros de engenharia de software também possuem suficiente bom senso para serem seletivos sobre que partes de um plano pré-fabricado poderiam funcionar.

6. Permitir que um Planejamento fuja à Realidade do Projeto

Uma abordagem comum para o planejamento é criar um plano no início do projeto e então colocá-lo no armário e deixá-lo acumular poeira apenas para lembrança. À medida que as condições do projeto mudam, o plano torna-se invariavelmente irrelevante, e, ao chegar à metade, o projeto correrá livre de formas, sem nenhuma relação real entre o planejamento inicial e realidade do projeto.

Este Pecado Mortal é exacerbado pelo **Pecado Mortal #5-** gestores que levantam a bandeira de metodologias pré-fabricadas são muitas vezes relutantes à mudança de seu modelo mental quando isso não funciona. Eles pensam que o problema está na aplicação do planejamento quando, de fato, o problema é o próprio planejamento. Bons planejamentos de projeto devem evoluir através do projeto.

7. Planejar em Muitos Detalhes Muito Cedo

Alguns gestores bem intencionados tentam mapear todo o valor das atividades de um projeto muito cedo. Mas um projeto de software consiste em um grupo de decisões constantemente desconhecido, e cada fase de um projeto cria dependências para as futuras decisões. Uma vez que ninguém possui bola de cristal, tentar planejar atividades distantes em muitos detalhes é um exercício de burocracia tão ruim quanto não ter nenhum planejamento.

Quanto mais trabalho houver em criar prematuramente planos detalhados, mais facilmente o planejamento se tornará arquivável (Pecado Mortal #6). Ninguém gosta de desperdiçar trabalhos anteriores, e gerentes algumas vezes tentam forçar a realidade do projeto encaixar em seu primeiro planejamento, ao invés de laboriosamente revisar seu plano prematuro.

Eu imagino um bom planejamento de projetos como dirigir à noite com as lanternas ligadas. Podem existir mapas que dirão como ir da *Cidade A* a *Cidade B*, mas a distância que posso ver em detalhes através de meus faróis é limitada. Em projetos de médio ou grande porte, planejamento deve ser mapeado fim-a-fim cedo no projeto. Detalhamento e micro planejamento deve geralmente ser conduzido apenas algumas semanas por vez, e deve ser guiado na forma "just in time".

8. Planejando para alcançar no futuro

Para projetos que trabalham em atraso, um erro comum é planejar para recuperar o tempo perdido no futuro. A racionalização típica é que "A equipe estava escalando a curva de aprendizado no início do projeto. Nós aprendemos muitas lições na maneira mais difícil. Mas agora nós entendemos o que estamos fazendo, e seremos capazes de finalizar o projeto rapidamente.". **Resposta Errada!**Uma pesquisa realizada em 1991 em mais de 300 projetos encontrou que projetos dificilmente recuperaram o tempo perdido - eles tendem a ficar ainda mais atrasados. A fluxo desta racionalização é que equipes de software fazem suas principais decisões no início do projeto - o momento em que novas tecnologias, novas áreas de negócio e novas metodologias são menos compreendidas.

À medida que a equipe progride para as fases mais adiantadas do projeto, a velocidade não aumenta; ela irá diminuir à medida que encontrar as consequências dos erros cometidos mais cedo e investirá tempo tentando corrigir tais erros.

9. Não aprender com os Pecados de Planejamentos Antigos

O Mais Mortal de todos os pecados é não aprender com os pecados anteriores. Projetos de Software podem durar muito tempo, e a memória das pessoas pode ser preenchida com egos e a própria passagem do tempo. No final de um longo projeto, pode ser difícil recordar todas as decisões anteriores que afetaram sua conclusão. Uma forma fácil de conter esta tendência e prevenir futuros pecados mortais é conduzir uma retrospectiva *post mortem*. Tal processo estruturado pode não apagar os pecados passados de um projeto, mas pode certamente ajudar a prevenir os pecados dos futuros.

Referências

1. Sanjiv Ahuja, "Laying the Groundwork for Success" (Interview), *IEEE Software*, Nov/Dec 1999, pp. 72-75.
2. Henry Petroski, *Design Paradigms*, University Press, 1994.
3. Tom Gilb, *Principles of Software Engineering Management*. Wokingham, England: Addison-Wesley, 1988.
4. Phillippe Kruchten, *The Rational Unified Process: An Introduction*, 2d Ed., , Mass.: Addison Wesley, 2000.
5. Kent Beck, *Extreme Programming: Embrace Change*, , Mass.: Addison Wesley, 2000.
6. Steve McConnell, *Software Project Survival Guide*, Redmond, WA.: Microsoft Press, 1997.
7. Michiel van Genuchten. "Why is Software Late? An Empirical Study of Reasons for Delay in Software Development." *IEEE Transactions on Software Engineering*, vol. 17, no. 6 (June 1991), pp. 582-590.
8. Bonnie Collier, Tom DeMarco, and Peter Fearey, "A Defined Process For Project Postmortem Review," *IEEE Software*, July 1996, pp. 65-72.

Links

1. Texto original: <http://stevemcconnell.com/ieeesoftware/eic19.htm>
2. Autor: <http://blogs.construx.com/blogs/stevemcc/default.aspx>
3. Blog do Tradutor: <http://malditacomedia.blogspot.com>

Visão Ágil pelo Mundo

Acompanhe a visitação ao nosso site(www.visaoagil.com) durante a última edição



8,086 visits came from 32 countries/territories

Detail Level: [City](#) | [Country/Territory](#) | [Sub Continent Region](#) | [Continent](#) Segment: [Choose...](#)

[Site Usage](#) [Goal Conversion](#)

Views:

Visits 8,086 % of Site Total:	Pages/Visit 1.24 Site Avg: 1.24	Avg. Time on Site 00:00:48 Site Avg: 00:00:48	% New Visits 76.28% Site Avg: 76.27%	Bounce Rate 87.97% Site Avg: 87.97%
--	--	--	---	--

Concluído

www.google.com

ScrumMaster por ele mesmo

Avalie o comportamento ideal de um ScrumMaster e suas habilidades vitais

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: axmagnog@gmail.com / <http://amagno.blogspot.com>



Introdução

Através deste estou iniciando uma série de artigos que irá avaliar detalhadamente cada um dos três principais papéis do Scrum. O que motivou esta série foi a grande quantidade de questionamentos que recebo em meus treinamentos e consultorias em volta deste assunto: quem deve ser o Product Owner? ScrumMaster é o gerente de projeto? Ele deve ser técnico?

Um membro do time pode exercer mais de um papel? Por quem são gerenciados os problemas burocráticos do projeto (custo, penalidades, etc.)? Estas são apenas algumas das mais freqüentes perguntas que tenho recebido e que acredito que habite a cabeça de muitos que estão iniciando (ou não) o trabalho com Scrum.

Na série de artigos “*...por ele mesmo*” não vou citar perguntas e respostas, mas sim situações reais que aconteceram em variados projetos em meus clientes. Por exemplo, no artigo “ScrumMaster por ele mesmo” estarei narrando um conjunto de histórias vivenciadas junto a ScrumMasters de alguns clientes. Obviamente não conseguirei abordar em detalhes tudo que envolve o dia-a-dia de um ScrumMaster, nem mesmo é este o objetivo do artigo, mas pretendo citar situações reais de lições aprendidas por clientes em relação a este importante papel em projetos Scrum.

Por motivos óbvios estarei utilizando nomes fictícios tanto para as empresas e verticais quanto para os ScrumMasters. Espero que o conjunto de artigos agrade a comunidade, e esclareça, dentro do alcance de um artigo, as principais dúvidas em volta dos papéis no Scrum.

Quem é o ScrumMaster?

O papel do ScrumMaster, diferentemente dos gerentes de projeto na maioria das práticas e metodologias, está distante do tradicional “comando e controle”. Em Scrum, um ScrumMaster trabalha com e, principalmente, para o time. É esperado que um ScrumMaster possua as seguintes responsabilidades:

- Permitir que o time seja auto-gerenciável:** esta talvez seja uma das mais desafiadoras responsabilidades de um ScrumMaster, pois no mundo de projetos que temos hoje a própria equipe espera que haja dentro do time alguém com o perfil comando-controle para dizer o que cada um tem que fazer e, na seqüência, controlar o desenvolvimento do que foi atribuído a cada membro do time.

Em Scrum espera-se que os times sejam auto-gerenciáveis, o que significa que cada membro do time será responsável por decidir o que vai fazer, como e em quanto tempo. Sendo assim a necessidade de termos nos times alguém cobrando status de atividades passa a ser nula, visto que cada membro estará comprometido com aquilo que planeja entregar. O grande problema aqui é que, devido a anos e mais anos de comando-controle, os próprios membros do time – que reclamam tanto deste modelo – se sentem inseguros ao se verem com uma carga de comprometimento que não estão acostumados. Ao sentir esta insegurança o mais comum é que estes membros tentem “empurrar” de volta o comando-controle para o líder do projeto (em nosso caso, o ScrumMaster) e é aqui que está a grande dificuldade pois, neste momento, o ScrumMaster não deve aceitar o comando-controle que o está sendo oferecido, mas sim orientar cada membro do time a manter-se de forma auto-gerenciada. Um bom ScrumMaster não aceita o controle, por mais tentador que isto possa parecer – principalmente para gerentes acostumados com a forma tradicional de gerenciamento de projetos.

- Garantir que os caminhos para a comunicação do time estejam abertos permanentemente:** sem uma comunicação eficiente um time Scrum certamente irá falhar pois ela é uma peça chave em todo o processo de gerenciamento de projetos com Scrum. Por este motivo temos que ter em um time alguém que seja responsável por garantir que a comunicação do time esteja funcionando de forma satisfatória, esse alguém é novamente o ScrumMaster. Neste cenário o ScrumMaster deve garantir que não haja nenhuma barreira de comunicação (seja física, política, hierárquica, etc.) entre os membros do time, entre time e Product Owner (ou especialistas de domínio) e entre Pigs e Chickens.

• Garantir e auxiliar o time a seguir corretamente as práticas do Scrum: o ScrumMaster é o responsável pela boa aplicabilidade das práticas do Scrum no projeto. O time deve enxergá-lo como alguém que conhece bastante de Scrum e que está pronto para corrigir qualquer deslize na utilização de suas práticas e para tirar as dúvidas relativas a Scrum de qualquer pessoa envolvida no projeto.

• Remover qualquer impedimento que o time encontre: durante a execução das Sprints os membros do time estarão envolvidos em mecanismos que farão com que seus problemas apareçam quase que de imediato. Por exemplo, em um reunião diária membros do time serão encorajados a informar se tiveram algum impedimento no dia anterior e, caso positivo, o ScrumMaster é quem deverá se empenhar para remover aquele impedimento, de forma que mantenha o membro do time focado em seu trabalho. O dia-a-dia do ScrumMaster está diretamente ligado à remoção de impedimentos e, por este motivo, ele deve conhecer habilidades que o façam enxergar os impedimentos o quanto antes e removê-los com rapidez.

• Proteger o time de interferências externas para garantir que sua produtividade não seja afetada: o ScrumMaster é um típico líder-servidor e, como tal, uma de suas principais atribuições é manter o time focado na meta da Sprint e na visão do produto. Para isso ele deve garantir que fatores externos não estejam interferindo na produtividade do time, isso inclui o surgimento de impedimentos, que por ele deverão ser removidos, a interferência no trabalho de algum membro do time através de decisões hierárquicas da empresa, que por ele deverão ser negociadas, e o surgimento da multi-tarefa(1) dentro

do time devido à solicitações externas ao projeto.

(1) multi-tarefa é quando alguém está paralelamente comprometido a executar um conjunto de tarefas de projetos, clientes e interesses diferentes.

• Facilitar as reuniões do projeto (Planning Meeting, Daily Meeting, Review e Retrospectives): Entenda por facilitação a habilidade de transformar divergência em convergência, a habilidade de fazer com que um tema seja explorado de diversas formas diferentes com o propósito de fazer com que todos os envolvidos na discussão consigam entender o que está sendo explorado da melhor forma possível.

As reuniões em Scrum, por serem freqüentes, precisam ser rápidas e eficientes, e é nestas reuniões que o papel de facilitador do ScrumMaster deve ser elevado, pois é neste ponto que ele precisará utilizar todas as suas técnicas e conhecimentos de Scrum e de facilitação para fazer com as reuniões sejam bem sucedidas.



Ministrado por



Um bom software é desenvolvido a partir de bons requisitos. Engana-se quem acredita que abordagens ágeis os ignoram.

Adail Retamal
Pioneiro na prática e divulgação da FDD no Brasil, junto com Scrum, Lean e TOC. Diretor da Heptagon TI.



Alexandre Magno, CSP
Primeiro Certified Scrum Practitioner da América Latina. É Diretor da Caelum.

31/05 (sábado) – 09 às 18hs –
Hotel InterCity Ibirapuera – São Paulo/SP

Informações e inscrições : www.heptagon.com.br e amagno.blogspot.com



Histórias de ScrumMaster

O ScrumMaster conhecedor

A **Cosmos Brothers** é uma empresa com mais de 15 anos no mercado brasileiro, eles possuem uma rede de postos de gasolina com unidades instaladas na regiões Sul e Sudeste do Brasil. **Paulo Cosmos** é sócio e diretor de tecnologia da empresa e, quando assumiu este papel, determinou que a empresa iria trabalhar com *softwares caseiros* para a gestão do negócio, ou seja, softwares desenvolvidos por sua própria equipe.

Gabrielle Moraes é a atual gerente de projetos da Cosmos, ela gerencia os principais projetos da empresa, que são desenvolvidos por uma equipe composta de 8 (oito) membros, dentre eles: programadores, analistas de requisitos, *testers* e arquitetos. Gabrielle participou de um treinamento de Scrum há alguns meses e, recentemente, começou a utilizá-lo no gerenciamento de projetos. Durante a reunião de planejamento do último Sprint, Gabrielle se sentiu insegura para exercer este papel, pois ao ser questionada por Frank sobre qual seria a **framework Java** ideal para a integração do produto deles com o de terceiros, ela não soube responder. Entrave similar ocorreu durante a Sprint, veja abaixo um trecho de uma reunião diária:

“Bom, ontem eu efetivei a criação dos métodos buscarClientesPorUF e buscarClientesPorRegiao. Iniciei a publicação destes métodos em nosso WebService principal, mas passei a ter problemas porque a versão da JRE instalada no servidor não contempla o recurso de parsing que precisei utilizar nos métodos...” - disse Cristiano, desenvolvedor.

“...eu já havia mencionado isto na Sprint passada...” - disse Frank, também desenvolvedor do time, *“tive um problema similar, conforme mencionei em nossa última Retrospectiva.”*

“Então, eu até pensei em fazer um upgrade da JRE, mas tive receio desta ação afetar o trabalho dos outros. O que você acha Gabrielle?” - complementou Cristiano.

“Bom gente...é...como vocês sabem eu não sou especialista em Java, mas andei dando uma pesquisada sobre este assunto desde a nossa última Retrospectiva. Me desculpem, mas eu estou fazendo o melhor que posso para tentar encontrar a melhor solução para o caso.” - disse

Gabrielle, a ScrumMaster do time.

Frank expôs seu ponto de vista - *“...ok, mas acho que o quanto antes tivermos isto melhor pois este é um impedimento que realmente nos está atrapalhando. Gabrielle, outra pergunta, devemos discutir estes assuntos técnicos aqui na reunião diária ou deixamos para citar estes problemas na retrospectiva?”*

“Bom, acho que não tem problema em falar aqui não...inclusive se alguém tiver alguma sugestão para me dar quanto à resolução do impedimento da JRE, eu ficaria grata.” - disse **Gabrielle**.

“...devemos discutir estes assuntos técnicos aqui na reunião diária ou deixamos para citar estes problemas na retrospectiva?”

Paulo Cosmos, Sócio-Diretor da empresa, estava participando da reunião e deu sua contribuição, “*Gabrielle*” - **disse ele**, “*eu acho que você deveria se reunir com alguns desenvolvedores mais experientes para que lhe ajudem neste caso. Se necessário contratar algum serviço de suporte técnico ou consultoria.*”

“Ótimo, é uma excelente idéia!...bom, Cecília, agora é sua vez de responder às 3 perguntas...mas pessoal, nossos 15 minutos já passaram, então solicito a todos que respondam às perguntas o mais rápido possível” **disse Gabrielle**.

Ao ser procurado por Gabrielle, ela me narrou o fato ocorrido naquela reunião diária e falou sobre sua apreensão por não conhecer Java. Falou também que como o papel do ScrumMaster ficava bem mais próximo do time do que um tradicional Gerente de Projetos, ela via agora claramente que só conseguiria exercer bem este papel se conhecesse profundamente assuntos técnicos.

“Muito bom *Gabrielle*” eu disse, “*conhecimento nunca é demais, e conhecer mais sobre a plataforma de desenvolvimento que seu time está utilizando será muito bom para você e para eles.*”.

“Pois é Alexandre, quando ocorreu este fato eu lembrei de certa vez quando você me disse que um ScrumMaster deve ser um conhecedor!” - complementou Gabrielle.

“É verdade *Gabrielle, eu disse isso. Mas veja, quando eu digo que um ScrumMaster deve ser conhecedor isto não significa necessariamente ser um grande conhecedor da plataforma ou linguagem de desenvolvimento que seu time utiliza.*” **eu disse**.

Gabrielle espantou-se com minha colocação e perguntou “*Não?? Então conhecedor em que?*”

Eu respondi com outra pergunta “*Gabrielle, me responda: quais são as principais atribuições de um ScrumMaster?*”.

“*Remover impedimentos, facilitar as reuniões, garantir o uso de Scrum...entre outras, certo?*” **disse ela**.

“Correto...então vamos lá, como você pode ser uma grande conhecedora na remoção de impedimentos?” **enviei mais uma pergunta como resposta**.

“*Bom, depende...mas no caso que estamos falando, conhecer Java me ajudaria a remover o impedimento mais rapidamente*” **rebateu Gabrielle**.

“Hmmm...vamos a uma metáfora *Gabrielle, imagine que você está viajando de carro com sua família, seu marido está ao volante, é noite e chove bastante... no caminho vocês encontram uma árvore caída no meio da estrada que impede a passagem de vocês. O que seria melhor neste momento: o seu marido ter lido um livro for Dummies de como remover árvores de uma estrada com*

Artigos
Cases
Comunidade
Eventos

Biblioteca Pública

Tudo sobre Agile



as próprias mãos ou você ter garantido levar consigo todos os números de emergência, tal como o da empresa responsável pelo socorro na estrada? Qual seria o mecanismo mais seguro para fazer com que sua família seguisse em frente?" - **disse eu.**

Gabrielle respondeu pensativa "É...neste caso não necessariamente saber remover uma árvore do caminho com as próprias mãos seria a melhor solução para remover o impedimento".

"Exatamente Gabrielle..." - **eu disse**, "ser conhecedor na remoção de impedimentos não significa saber fazer tudo, mas sim saber direcionar o melhor caminho para a resolução do problema. Seus filhos estavam no carro, com frio e fome, enquanto isto seu marido lutava com a árvore teimando em tirá-la do caminho com as próprias mãos, você pegou o celular e ligou para o socorro que em 20 minutos estava lá resolvendo o problema. Em quem você acha que seus filhos mais confiarão na próxima vez que estiverem em apuros? Em você ou no seu marido?".

Gabrielle entendeu a mensagem, e percebeu que ao tentar resolver o problema da JRE do servidor com as próprias mãos ela não só estava atrasando a resolução de um impedimento – e consequentemente atrapalhando o time – como também se mostrando como uma péssima removedora de impedimentos, o que, talvez, num futuro faria com que algum membro do time omitisse a existência de algum impedimento só para não ter que vê-la novamente "lutando com a árvore".

“...não necessariamente saber remover uma árvore do caminho com as próprias mãos seria a melhor solução para remover o impedimento”

Eu disse "Gabrielle, você deve ter o conhecimento para escolher o melhor caminho na hora de remover um impedimento, isto sim é ser um ScrumMaster conhecedor! No entanto não é só isso. Voltamos ao assunto da reunião diária que você citou, nela você não se mostrou muito boa na facilitação da reunião".

"Como assim?", **espantou-se Gabrielle.**

"Ficou claro para mim que você não se colocou na postura de um bom facilitador" - **eu disse** "Por exemplo, um assunto técnico citado pelo Cristiano tomou conta da reunião, além disso, ao se ver apressada você sugeriu que todos respondessem suas perguntas o mais rápido possível – o que diminui a qualidade nas resposta – e nem sequer permitiu que o Cristiano finalizasse seu raciocínio!".

“...você deve ter o conhecimento para escolher o melhor caminho na hora de remover um impedimento, isto sim é ser um ScrumMaster conhecedor!”

"É, agora percebo que o meu desespero em providenciar uma solução técnica para algo que já atrapalhava o time desde a Sprint anterior me fez perder o rumo da reunião" **disse Gabrielle.**

"Não só isso Gabrielle. Por não ter facilitado bem a reunião você acabou comprometendo o bom uso do Scrum e, como você mesmo citou, garantir o uso do Scrum é uma das principais atribuições do ScrumMaster. Além de permitir que um assunto técnico invadisse a reunião, você permitiu que um chicken interferisse no andamento da mesma" - **eu disse.**

“Puxa, estou me sentido péssima...acho que realmente não sou uma boa ScrumMaster” - lamentou Gabrielle.

“Não é isso, você apenas estava investindo sua atenção para o conhecimento errado (pelo menos para o momento). Para ser uma boa ScrumMaster você deve primeiramente ser conhecedora da arte de remover impedimentos, ser conhecedora em técnicas de facilitação e comunicação, ser conhecedora da cultura da empresa e, tão importante quanto, ser conhecedora de Scrum para saber exatamente o que fazer em cada prática e para enxergar as armadilhas o quanto antes. Ser conhecedora na plataforma de desenvolvimento de seu time será um ótimo plus, mas sem dúvida nenhuma os pontos citados inicialmente são os mais importantes para um ScrumMaster” - conclui.

O ScrumMaster facilitador

Élder era ScrumMaster da Essential Systems, uma empresa que fabricava um ERP voltado para a vertical de móveis modulados. Ele havia sido meu aluno em uma turma de Scrum e depois havia me chamado para realizar um **coaching** de 60 dias na Essential, e agora, 12 meses depois, entrou em contato para me convidar a participar de uma reunião de planejamento do mais importante projeto que ele estava envolvido atualmente. Aceitei o convite de pronto e, como um **chicken**, fiquei quieto em um canto apenas acompanhando o desenrolar da reunião.

Élder tinha sido um excelente aluno e agora estava claro para mim que ele realmente havia se tornado um grande ScrumMaster. Minha primeira surpresa foi ao constatar que a equipe era muito focada durante as reuniões, a tal “bolinha” para pedir a vez de falar já havia sido apresentada por Élder pelo jeito há muito tempo, pois todos respeitavam a vez de cada um falar e faziam isso de uma forma extremamente natural.

Num momento seguinte, ainda na reunião, Élder interrompeu a apresentação do **Product Owner**, que estava sendo feita em volta de uma das Features mais complexas do **Product Backlog**. Ele informou ao time que aquela interrupção estava sendo feita porque ele acreditava que o foco da conversa estava tendo desvios e que o assunto “**Impressão de Boleto**”, mesmo sendo deveras importante, não estava diretamente ligado àquela Feature. Élder então sugeriu que este tópico fosse colocado em uma área denominada **Rat Hole** que ele havia criado no quadro-branco, esta área iria conter assuntos importantes a serem discutidos, mas que não estavam diretamente ligados àquela reunião de planejamento. Ao fazer isto ele trouxe a conversa novamente para o foco, mas sem causar o mal que seria gerado caso ele sugerisse abandonar aquele assunto por ser “menos importante”. Uma boa técnica de **facilitação**.

Na segunda parte da reunião de planejamento o time solicitou a presença de algum especialista de domínio que pudesse explicar-lhes em mais detalhes como deveria ser a página na qual os usuários conseguiriam visualizar toda a movimentação financeira de um ano para um cliente especificado, pois muito já havia sido discutido com muitas “idas-e-vindas” na conversa.

Carla, a especialista de negócio, falou durante 20 minutos sobre aquela funcionalidade e pareceu colocar mais interrogações na cabeça do time, que discordou em grande parte do que Carla havia explanado. Élder pediu a palavra e solicitou que Carla e um ou dois membros do time se aproximassem de um grande quadro branco que ficava ao fundo da sala. Reunindo todos lá ele pediu para que tentassem em conjunto desenhar a tal página no quadro branco, de forma com que todos os outros envolvidos na reunião pudessem participar. Cada um desenhava um pouco, apagava um pouco e em poucos minutos conseguiram chegar a um consenso, tendo agora em mãos (ou melhor no quadro) uma tela que havia sido desenhada por todos. Um conjunto de fotos, tiradas através de uma máquina digital, documentou o trabalho. Transformar divergência em convergência, esse é o objetivo da facilitação.

No fim da reunião Élder me chamou para tomarmos um *capuccino* na área de convivência da empresa e lá chegando ele foi direto ao ponto “*Alexandre, que tal, o que achou?*”.

“*Élder, foi muito bom!*” - eu disse, “*Você utilizou simples – mas poderosas - técnicas de facilitação e, o mais importante, demonstrou entender perfeitamente qual é o papel do ScrumMaster dentro desta reunião. Além disso o time se mostrou educado e focado, o que com certeza deve ser mérito de sua postura nas reuniões ao longo desses últimos meses. Eles lhe enxergam como um facilitador, como alguém que vai ajudá-los e, por isso, permitem e enxergam sempre com bons olhos a sua entrada no meio de alguma discussão.*”

“*É verdade Alexandre*” - disse ele, “*eu percebi que com o tempo e com os resultados que eu ajudava a gerar durante as reuniões, eles passaram a enxergar minha entrada nas conversas como uma saída para a discussão. Certas vezes percebo inclusive que eles param e ficam olhando para mim, como dizendo: por favor, nos salve desse debate.*” Élder continuou, “*Mas e agora, como aprender mais sobre facilitação? Já li todos os livros sobre este tema e, mesmo sabendo que a boa facilitação parte de ações simples, percebo que preciso aprender mais.*”.

**Transformar divergência em convergência,
esse é o objetivo da facilitação.**

“*Veja, facilitação, na minha opinião, é uma expansão da habilidade de comunicar. Você só consegue ser um bom facilitador se for um bom comunicador. A facilitação é a arte de tornar mais eficiente a comunicação entre grupos maiores, e quem foi um dos maiores comunicadores de toda a nossa história se não Sócrates? Grande parte das técnicas de facilitação que utilizo hoje em reuniões e retrospectivas foram aprendidas através das lições por ele nos ensinada. Facilitação é uma daquelas habilidades do ScrumMaster que mais está ligada ao lado humano do que ao lado técnico da pessoa.*”, conclui.

O ScrumMaster Líder

João Pedro era Gerente de Projetos de uma das principais empresas de aviação do país quando nos conhecemos a 2 anos atrás.

Naquela época eu havia sido contratado como consultor para ajudar a agilizar os processos da empresa, e um dos passos a serem dados envolvia a adoção de Scrum no gerenciamento de projetos. Durante o treinamento que ministrei para todos os gerentes de projetos de lá, João havia se mostrado bastante resistente à idéia do ScrumMaster não possuir poder sobre o time. Lembro dele ter me questionado bastante sobre como ser responsável por algo que não se pode comandar e controlar. Foi bom encontrá-lo novamente em uma conferência sobre gestão de projetos e poder convidá-lo para um almoço.

“Então João, pelo que conversei com o Borges (diretor da empresa) recentemente, parece que Scrum realmente se institucionalizou na empresa. O que você pode me falar sobre isso?” - eu disse.

“Como você sabe muito bem tivemos um trabalho muito duro no início. Mas a boa execução daquele desafiador projeto piloto que você esteve envolvido conseguiu motivar o restante da empresa, tivemos quer criar um Backlog de Projetos – uma espécie de portfólio – para conseguir definir a prioridade de cada e conseguir atender a todos.”, - disse João.

Resolvi expandir a conversa, “Muito bom, me fale um pouco sobre as lições aprendidas, sobre o que aprendeu com o projeto piloto e que conseguiu melhorar nos projetos seguintes?”

“Alexandre, o ponto principal que posso citar foi a questão da postura, percebi a grande diferença entre ser líder e ser gerente. Antigamente, como um gerente de projetos, eu me portava de uma forma que fazia com que os membros do time me enxergassem do lado de fora do círculo. Percebi que eles só passaram a me enxergar como alguém que estava junto com eles dentro do círculo a partir do momento em que mudei minha postura nas pequenas coisas do dia-a-dia.” - disse João.

“Você pode me citar um exemplo de algum fato que tenha deixado isso claro pra você?” - eu disse.

“Olha, certa vez quando estávamos realizando uma reunião de planejamento, eu estava bastante preocupado para que a reunião fosse executada de forma otimizada, então a conduzi de forma a fazer com que o time estimasse e planejasse tudo o mais rápido possível. A consequência disso foi que durante a Sprint o planejamento se mostrou desastroso, e eu tive que sugerir o cancelamento da Sprint. Marquei uma reunião com todos – incluindo Product Owner – e expus a situação, falei que o planejamento havia sido fraco e que eu me sentia responsável por isso, por justamente ter forçado o time a acelerar o processo. Eu mesmo me surpreendi com essa minha atitude, pois em outros tempo eu, sinceramente, buscava um culpado. Para o Product Owner eu diria que o time era inexperiente e se precipitou, para o time diria que o Product Owner não entendia bem do negócio.

Faria isso principalmente para que o time não enxergasse minha falha, afinal nada pode ser pior do que ver um gerente seu falhando, certo? Engano meu! Veja que impressionante. Com essa atitude o efeito foi inverso ao que eu imaginava. O time se colocou ao meu lado, dividiu a responsabilidade do erro, e a partir daí poucas foram as vezes em que vimos alguém no time criando desculpas para algum problema ou tentando colocar a culpa em alguém. Vi que aquela minha atitude serviu de exemplo para eles, e a partir daquele momento passei a me sentir um líder de verdade.”, - finalizou João.

“Muito interessante esta experiência João. É esse exatamente o papel do líder, deixar visível que você é igual a todos, tem fraquezas, comete falhas, e por aí vai, mas que está ali para ajudá-los a fazer um trabalho melhor. Não adianta falar de time, tem que sempre apresentar a prova. Se fala que não há indivíduos e sim um time, tem que mostrar como é que se faz isso, porque todos já estão cansados de discursos. Um bom líder tem que produzir energia diariamente, tem que fazer com que todos se sintam realizando o trabalho mais importante do mundo, tem que inspirar a audácia.” - conclui.

O ScrumMaster influente

Mário Wernner é gerente de sistemas do **Duff Bank**. Ele me procurou a alguns meses para atuar como consultor em um projeto de alto-risco que estava a ser iniciado utilizando Scrum. **ScrumMaster** deste time. Eu o informei que não achava que esta formação fosse a ideal, citei o fato de eu não conhecer a cultura do banco e de não possuir influência em volta de todos os ambientes que iriam estar envolvidos com o projeto. Sugerir a minha participação como **coach**.

“Mas Alexandre,” disse Mário, “Scrum é algo completamente novo para nós, não vejo ninguém do nosso time que possa exercer este papel. Sem contar que acho completamente arriscado colocar um projeto nas mãos de alguém que ainda não liderou nenhum projeto com Scrum.”

“Mário, justamente por este motivo que estou sugerindo que eu atue como coach, para ajudar o ScrumMaster a seguir pelo caminho correto, ajudá-lo a fazer os desvios necessários e a não cair em nenhuma das armadilhas que surgirão. No entanto, pense no ScrumMaster como alguém que terá que remover os impedimentos do time, esses impedimentos estão espalhados por toda a empresa e seus departamentos, e muitas vezes até fora dela. Agora imagine que um desenvolvedor do time que estarei fazendo parte me fale sobre um impedimento que precisa ser resolvido, é um impedimento burocrático que envolve pessoas de outros departamentos. Quem devo procurar? Como devo fazer a solicitação? Mas tudo bem, com algum pouco de comunicação eu consigo pular estas questões e fazer a solicitação à pessoa certa. Esta pessoa não me conhece, o Duff é enorme eu sou novo aqui...será que eu terei a influência necessária para remover este impedimento da melhor forma possível?”, eu disse.

“... para ser influente não basta ter um alto cargo impresso em seu cartão de visitas, mas sim ter um conhecimento verdadeiro da cultura da empresa ...”

“Entendo sua colocação, mas ninguém nasce sabendo não é verdade? Acho que você conseguiria aprender como as coisas funcionam aqui num espaço de tempo razoável. Eu poderia colocar ao seu lado algumas pessoas influentes que poderiam lhe auxiliar...” - disse Mário.

“Certo, realmente poderia funcionar.” eu disse, “Mas veja, o que você está me propondo é exatamente o mesmo que eu. A diferença é que eu estou lhe propondo guiar um ScrumMaster seu, e você está propondo colocar algumas pessoas que conhecem a cultura da empresa e tenham influência aqui para me guiar. O que será que é mais complexo: Scrum ou a cultura da sua empresa?” perguntei.

“Hmmm...olhando por essa ótica vejo que você tem razão, é mais fácil eu colocar alguém para ajudar um ScrumMaster do meu time com Scrum do que tentar tornar alguém influente aqui dentro.”

“Exato, além do que eu como consultor sairei da sua empresa ao final do projeto, e você voltará a ter o mesmo problema de antes. E lembre-se, para ser influente não basta ter um alto cargo impresso em seu cartão de visitas, mas sim ter um conhecimento verdadeiro da cultura da empresa e ser bem visto dentro dela”, conclui.

Palavras finais

Mike Cohn, autor de importantes livros sobre abordagens e práticas ágeis, citou em seu artigo “Leader of the band: Six Attributes of a Good ScrumMaster” que as principais qualidades esperadas em um ScrumMaster são:

responsabilidade, colaboração, humildade, comprometimento, influência e conhecimento. Note que grande parte destas qualidades está ligada diretamente ao lado humano, e não técnico, de um profissional. Isto vai de encontro com o que foi bastante discutido quando da reunião que originou o manifesto ágil, ou seja, precisamos mais de uma mudança de atitude e comportamento em nossos projetos do que de novos processos e ferramentas, e para esta mudança comportamental – a qual está diretamente ligada com as responsabilidades do **ScrumMaster** – precisamos mais de alguém com habilidades humanas do que técnicas. É óbvio que é muito interessante quando o ScrumMaster possui um conhecimento técnico suficiente para conversar na mesma linguagem com o time, isto é ótimo pois facilita a comunicação e faz com que o ScrumMaster consiga mais rapidamente ajudar o time, mas de longe suas habilidades humanas, necessárias para conduzir o time à mudança comportamental desejada, são mais importantes.

Referências

Schwaber, K. 2004. *“Agile Project Management with SCRUM”*. Microsoft Press

Cohn, M. 2007. "Leader of the Band: Six Attributes of a Good ScrumMaster". Scrum Alliance website.

Magno, A. 2007. “The Daily Meeting Trap”. Scrum Alliance website.

Goddard, P. 2008. “The Hills are Alive with the Sound of Scrum”. Scrum Alliance website.

Chamada de Participação

O Visão Ágil 2008 será o evento realizado pela Revista Visão Ágil em parceria com grupos de usuários, que reunirá toda a comunidade brasileira do segmento corporativo, técnico e acadêmico, para debater, aprender e construir fortes bases de conhecimento sobre a adoção de metodologias ágeis nos diversos níveis de uma empresa, através de palestras e workshops feitas por grandes profissionais do cenário nacional e internacional, nas mais diversas vertentes agile como XP(Extreme Programming), FDD(Feature Driven Development), Scrum, OpenUp, DSDM, MSF, Lean, etc.

Datas: 19 e 20 de Setembro de 2008, **Local:** São Paulo, SP – Brasil

Sugestões de Macro-Temas:

#Agile for Business

- Planejamento
- Estimativas
- Métricas
- Custos
- Contratos
- Gestão de resultados
- Gestão de riscos
- Modelos de Maturidade
- Liderança Ágil
- Estudo de Casos

Core Agile

- Requisitos
- Modelagem
- TDD
- Testes
- Integração contínua
- Gerência de Configuração
- Documentação
- Qualidade
- Refactor
- Ferramentas



Participe da organização do evento

-Como Palestrante:

A Revista Visão Ágil está abrindo chamada para palestrantes de acordo com os macro-temas do evento, portanto, se você deseja palestrar, envie um e-mail contendo a seguinte estrutura:

- Mini-CV
- Título e Macro-Tema
- Descrição
- Tópicos
- Por que sua palestra será interessante para os participantes do evento? (Mínimo 3 respostas)

-Como Grupo de Usuários

Sabedora da importância e da força da comunidade agile nacional, estamos convidando TODOS os grupos de usuários interessados para participarem da comissão organizadora do evento, dessa forma, os grupos terão possibilidade de contribuir para o acontecimento do mesmo, através de:

#Responsabilidades:

- Pessoas para staff,
- Apoio aos palestrantes
- Coordenadores de grade,
- Coordenadores de mesa,
- Divulgação

#Benefícios

- Nome do grupo vinculado como apoiador do evento,
- Espaço para apresentar o grupo de usuários ao público.
- Direito a 4 (quatro) inscrições gratuitas

Importante: Todas as submissões deverão ser enviadas para o e-mail eventos@visaoagil.com até o dia **31 de Junho de 2008**.

QConference 2008 em Londres



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.

No período entre os dias **10 e 14 de Março**, tive o prazer de participar da **QConference**. Conferência de arquitetos de software, realizada em **Londres/UK** pela **InfoQ** e que engloba também a conferência **JAOO**. A primeira coisa interessante que notei foi o local. A conferência foi realizada no **Queen Elizabeth Center** em **Westminster**, ao lado da Westminster Abbey (<http://www.westminster-abbey.org/>), próximo às casas do **Parlamento** onde fica o **Big Ben** e a quatro quarteirões do castelo de **Buckingham** e com vista para a **London Eye**. Local privilegiado em Londres.

Tudo isso foi patrocinado pelos participantes que pagaram valores que variavam de USD\$ 680,00 à USD\$ 2896,00. O mais incrível é que a conferência estava cheia. Com bem mais de **1.500 pessoas**. O que demonstra o interesse do público europeu para com esse tipo de encontro.

No segundo dia, tivemos vários workshops sobre **Ruby** e **Ruby on Rails**, além de um workshop sobre **Testes** e um Workshop sobre **Agile**, ministrado por **David Anderson**, um dos criados do **FDD**.

No terceiro dia começou a conferência propriamente dita, com participação de incríveis 105 palestrantes. Nomes como **Erich Gamma, Rod Johnson, Martin Fowler, Kevlin Henney, Neal Ford**, entre outros, abrilhantaram ainda mais o evento, trazendo as tendências e visões sobre os problemas atuais vividos por nós, mortais arquitetos.

Os assuntos foram separados em **16 Tracks** (<http://qcon.infoq.com/london-2008/tracks/>), de forma que cada track tinha seis apresentações.

Um ponto interessante foi a aplicação de conceitos ágeis na organização da conferência. Por exemplo, na forma do participante expressar se gostou ou não de uma apresentação. Na saída da sala, foram disponibilizados cartões coloridos de papel e, um balde. O participante poderia escolher um cartão verde, vermelho ou amarelo para colocar no balde. Ao final os cartões verdes eram respostas positivas, os amarelos significam que poderia ser melhor e os vermelhos significavam que não foi boa a apresentação. Isso permitia agilidade e garantia melhor aderência ao **sistema de avaliação**.



Figura 01 – Palestra de Martin Fowler



Figura 02 – Debate com grandes nomes sobre arquitetura

Pude assistir a algumas palestras sobre **Groovy, Grails, Ruby, Rails, Henkel, DSL** e várias outras. Infelizmente não era possível estar em todos os lugares ao mesmo tempo. Mas conversando com outros participantes, pude perceber que todas as tendências estão linhadas de acordo com o pensamento daqueles palestrantes.

Eu realizei minha palestra sobre **Domain Driven Design**, apresentando conceitos e tendências nessa área. Minha palestra fez parte da track **Effective Design**, que foi organizada por **Kevlin Henney**(InfoQ) e me deu a oportunidade de conhecer figuras como **Pete Goodliffe, Udi Dahan, Kent Beck** e o próprio **Kevlin Henney**.

Sentado ao lado de **Neal Ford**, pude observar a palestra de **Venkat Subramaniam**, que falava sobre como criar **DSLs com Groovy**, mostrando a força desta poderosa linguagem. Nesta mesma palestra descobri que meu destino é utilizar **TextMate** para casos em que uma **IDE** não é necessária. Como uso Windows vista em meu notebook, procurei uma alternativa e encontrei o **E-Text-Editor** e estou gostando muito.

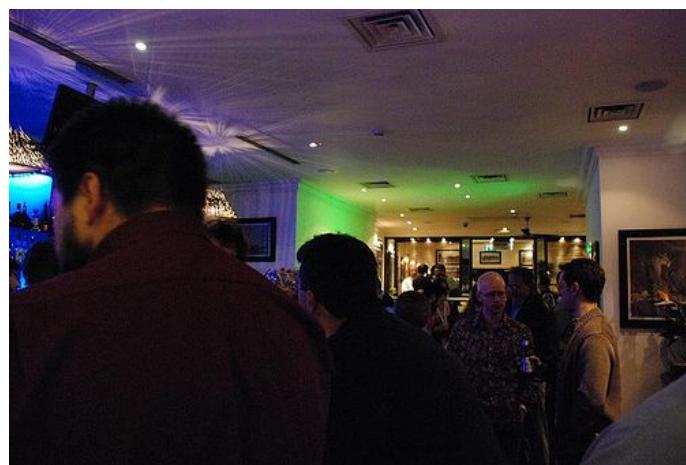


Figura 03 – Noite em um pub para reunir os palestrantes

Outra palestra que me chamou muito a atenção foi :

“**Panel: When is Rails an appropriate choice?**” com **David Chelimsky, James Cox, Aslak Hellesøy, Nic Williams** e **Ola Bini**. Onde eles debateram a qualidade da linguagem **Ruby** e do framework **Rails** para uso em **aplicações corporativas**.

Na quarta-feira a noite, os participantes se reuniram para **Drinks and Snacks** em um **pub** próximo ao local do evento. Extremamente divertido e com a cara de Londres, pudemos dar boas risadas.

Na sexta-feira a noite, foi a vez dos palestrantes e expositores irem à um pub. Pude observar que os arquitetos também tem vida social. Bom, pelo menos alguns têm. Outra noite de gargalhadas e muitas conversa com pessoas de todo lugar no mundo. Dentre os americanos a piada foi o pub servir X-Burgers durante boa parte da noite, coisas que eles estão cansados de comer.

No pub conversei muito com **Floyd Marinescu** sobre a **InfoQ Brasil** e ele garantiu que deseja lança o site no Brasil ainda este ano. O objetivo é traduzir o conteúdo para português e também gerar conteúdo, aproveitando nossos profissionais. Eu gostei tanto da idéia que a **Fratech** está negociando a **gerência** do site na **versão em português**.

Depois foi a vez de conversar com **Pete Goodlife**, sobre como eu palestrar em inglês pode ser uma analogia ao desenvolvimento de software quando possuem times que falam idiomas diferentes. O ponto chave foi, será que conseguimos pensar em um idioma não nativo como a mesma agilidade que pensamos em nosso idioma? Não chegamos a nenhuma resposta, mas o assunto foi bem explorado e nos rendeu algumas horas para tomar cerveja.

Após o painel de encerramento, voltei para o hotel e no dia seguinte pela manhã fui para **Paris**. Mas essa é uma outra história..



Figura 04 – Encontro de alguns participantes frente a ponto turístico de Londres.

Gerenciamento de resultados com o Product Storage Chart



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). É Diretor Editorial da Revista Visão Ágil, Membro da Agile Alliance e foi um dos pioneiros na utilização e divulgação de métodos ágeis no Brasil. Já escreveu artigos para importantes revistas e portais especializados no Brasil e no exterior. Possui as certificações CSM e CSP da Scrum Alliance. Já participou do time de Desenvolvimento do NetBeans, foi criador do framework BoxSQL, fundador do grupo XPNorte e do NUG-BR e frequentemente palestra em eventos sobre processos e tecnologias. Maiores informações em: <http://manoelpimentel.blogspot.com>

Introdução: O Desafio do ROI (Return on Investment)

No mundo dos projetos, há muito tempo, existe uma necessidade latente em se criar formas de gerenciar o retorno proporcionado por um determinado investimento. Com essa premissa, no mundo do desenvolvimento de software, temos grandes dificuldades em usar mecanismos que forneçam formas de mensurar a relação entre o valor investido e o volume de software entregue.

Mas para minimizar essa dificuldade, nos últimos anos, uma idéia que vêm ganhando muita força em função da grande disseminação da **metodologia Scrum**, é o uso de **Business Value** como medida para expressar o valor que aquele projeto irá retornar para a empresa e também uma forma de definir a prioridade relacionada a cada requisito desejado pelo cliente em forma decrescente, pois a equipe irá sempre trabalhar primeiro nos itens de maior valor de acordo com sua capacidade de entrega.

Outra atribuição muito importante dessa medida, é também fornecer meios para acompanhar a velocidade e o volume de entrega que a equipe de desenvolvimento está alcançando, dessa forma, começamos a estabelecer um ponto de equilíbrio da famosa relação entre o investimento financeiro em um projeto e o valor entregue pela equipe em forma de software ou seja, ter mecanismos para tentar maximizar o **ROI(Return on Investment)** de um projeto de software em uma menor faixa de tempo.

Para explicar bem claramente essa idéia, imagine que o Business Value será uma moeda de troca durante o projeto e o cliente empresta um determinado valor dessa moeda para a equipe e esta por sua vez, terá que devolver o valor correspondente em forma de software, ou seja, é uma dívida que a equipe assume com o cliente e que deverá ser amortizada a cada ciclo(Sprint), até que a mesma seja totalmente liquidada (zerada).

O Fluxo do Scrum

É importante lembrar que segundo o próprio **Ken Schwaber**: “*O Product Backlog é priorizado pelo os itens que têm maior probabilidade de gerar valor*”,

BoxSQL - Sprint Dashboard 01 - 15/05/2007 a 29/05/2007

Goal: Entregar o módulo de templates

Daily Evolution	Left	BV Delivered
15-mai-07	50	0
16-mai-07	50	0
17-mai-07	40	10
18-mai-07	40	0
19-mai-07	40	0
20-mai-07	40	0
21-mai-07	30	10
22-mai-07	30	0
23-mai-07	30	0
24-mai-07	30	0
25-mai-07	20	10
26-mai-07	20	0
27-mai-07	20	0
28-mai-07	20	0
29-mai-07	10	10

Impediments	Start Date	End Date
Modelo arquivo SQL compatível com o FireBird	18/5/2007	20/5/2007
Teste em aplicação Oracle 9i e 10g	26/5/2007	26/5/2007



Figura 01 – Exemplo de um típico Burn-Down Chart.

ou seja, no fluxo que o Scrum utiliza, existe um artefato chamado **Product Backlog**, que reúne os **requisitos funcionais e não funcionais** em uma lista priorizada do maior valor (**Business Value**) para o item de menor valor definido pelo **Product Owner** e que no início de cada **Sprint** (Iteração), esses itens serão selecionados pela **equipe** com base nas medidas de **tamanho** ou **complexidade** e de acordo com a meta estabelecida para criar um incremento de versão de produto potencialmente usável.

O Desafio e o Início da solução

Outra grande questão é como definir formas eficientes e eficazes para acompanhar a evolução diária das entregas de Business Value, por isso, como mostra **Figura 01**, o Scrum, oferece o **BurnDown Chart** como uma forma muito prática para fazer esse acompanhamento, pois fornece normalmente um gráfico em linha para mostrar diariamente o tamanho das entregas a partir do total selecionado para aquele sprint ou projeto, de maneira que esse valor vá gradativamente sendo decrementado até chegar em zero.

Mas o meu objetivo com esse artigo, é compartilhar com você, uma solução bem interessante que surgiu devido a situações específicas que vivenciei em alguns projetos. Essas situações me estimularam a criar e usar uma ferramenta adicional que batizei de **Product Storage Chart**, que simplesmente é um gráfico para responder algumas perguntas comuns ao gerenciamento de resultado em projetos baseado em Business Value, influenciado por algumas idéias de controle de níveis de estoque numa linha de produção, exemplo: Em matéria-prima, produção, perdas e beneficiado (Daí o nome Product Storage).

Na verdade, foram várias as perguntas que surgiram durante alguns projetos sobre o acompanhamento das entregas com base em Business Value, porém o conteúdo dessas perguntas geralmente era: “**Em que situação está o valor(Business Value) que eu emprestei para esse projeto?**” ou seja, precisavam saber quanto do Business Value destinado para aquele projeto ainda estava **pendente** no Backlog, se já estava **selecionado** para algum Sprint, se estava **em processo**, ou simplesmente se já havia sido **entregue**.

O Agrupamento de dados

Com base nessas questões, achar a solução não foi difícil, pois bastou apurar o total de **Business Value** agrupado por **Milestone de status (PENDENTE, A FAZER e FEITO)**. Note que esses milestones podem variar de acordo com seu projeto, por exemplo, além dos três milestones citados acima, você pode trabalhar também com os milestones “**EM IMPEDIMENTO**”, “**EM INSPEÇÃO**” e “**REJEITADO**”. Porém por uma questão didática, resolvi mostrar apenas os três milestones mais simples.

Como funciona?

Como você viu no tópico anterior, primeiramente para fornecer as informações para que o gráfico seja gerado, é importante que o total de Business Value seja **agrupado** por Milestones de cada sprint de seu projeto. Portanto, como você pode ver na **Figura 02** temos um resumo de cada Sprint, onde há um quadro que mostra os totais devidamente separados por milestones e também adicionamos duas outras informações nesse quadro, que são:

- O total “**Em Backlog**” - Para agrupar o total de Business Value ainda não selecionado para algum Sprint.
- O total “**Acumulado**” - Para somar quantos Business Value já foram entregues durante todo o projeto, com base na soma das finalizações de todos os Sprints anteriores.

Veja que uma vez de posse desses dados, basta usar alguma ferramenta de planilha eletrônica ou o meio que achar mais conveniente, para gerar de maneira automática um gráfico de barras verticais conforme também mostra a **Figura 02**.

Note que apesar que minha preferência é por gráfico de barras verticais, na prática, esse gráfico pode ser feito com barras horizontais ou gráfico de pizzas, pois na idéia de um Product Storage Chart, o conteúdo é mais importante do que a forma.

Resumo do Sprint 01	
Milestone	BV
Em BackLog	800
Afazer	100
Progresso	50
Feito	50
Acumulado*	50

*Total já entregue no projeto

Product Storage Chart - Sprint 01

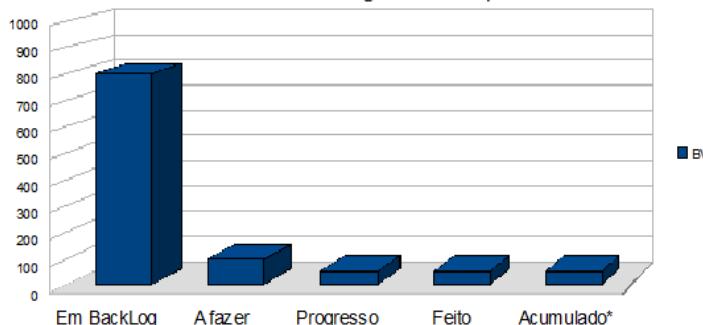


Figura 02 – Resumo com os totais e o Product Storage Chart para Sprint 01.

Observe também na **Figura 02**, que no primeiro Sprint, teremos uma barra de **Acumulado** pequena e principalmente bem menor do que a barra “**Em Backlog**”.

Na **Figura 03**, você poderá observar a mecânica desse gráfico no segundo Sprint. Veja que a barra de acumulado já está bem maior, pois está levando em consideração todos os valores entregues no final do **Sprint 1** e somando com valores já entregues nesse segundo Sprint.

Resumo Sprint 02	
Milestone	BV
Em BackLog	600
Afazer	100
Progresso	50
Feito	150
Acumulado*	350

*Total já entregue no projeto

Product Storage Chart - Sprint 02

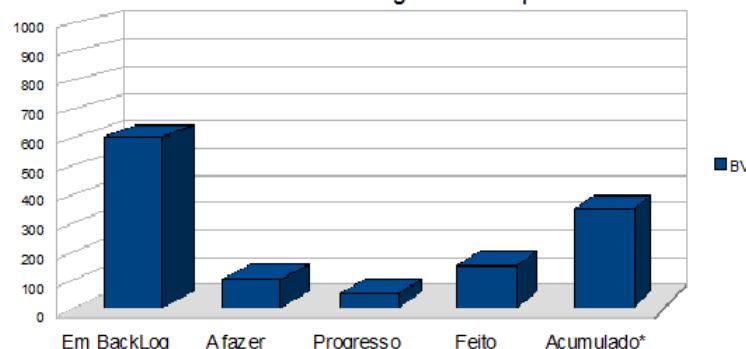


Figura 03 – Product Storage Chart para Sprint 02

Veja que esse mesmo raciocínio será repetido em todos os sprints até zerar a barra “Em BackLog” e ter todos os Business Value na barra “Acumulado” no final do último sprint conforme mostra a **figura 04**

Resumo do Sprint 04	
Milestone	BV
Em BackLog	0
Afazer	0
Progresso	0
Feito	150
Acumulado*	1000

*Total já entregue no projeto

Product Storage Chart - Sprint 04

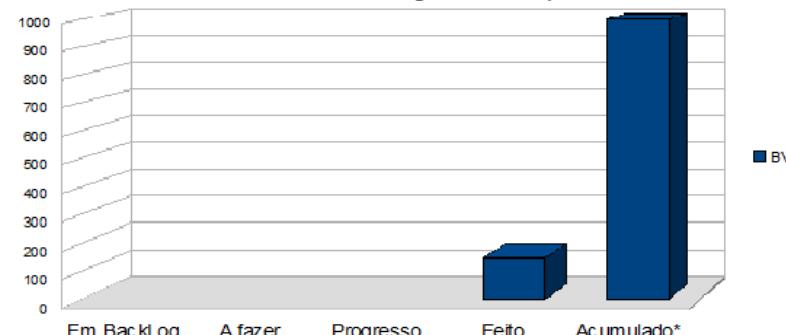


Figura 04 – Product Storage Chart para Sprint 04(Final)

Leituras e Interpretações

Agora que você já viu como funciona essa ferramenta, é importante saber quais as possíveis leituras e interpretações das variações desse gráfico.

A **primeira interpretação** possível, obviamente é o próprio objetivo do gráfico de fornecer uma visão sintetizada do andamento de um Sprint e de um projeto.

A **segunda interpretação** é visualizada ao final de um Sprint, pois será facilmente verificado se todos os valores selecionados para aquele Sprint, foram realmente entregues, ou ficaram pelo caminho (Em progresso, Impedimento, Inspeção, etc).

A **terceira interpretação** é que durante um Sprint, podemos ver se têm um volume muito alto de Business Value em progresso, isso pode significar uma série de efeitos indesejáveis como por exemplo, erro de seleção do tamanho dos itens ou excesso de itens em impedimento.

A **quarta interpretação** possível, é a identificação de possíveis restrições do **lead time** de um sprint, por exemplo, se em uma data próxima ao final do sprint atual, ainda houver uma quantia muito significativa de Business Value em “**INSPEÇÃO**”, pode ser um sintoma de algum problema nas práticas de inspeção e testes que a equipe está enfrentando ou simplesmente os impedimentos não estão sendo removidos em tempo hábil.

É importante observar que é vital que essas leituras, estimulem ações gerenciais por parte da equipe, visando aplicar medidas corretivas ou preventivas para evitar que o projeto sofra grandes impactos na sua qualidade.

Conclusões

Como você viu nesse artigo, é extremamente importante ter mecanismos para gerenciar de maneira simples e eficaz, os avanços e o retorno de valor agregado de um investimento em algum projeto de desenvolvimento de software.

Com essa premissa, você observou que a ferramenta **Product Storage Chart** que apresentei nesse artigo, pode adicionar uma visão mais ampla para o acompanhamento do dia-a-dia da evolução das entregas de Business Value em um típico projeto usando Scrum.

Observe porém, que o Product Storage Chart, em hipótese alguma deseja concorrer com ferramentas já conhecidas pelo grande público como o BurnDown Chart, KanBan, Parking Lot, etc; E sim, o Product Storage Chart pode e deve ser combinado com algumas dessas ferramentas para um melhor e completo gerenciamento de um projeto ágil.

Para finalizar, acredito que esse artigo tenha alcançado seu objetivo principal de divulgar essa técnica de Product Storage Chart e estimular você para que sinta-se encorajado em usá-lo, como uma ferramenta adicional para o gerenciamento de Business Value em seus projetos, usando a metodologia Scrum, porém, lembre-se que essa ferramenta é relativa nova e funcionou muito bem em meus projetos, o que não significa que você não precise fazer uma ou outra adaptação para usá-la em seus próprios projetos. Portanto, no melhor sentido agile, adapte e evolua!

Referências

Schwaber, Ken - Documento eletrônico "**What Is Scrum?**"
Disponível no site da ScrumAlliance - www.scrumalliance.com

Blog do Autor do artigo e da ferramenta
manoelpimentel.blogspot.com

Visão Ágil no FISL 9.0

Nos dias 17, 18 e 19 de Abril de 2008, a Visão Ágil, participou com um stand na área de grupo de usuários do FISL 9.0 (9º Fórum Internacional de Software Livre) em Porto Alegre(RS), portanto, acompanhe aqui os melhores momentos desse grande acontecimento no mundo de TI.



Victor Hugo, dando um show nas explicações sobre **testes e integração contínua** para o visitantes de nosso stand.



Manoel Pimentel em sua palestra sobre **Integração entre o Struts2 e o JasperReports** para relatórios em aplicações web em Java.



Manoel Pimentel interagindo com **Greg Sporar** (Evangelista da Sun para o NetBeans)



Victor Hugo conversando sobre agile com um visitante.



Manoel Pimentel em contato com as tradições do RS, tomando o famoso chimarrão.



Victor Hugo e Manoel Pimentel em entrevista ao pessoal do grupo SAFOS de Belém do Pará (br.groups.yahoo.com/group/safos)



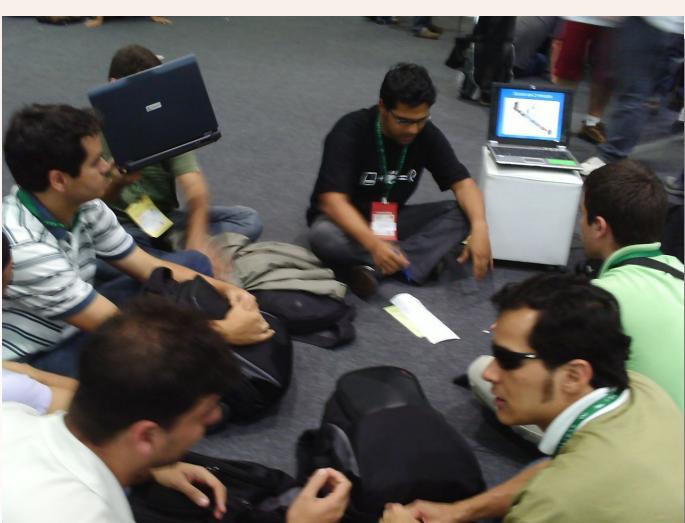
Victor Hugo ladeado por visitantes ilustres, Na esquerda: **Silvestre Mergulhão(XP-Rio)** e na direita: **Vinicius Manhães Teles(XP-Rio)**.



Guilherme Chapiewski em sua palestra sobre **Desenvolvimento Ágil de software com XP e Scrum**, onde apresentou alguns conceitos e um pouco de sua experiência com agile na **Globo.com**

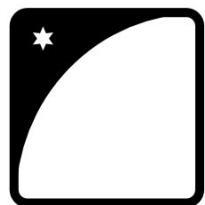


Manoel Pimentel em roda de discussões sobre diversos temas agile (Nessa parte, explicando sobre o funcionamento da técnica de KanBan).



Victor Hugo em roda de discussões sobre diversos temas agile (Nessa parte, explicando sobre o fluxo do Scrum).

Apoio



Caelum

HEPTAGON
TECNOLOGIA DA INFORMAÇÃO



www.tc4digital.com

