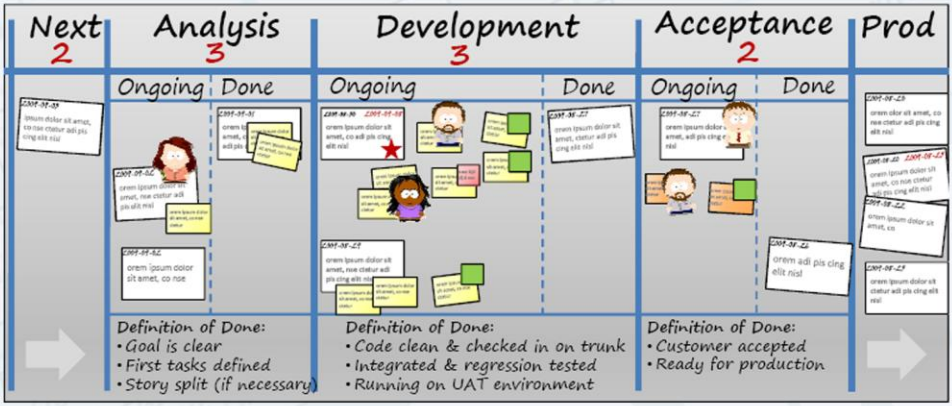


Kanban e Scrum
obtendo o melhor de ambos

Henrik Kniberg & Mattias Skarin

Prefácio por Mary Poppendieck e David Anderson

ENTERPRISE SOFTWARE DEVELOPMENT SERIES

InfoQueue

EDIÇÃO ONLINE GRATUITA

Feito para você

Cortesia da

Visite a página deste livro em:

http://www.infoq.com/br/minibooks/kanban-scrum-minibook

Kanban e Scrum - obtendo o melhor de ambos

Henrik Kniberg & Mattias Skarin
Prefácio por Mary Poppendieck & David Anderson

C4Media, Editora do InfoQ.com.

Este livro é parte da série de livros do InfoQ Enterprise Software Development.

Para informações ou compra deste ou de outros livros do InfoQ, por favor entre em contato com books@c4media.com.

Editora chefe: Diana Plesa
Arte de capa: Bistrian Iosip
Composição: Accurance

Dados Catalográficos da Publicação na Biblioteca do Congresso:

ISBN: 978-0-557-13832-6
Impresso nos Estados Unidos da América

Conteúdo

Prefácio por Mary Poppendieck

Henrik Kniberg é uma das raras pessoas que conseguem extrair a essência de uma situação complicada, separar as idéias principais das distrações acidentais, e fornecer uma explicação cristalina que é incrivelmente fácil de entender. Nesse livro, Henrik faz um trabalho brilhante ao explicar a diferença entre Scrum e Kanban. Ele torna claro que ambos são apenas ferramentas, e o que você realmente precisa é ter um kit completo delas, entender os pontos fortes e fracos de cada uma e como usá-las.

Neste livro você vai aprender o que é exatamente o Kanban, suas qualidades e limitações, e como usá-lo. Você também vai aprender coisas bem interessantes sobre como e quando melhorar o Scrum, ou qualquer outra ferramenta que você estiver usando. Henrik deixa claro que o importante não é a ferramenta que você escolhe usar inicialmente, mas o modo como você constantemente aperfeiçoa a utilização dela e amplia seu kit de ferramentas no decorrer do tempo.

Na segunda parte do livro, Mattias Skarin torna-o ainda mais prático mostrando uma utilização de Scrum e Kanban numa situação real. Você verá um exemplo de como as ferramentas foram usadas separadamente e combinadas para melhorar o processo de desenvolvimento de um software. Você vai notar que não existe uma única ”melhor forma” de fazer as coisas; você tem que refletir e imaginar – baseado na situação – seu próximo passo para uma forma melhor de desenvolver software.

Mary Poppendieck

Prefácio por David Anderson

O Kanban é baseado numa idéia muito simples. As Atividades em andamento devem ser limitadas. Algo novo só deve ser iniciado quando uma peça de trabalho existente é liberada ou quando uma função automática inicia isso.

O Kanban, ou cartão de sinalização, é um sinal visual produzido indicando que novo trabalho pode ser iniciado e que a atividade atual não coincide com o limite acordado. Isso não soa muito revolucionário nem parece afetar profundamente o desempenho, cultura, capacidade e maturidade de uma equipe e a organização na qual está inserida. Mas o impressionante é que afeta! O Kanban parece uma mudança pequena e, no entanto, muda tudo a respeito de uma empresa.

O que percebemos sobre o Kanban é que ele é uma abordagem para mudança gerencial. Ele não é um processo ou ciclo de vida de gerenciamento de projetos ou de desenvolvimento de software. O Kanban é uma abordagem para introduzir mudanças em um ciclo de desenvolvimento de software ou metodologia de gerenciamento de projetos. O princípio do Kanban é que você inicia com o que estiver fazendo agora. Você entende seu processo atual ao mapear o fluxo de valor e ao concordar, em seguida, em limitar as Atividades em andamento (do inglês WIP) para cada estágio desse processo. A partir daí você começa a rastrear as atividades pelo sistema para iniciá-las quando os sinais do Kanban aparecerem.

O Kanban tem sido útil para equipes ágeis de desenvolvimento de software, mas tem ganhado popularidade, igualmente, em equipes que utilizam uma abordagem mais

tradicional. Ele está sendo introduzido como parte de uma iniciativa Lean (enxuta) para moldar a cultura das organizações e encorajar a melhoria contínua.

Porque o WIP é limitado em um sistema Kanban, tudo que fica bloqueado por qualquer motivo tende a parar o sistema. Se certa quantidade de itens de trabalho fica bloqueada, todo o processo pára de funcionar. Isso cria a necessidade de concentrar toda a equipe e toda a empresa na solução do problema para desbloquear o item e restaurar o fluxo.

O Kanban usa um mecanismo de controle visual para acompanhar o trabalho à medida que ele flui através das várias etapas do fluxo de valor.

Tipicamente usa-se um quadro branco com post-its, ou um sistema de cartões eletrônicos. Fazer os dois é, provavelmente, uma boa prática. A transparência que isso gera também contribui para a mudança cultural. Métodos ágeis têm sido bons provendo transparência sobre as atividades em andamento e concluídas, e reportando métricas como velocidade (a quantidade de trabalho finalizado em uma iteração).

O Kanban, no entanto, vai um passo além e dá transparência ao processo e seu fluxo. O Kanban expõe gargalos, filas, variabilidade e desperdício. Tudo que impacta o desempenho da organização em termos de quantidade de trabalho de valor entregue e o tempo de ciclo necessário para entregá-lo. Proporciona aos membros da equipe e às partes interessadas externas a visibilidade sobre os efeitos de suas ações (ou falta de ações). Sendo assim, os primeiros estudos de caso estão mostrando que o Kanban muda o comportamento e incentiva uma maior colaboração no trabalho.

A visibilidade dos gargalos, desperdício e variabilidade, e seus impactos, também incentivam a discussão sobre melhorias e as equipes rapidamente iniciam as melhorias nos seus processos.

Como resultado, o Kanban encoraja a evolução incremental de processos existentes, evolução que é geralmente alinhada a valores Ágeis e Lean. O Kanban não demanda uma revolução drástica no modo como as pessoas trabalham, encoraja, ao invés disso, uma mudança gradual. É mudança entendida e aceita por consenso entre trabalhadores e seus colaboradores.

O Kanban, através da natureza do sistema *pull*[1], encoraja também comprometimento tardio, tanto em priorização de trabalho novo quanto na entrega de trabalho existente.

Tipicamente, os times vão concordar em uma cadência de priorização para atender partes interessadas que estejam contra a maré e decidir a próxima coisa em que trabalhar. Estas reuniões podem ser feitas regularmente porque são normalmente muito curtas.

Uma questão muito simples deve ser respondida, algo como, ”Desde nossa última reunião, 2 vagas ficaram livres. Nosso tempo de ciclo corrente é de 6 semanas para entrega. Quais 2 coisas vocês mais gostariam de ter entregues daqui a 6 semanas?” Isso tem dois desdobramentos. Perguntar uma simples questão geralmente resulta em uma resposta de boa

1 *Pull system*, ou sistema puxado: sistema em que os times é que puxam o trabalho, na medida de sua capacidade, e não o oposto, quando o trabalho é empurrado à equipe – sistema de trabalho mais comum. Nota do Tradutor

qualidade, rapidamente desdobrada, e mantém a reunião breve.

A natureza da questão significa que o comprometimento em que se trabalhará é atrasado até o último momento responsável. Isto aumenta a agilidade gerenciando expectativas, diminuindo tempos de ciclo do comprometimento à entrega e eliminando retrabalho, pois a chance de que prioridades mudem é minimizada.

Uma última coisa sobre Kanban é que o efeito de limitar o WIP fornece previsibilidade de tempo em ciclos e faz as entregas mais confiáveis. A abordagem de "parar a linha de produção" para superar os obstáculos e os erros encontrados, também parece encorajar níveis mais elevados de qualidade e uma queda rápida de retrabalho.

Enquanto tudo isso se tornará evidente com explicações maravilhosamente claras neste livro, como nós chegamos até aqui permanecerá sombrio. Kanban não foi concebido em uma única tarde através de alguma aparição divina incrível. Em vez disso, ele emergiu de vários anos de experiência.

Muitos dos profundos efeitos psicológicos e sociológicos que mudam a cultura, capacidade e maturidade ou organizações nunca foram imaginados. Ao contrário, eles foram descobertos. Muitos dos resultados com Kanban são contra intuitivos. O que parece ser uma abordagem muito mecânica – o limite de WIP e o trabalho puxado – na verdade, tem efeitos profundos sobre as pessoas e como elas interagem e colaboram umas com as outras.

Eu e nem qualquer outra pessoa envolvida com Kanban nos primeiros dias antecipou isto.

Eu segui o que é feito com o Kanban, como uma abordagem de mudança, que tivesse êxito com mínima resistência.

Isto ficou claro para mim já em 2003. Eu também adotei isso pelos benefícios mecânicos. Como eu estava descobrindo através da aplicação de técnicas de Lean durante esse tempo, que se a gestão com atividades em andamento fazia sentido, então limitando isso fazia mais sentido. Levei a sobrecarga do gerenciamento, fora de sua gestão.

Então em 2004, eu decidi tentar implementar um sistema de demanda (*pull system*) a partir dos primeiros princípios. Eu tive uma oportunidade quando um gerente da Microsoft me abordou e me pediu para ajudá-lo a gerenciar mudanças em sua equipe, fazendo manutenção de melhorias em uma aplicação interna de TI.

A primeira implementação foi baseada na solução do sistema de demanda da Teoria das Restrições (TOC), conhecido como Tambor-Pulmão-Corda (*Drum-Buffer-Rope* em inglês). Foi um enorme sucesso: o tempo do ciclo reduziu em 92%; o rendimento aumentou mais de 3 vezes; e a previsibilidade (data limite de desempenho) foi muito aceitável em 98%.

Em 2005, Donald Reinertsen me convenceu a implementar um sistema Kanban completo. Eu tive a oportunidade em 2006, quando eu assumi o comando do departamento de engenharia de software em Corbis, Seattle.

Em 2007, eu comecei a apresentar os resultados. A primeira apresentação foi em maio de 2007 no *Lean New Product Development Summit*, em Chicago. Eu continuei com um espaço aberto no Agile 2007 em Washington DC, em agosto daquele ano. 25 pessoas compareceram. 3 deles eram do Yahoo! Aaron Sanders, Karl Scotland e Joe Arnold. Eles foram para casa para Califórnia, Índia e o Reino Unido e implementou o Kanban com seus times ainda lutando com o Scrum. Eles também iniciaram um grupo de discussão no Yahoo! que na época em que este livro foi escrito tinha quase 800 membros.

O Kanban estava começando a se espalhar e os primeiros a adotarem estavam falando sobre suas experiências.

Agora em 2009, a adoção do Kanban realmente está aumentando e cada vez mais relatórios de campo estão chegando. Nós aprendemos muito sobre o Kanban nos últimos 5 anos e todos nós continuamos a aprender a cada dia. Eu foquei meu próprio trabalho em usar Kanban, escrever sobre Kanban, falar sobre Kanban e pensar sobre Kanban para poder entendê-lo melhor e explicá-lo para os outros. Eu deliberadamente deixei de comparar o Kanban com métodos *Agile* existentes, embora algum esforço tenha sido gasto em

2008 explicando porque o Kanban merecia ser considerado uma abordagem compatível com *Agile*.

Eu deixei para que outros com maior experiência respondessem questões como ”Como é o Kanban comparado ao Scrum?” Eu agradeço muito por Henrik Kniberg e Mattias Skarin terem emergido como líderes nesta área. Você, o trabalhador do conhecimento no campo, precisa de informação para tomar decisões e avançar com seu trabalho. Henrik e Mattias estão atendendo às suas necessidades de uma forma que eu nunca pude.

Eu estou particularmente impressionado com a profunda abordagem para comparação de Henrik e seu trabalho concreto, imparcial e balanceado. Seus desenhos e ilustrações são particularmente informativos e freqüentemente lhe salvam de ter que ler muitas páginas de texto.

O estudo de caso de campo de Mattias é importante porque demonstra que o Kanban é muito mais que teoria e mostra a você com exemplos como ele pode lhe ser útil em sua empresa.

Eu espero que você goste do livro comparando Kanban com Scrum e que ele lhe dê um conhecimento maior sobre *Agile* em geral e tanto Kanban quando Scrum em particular. Caso você queira aprender mais sobre Kanban, por favor visite o web site de nossa comunidade, *The Limited WIP Society*, http://www.limitedwipsociety.org/

David J. Anderson

Sequim, Washington, USA
8 de Julho de 2009.

Introdução

Normalmente nós não escrevemos livros. Nós preferimos gastar nosso tempo nas trincheiras ajudando clientes a otimizarem, testarem e refatorarem seu processo de desenvolvimento e organização. Entretanto, nós notamos uma clara tendência nos últimos tempos e gostaríamos de compartilhar alguns pensamentos sobre isso. Este é um caso típico:

- **Jim:** "Agora finalmente estamos usando totalmente o Scrum!"
- **Fred:** " E como está sendo?"
- **Jim:** "Bom, é muito melhor do que o tínhamos antes..."
- **Fred:** "...mas?"
- **Jim:** "... mas você sabe que somos uma equipe de suporte e manutenção."
- **Fred:** "Sim, e?"
- **Jim:** "Bem, nós amamos toda essa coisa de ordenar itens por prioridade no *product backlog*, equipes auto-organizáveis, *daily scrums*, retrospectivas, etc..."
- **Fred:** "Então qual é o problema?"
- **Jim:** "Nós continuamos falhando em nossos *sprints*."

  Fred: "Por quê?"

- **Jim:** "Porque nós achamos difícil nos comprometermos com um plano de duas semanas. Iterações não fazem muito sentido para nós, nós

apenas trabalhamos no que é mais urgente para hoje. Talvez devêssemos fazer iterações de uma semana?"

- **Fred:** "Você poderia se comprometer com uma semana de trabalho? Você poderá manter o foco e trabalhar em paz por uma semana?"

- **Jim:** "Na verdade não, nós temos novos problemas diariamente. Talvez se fizéssemos *sprints* de um dia..."

- **Fred:** "Seus problemas levam menos de um dia para serem resolvidos?"

- **Jim:** "Não, às vezes eles levam vários dias."

- **Fred:** "Então *sprints* de um dia também não iriam funcionar. Você chegou a considerar a possibilidade de não usar *sprint* nenhum?"

- **Jim:** "Bem, francamente, nós gostaríamos disso. Mas isso não seria contra o Scrum?"

- **Fred:** "O Scrum é somente uma ferramenta. Você escolhe quando e como utilizá-la. Não seja um escravo dela!"

- **Jim:** "Então o que deveríamos fazer?"

- **Fred:** "Você já ouviu falar sobre Kanban?"

- **Jim:** "O que é isso? Qual a diferença entre isso e o Scrum?"

- **Fred:** "Aqui, leia este livro!"

- **Jim:** "Mas apesar disso, nós realmente gostamos do resto do Scrum, eu terei que mudar agora?"

- **Fred:** "Não, você pode combinar as técnicas!"

- **Jim:** "Que? Como?"

- **Fred:** “Apenas continue lendo...”

Propósito desse livro

Se você está interessado no desenvolvimento de software ágil Você provavelmente já ouviu falar sobre Scrum, e você também pode ter ouvido falar sobre Kanban. Uma pergunta que ouvimos com mais e mais freqüência é: "Mas o que é Kanban, e como ele se compara ao Scrum?" Onde é que eles se complementam? Existem alguns tipos de conflitos?

O objetivo deste livro é clarear o nevoeiro, assim você pode descobrir como Kanban e Scrum podem ser úteis no seu ambiente. Deixe-nos saber se conseguiremos!

Parte I – Comparação

A primeira parte do livro é uma tentativa de fazer uma comparação objetiva e prática entre Scrum e Kanban. É uma versão levemente atualizada do artigo original, "Kanban VS Scrum", de abril de 2009. Esse artigo se tornou popular, de modo que decidi transformá-lo em um livro e pedir a meu colega Mattias para combiná-lo com um estudo de caso "das trincheiras" de um de nossos clientes. Coisa boa! Sinta-se livre para pular para a parte II se você preferir começar com o estudo de caso; eu não vou ficar ofendido. Bem, talvez apenas um pouco.

/Henrik Kniberg

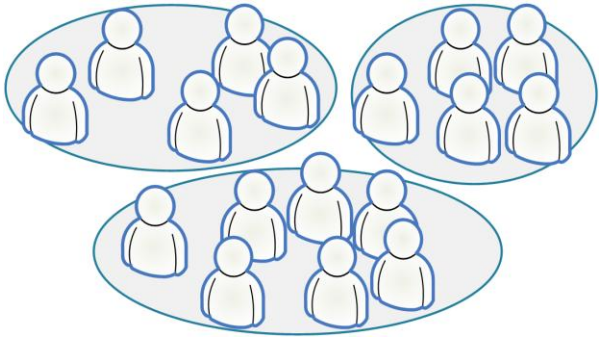
1

O que são mesmo Scrum e Kanban?

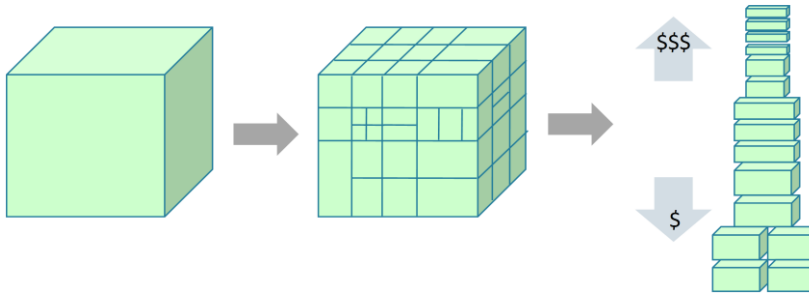
OK! Vamos tentar resumir Scrum e Kanban em menos de 100 palavras cada.

Scrum em poucas palavras

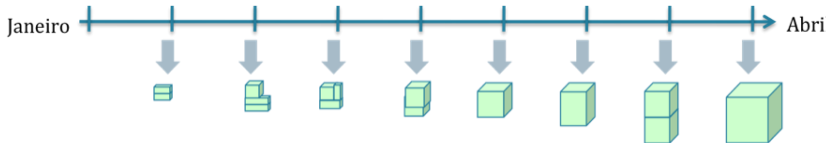
- **Divida sua organização** em equipes pequenas, multifuncionais e auto-organizadas.



- **Divida o seu trabalho** em uma lista de entregáveis pequenos e concretos. Classifique a lista por prioridade e estime o esforço relativo de cada item.



- **Divida o tempo** e pequenas e curtas iterações de duração fixa (geralmente 1 – 4 semanas), com código potencialmente entregável demonstrado depois de cada iteração.



- **Otimize o plano de entrega** e atualize prioridades em colaboração com o cliente, baseados em *insights* através de inspeção da entrega depois de cada iteração.

- **Otimize o processo** executando uma retrospectiva depois de cada iteração.

Então ao invés de um **grande grupo** gastando **um monte de tempo** construindo uma **grande coisa**, temos uma **equipe pequena** gastando um **tempo curto** construindo uma **pequena coisa**. Mas **integrando regularmente** para ver o todo.

133 palavras... Perto o suficiente.

Para mais detalhes, verifique "Scrum e XP direto das Trincheiras". O livro é grátis e está online. Eu conheço o autor, ele é um cara legal :o)

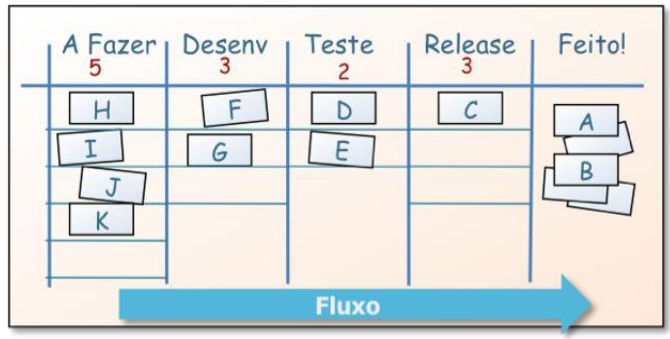
http://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches

Para mais *links* sobre Scrum vá em http://www.crisp.se/scrum

Kanban em poucas palavras

- **Visualize o fluxo de trabalho**
    - Divida o trabalho em partes, escreva cada item em um cartão e coloque na parede.
    - Use colunas nomeadas para ilustrar onde cada item está no fluxo de trabalho.
- **Limite o trabalho em progresso (*WIP - work in progress*)** – associe limites explícitos para quantos itens podem estar em progresso em cada estado do fluxo de trabalho.

- **Acompanhe o tempo de execução da tarefa** (tempo médio para completar um item, algumas vezes chamado de "tempo de ciclo"), otimize o processo para tornar o tempo de execução o menor e mais previsível possível.



Nós coletamos *links* úteis de Kanban em:
http://www.crisp.se/kanban

2

Então como Scrum e Kanban se relacionam?

Scrum e Kanban são ambos ferramentas de processo

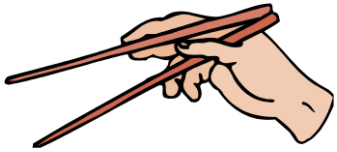
Ferramenta = qualquer coisa utilizada com a finalidade de realizar uma tarefa ou atingir um objetivo. Processo = como você trabalha.

Scrum e Kanban são ferramentas de processo que, em certa medida, te ajudam a trabalhar de maneira mais eficaz, dizendo a você o que fazer. Java também é uma ferramenta, ela lhe fornece uma maneira mais simples de programar. A escova de dentes também é uma ferramenta, que ajuda você a alcançar seus dentes para que você possa limpá-los.

Compare ferramentas para compreensão, Não julgamento

Faca ou garfo – qual é a melhor ferramenta?

Uma pergunta completamente sem sentido certo? Porque a resposta depende do seu contexto. Para comer almôndegas provavelmente o garfo é o melhor. Para cortar cogumelos a faca provavelmente é melhor. Para batucar na mesa qualquer um dele serve. Para comer um bife você provavelmente vai querer usar as duas ferramentas juntas. Para comer arroz... bem... alguns preferem garfo, enquanto outros preferem os pauzinhos.



Sendo assim, quando comparamos ferramentas devemos ter cuidado. Compare para compreensão, não para julgamento.

Nenhuma ferramenta é completa, nenhuma ferramenta é perfeita

Como quaisquer ferramentas, Scrum e Kanban não são perfeitos nem tampouco completos. Eles não lhe dizem *tudo* o que você precisa fazer, eles apenas lhes oferecem algumas restrições e orientações. Por exemplo, Scrum lhe restringe a ter iterações de tempo fixo e equipes multifuncionais, enquanto que Kanban lhe restringe a utilizar quadros visíveis e limitar o tamanho de suas linhas de produção.

O interessante aqui é que o valor de uma ferramenta é que ela *limita suas opções*. Uma ferramenta-processo que lhe permite fazer tudo não é lá muito útil. Nós poderíamos chamar este processo "Faça Qualquer Coisa" ou que tal "Faça a Coisa Certa". O processo "Faça a Coisa Certa" certamente funciona, é uma bala de prata! Afinal, se ele não funcionar, é óbvio que você não está seguindo o processo :o)

Usar as ferramentas adequadas vai lhe ajudar a ter sucesso, mas isso não vai lhe garantir o sucesso. É fácil confundir sucesso/falha de um *projeto* com sucesso/falha da *ferramenta*.

- Um projeto pode ter sucesso por utilizar uma boa ferramenta.
- Um projeto pode ter sucesso apesar de uma péssima ferramenta.
- Um projeto pode falhar por causa de uma péssima ferramenta.
- Um projeto pode falhar apesar de utilizar uma boa ferramenta.

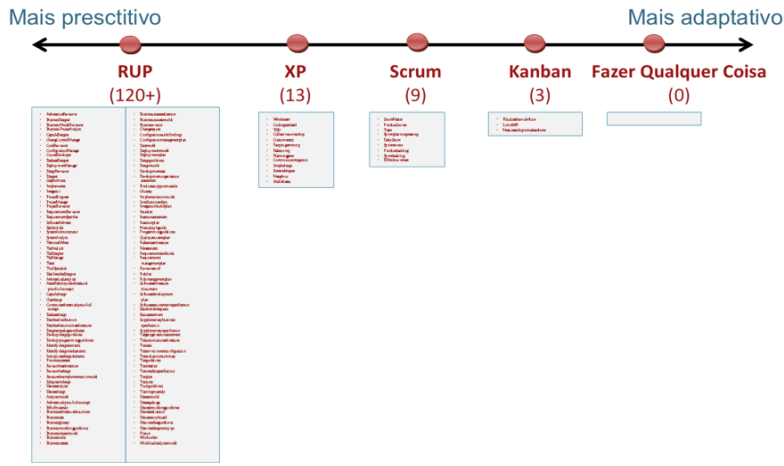
O Scrum é mais prescritivo que o Kanban

Podemos comparar ferramentas analisando quantas regras elas oferecem. *Prescritivo* significa "mais regras a seguir" e *adaptativo* significa "menos regras a seguir". *Prescritivo* 100% significa que você não precisa usar seu cérebro, já há regra para tudo. *Adaptativo* 100% significa faça qualquer coisa, não existe nenhum tipo de regras ou restrições. Como podemos ver, os dois extremos da escala acabam se tornando ridículos.

Métodos Ágeis são às vezes chamados de métodos *leves* (*lightweight),* especificamente porque eles são menos prescritivos que os métodos tradicionais. De fato, o primeiro princípio do Manifesto Ágil é "Indivíduos e Interações sobre Processos e Ferramentas".

Scrum e Kanban são ambos altamente adaptativos, mas *falando relativamente*, Scrum é mais prescritivo que Kanban. Scrum lhe dá mais restrições, por conta disso, deixa menos opções abertas. Por exemplo, o Scrum prescreve o uso de iterações de duração fixa, o Kanban não.

Vamos comparar mais alguns processos e ferramentas na escala prescritivo x adaptativo:



RUP é bastante prescritivo – ele tem mais de 30 papéis, mais de 20 atividades e mais de 70 artefatos; uma quantidade enorme de coisas para aprender. Entretanto não consideramos que você vai usar tudo isso; consideramos que você vai selecionar um subconjunto adequado para seu projeto.

Infelizmente isso parece ser difícil na prática. "Hmmmm… vamos precisar do artefato *Registro da Auditoria de Configuração*? Vamos precisar do papel de *Gerente de controle de mudanças*? Não tenho certeza, então vamos

mantê-los caso for necessário." Esta pode ser uma das razões pelas quais as implementações do RUP normalmente são consideradas muito pesadas quando comparadas aos métodos ágeis como Scrum e XP.

XP (*eXtreme programming* – programação extrema) é bastante prescritiva comparada ao Scrum. Ela inclui quase tudo do Scrum + algumas boas práticas de engenharia bem específicas como desenvolvimento orientado a testes e programação em par.

Scrum é menos prescritivo que XP, uma vez que ele não prevê nenhuma prática específica de engenharia. Porém, Scrum é mais prescritivo que Kanban, uma vez que prescreve coisas como iterações e equipes multifuncionais.

Uma das principais diferenças entre Scrum e RUP é que no RUP você recebe coisas demais, e você supostamente remove o material que não precisa. No Scrum você recebe muito pouco, e supostamente adiciona o material que lhe falta.

Kanban deixa quase tudo em aberto. As únicas restrições são: Visualize Seu Fluxo de Trabalho e Limite Suas Atividades em Andamento. Apenas a alguns centímetros de Faça Qualquer Coisa, mas ainda assim surpreendentemente poderoso.

Não se prenda a uma única ferramenta!

Misture e combine as ferramentas de que você precisa! Eu mal posso imaginar uma equipe Scrum de sucesso que não inclui, por exemplo, a maioria dos elementos do XP. Muitas equipes Kanban usam reuniões diárias (uma prática Scrum). Algumas equipes Scrum escrevem alguns dos seus itens de *backlog* como casos de uso (uma prática RUP) ou limitam seus tamanhos de fila (uma prática Kanban). O que funcionar para você.

Miyamoto Musashi (famoso samurai do século 17, famoso por sua técnica de luta das espadas gêmeas) disse, brilhantemente:

Não desenvolva apego a nenhuma arma ou escola de combate.

- Miyamoto Musashi

Contudo, preste atenção nas limitações de cada ferramenta. Por exemplo, se você usa Scrum e decide parar de usar iterações de tempo fixo (ou qualquer outro aspecto central do Scrum), então não diga que você está usando Scrum. Scrum é suficientemente minimalista tal como é, se você remover coisas e continuar chamando isso de Scrum a palavra ficará sem sentido e confusa. Chame de algo como "inspirado em Scrum" ou "uma derivação do Scrum", ou que tal "Scrumish" :o)

3

Scrum prescreve papéis

Scrum prescreve 3 papéis: *Product Owner* (cria a visão do produto e prioridades), Equipe (implementa o produto) e *ScrumMaster* (remove impedimentos e fornece liderança de processo).

Kanban não prescreve papel nenhum.

Isso não significa que você não possa ou não deva ter o papel de um *Product Owner* em Kanban! Significa apenas que você não *precisa*. Tanto em Scrum quanto em Kanban, você é livre para incluir quaisquer papéis adicionais que precisar.

Mas tenha cuidado ao incluir papéis, tenha certeza de que os papéis adicionais realmente agregam valor e não entram em conflito com outros elementos do processo. Você tem certeza de que precisa do papel de Gerente de Projetos? Em um grande projeto talvez seja uma grande idéia, talvez seja a pessoa que ajuda a sincronizar múltiplos projetos e *Product Owners* entre si. Em um projeto pequeno esse papel pode ser um desperdício, ou pior, pode levar a uma sub-otimização e ao micro gerenciamento.

O raciocínio comum tanto em Scrum quanto em Kanban é "menos é mais". Então, na dúvida, comece com menos.

No resto do artigo eu usarei o termo "*Product Owner*" para representar aquele que define as prioridades de uma equipe, independente do processo usado.

4

Scrum prescreve iterações em *time-box*

Scrum é baseado em iterações de tempo fixo. Você pode escolher a duração da iteração, mas a idéia geral é manter a mesma duração de iteração por um período de tempo, desta forma estabelecendo uma *cadência*.

- **Início da iteração:** Um plano de iteração é criado, isto é, a equipe puxa um número específico de itens do *product backlog*, baseado nas prioridades do *Product Owner* e no quanto a equipe acha que pode completar em uma iteração.

- **Durante a iteração:** A equipe concentra-se em entregar os itens com que se comprometeu. O escopo da iteração é fixado.

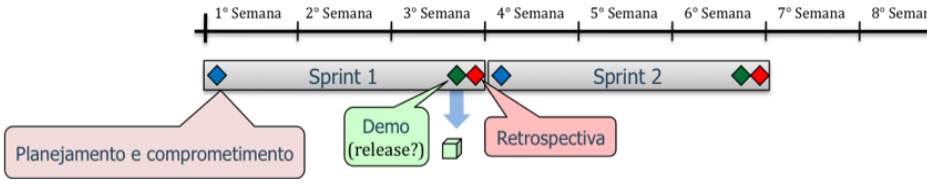
- **Fim da iteração:** A equipe demonstra o código de trabalho para as partes interessadas relevantes. Este código, idealmente, deveria ser *potencialmente entregável* (isto é, testado e pronto para funcionar). Em seguida, a equipe faz uma retrospectiva para discutir e melhorar seu processo.

Então o Scrum é uma única cadência de tempo fixo que combina três diferentes atividades: planejamento, melhoria de processo e (idealmente) *release*.

No Kanban, iterações de duração fixa não são prescritas. Você pode escolher quando planejar, melhorar processo e entregar. Pode escolher fazer essas atividades numa periodicidade regular ("*release* toda segunda"), ou por demanda ("*release* sempre que tivermos algo útil a entregar").

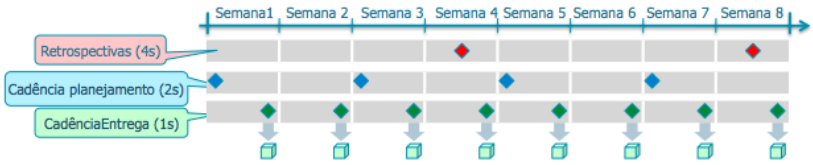
Equipe #1 (cadência única)

"Nós fazemos iterações do Scrum"



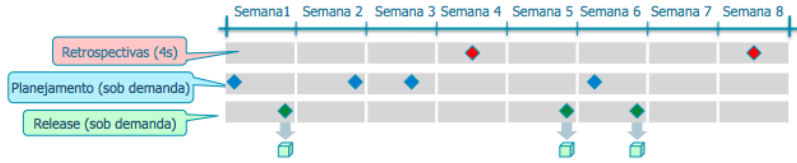
Equipe #2 (três cadências)

"Nós temos três cadências diferentes. Toda semana nós liberamos tudo o que está pronto para a *release*. Toda segunda semana nós temos uma reunião de planejamento e atualizamos nossas prioridades e planos para *release*. Toda quarta semana nós temos uma reunião de retrospectiva para ajustar e melhorar nosso processo"



Equipe #3 (mais orientada para eventos)

"Nós Iniciamos uma reunião de planejamento sempre que esgotamos nosso trabalho a fazer. Iniciamos uma *release* sempre que temos um grupo Mínimo Aceitável de Funcionalidades (MAF's) pronto para entrega. Iniciamos um círculo de qualidade espontâneo sempre que nos deparamos com o mesmo problema pela segunda vez. Também fazemos uma retrospectiva mais profunda toda quarta semana."

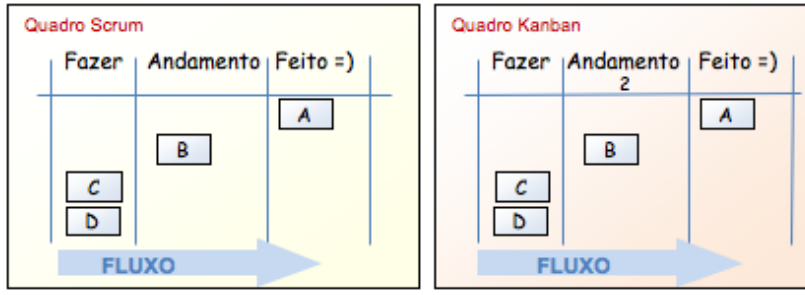


5

Kanban limita WIP por estado de fluxo de trabalho, Scrum por iteração

Em Scrum, o *sprint backlog* mostra quais são as tarefas a serem executadas durante a iteração corrente (= "*sprint*" na linguagem Scrum). Geralmente ele é representado usando cartões na parede, conhecido também por quadro do Scrum ou quadro de tarefas.

Então qual é a diferença entre o quadro de Scrum e o quadro de Kanban? Vamos começar com um projeto trivialmente simples e compará-los:



Em ambos os casos nós estamos controlando um grupo de itens ao longo do progresso pelo fluxo todo. Nós selecionamos três estados: Não iniciado, Iniciado e Pronto. Você pode escolher os estados que quiser – algumas equipes

adicionam outros estados como, por exemplo, Combinar, Testar, Entregar, etc. Mas não se esqueça da regra *menos é mais.*

Então qual a diferença entre esses dois exemplos de quadros? Sim – o pequeno 2 na coluna do meio no quadro Kanban. E isto é tudo. Este 2 significa "não pode haver mais de 2 itens nesta coluna ao mesmo tempo".

Em Scrum não há nenhuma regra advertindo a equipe para não colocar todas as atividades na coluna "Em execução" ao mesmo tempo! Porém, há um limite implícito, já que a interação, por si só, tem um escopo fixo. Neste caso, o limite implícito por coluna é 4, já que há apenas 4 itens em todo o quadro. Então, Scrum limita as atividades em andamento[2] indiretamente, enquanto Kanban as limita diretamente.

A maioria das equipes Scrum aprende na prática que não é uma boa idéia ter muitas atividades na coluna "em execução", e criam uma cultura de tentar finalizar as atividades atuais antes de começar outras.

Alguns até decidem claramente limitar os números de atividades permitidas na coluna "em execução" e então – tchammm! – o quadro Scrum se transforma em um quadro Kanban!

Então, tanto Scrum quanto Kanban limitam as atividades em andamento, mas de formas diferentes. Equipes Scrum geralmente medem a velocidade – quantos itens (ou unidades de medidas correspondentes como "pontos da estória"[3]) serão feitos por iteração.

Uma vez que a equipe sabe sua velocidade, ela torna-se seu limite de atividades em andamento. Uma equipe que tem velocidade média de 10, geralmente, não irá colocar mais de 10 itens (ou "pontos da estória") em uma iteração.

---

2 NT.: WIP (*Work Itens in Progress*) no original.

3 NT.: *History Points* no original

Desta forma, em Scrum as *atividades em andamento são limitadas por unidade de tempo.*

Em Kanban as *atividades em andamento são limitadas pelo fluxo de trabalho.*

No exemplo de Kanban acima, no máximo 2 itens podem estar no estado "Iniciado" ao mesmo tempo, independente da cadência. Você precisa escolher qual limite aplicar para cada estado dentro do fluxo de trabalho, mas a idéia geral é limitar as atividades em andamento de *todos* os estados e suas fases, iniciando o quanto antes e terminando o mais tarde possível dentro dos limites estabelecidos.

Desta forma, no exemplo acima devemos considerar um limite para as atividades em andamento "Não iniciado" também (ou como quer que se chame o primeiro estado das suas tarefas).

Uma vez que tenhamos os limites das atividades em andamento devidamente estabelecidos, podemos começar a medir e prever o tempo de execução do ciclo, isto é, o tempo médio que cada item leva para cumprir todo o ciclo através do quadro.

Prever o tempo gasto em todo o fluxo nos permite uma maior fidelidade aos ANS - Acordos de Nível de Serviço (ou do inglês *SLAs – Service-Level Agreements*) e fazer planos de entrega mais realistas.

Se o tamanho dos itens variar muito, talvez você deva, alternativamente, considerar ter os limites das atividades em andamento definidos em pontos da história, ou qualquer outra unidade de medida que você esteja usando.

Algumas equipes investem e se esforçam em dividir itens, aproximadamente, nos mesmos tamanhos para evitar esses tipos de considerações e reduzir o tempo gasto nestas estimativas (você pode até considerar a estimativa a ser gasta).

É mais fácil criar um fluxo regular se os itens estiverem aproximadamente uniformizados.

6

Ambos são empíricos

Imagine se houvesse botões nestes medidores, e você pudesse configurar o seu processo apenas girando os botões. "Eu quero alta capacidade, baixo tempo de execução, alta qualidade, e alta previsibilidade. Então, eu giraria os botões para 10, 1, 10, 10 respectivamente."

Não seria ótimo? Infelizmente, não existem tais controles diretos. Não que eu saiba, pelo menos. Avise-me se você encontrar algum.

Em vez disso o que temos é um monte de controles *indiretos.*

Tanto Scrum quanto Kanban são empíricos no sentido que se espera que você experimente o processo e personalize ao seu ambiente. Na verdade, você *tem que* experimentar. Nem Scrum nem Kanban fornecem todas as respostas – eles apenas fornecem um conjunto básico de restrições para conduzir o seu próprio processo de melhoria.

- O Scrum diz que você deve ter uma equipe multifuncional. Então, quem deve estar em qual equipe? Não sei, experimente.

- O Scrum diz que a equipe define quanto trabalho realizar em um *sprint*. Então, quanto trabalho eles devem escolher? Não sei, experimente.

- O Kanban diz que você deve limitar o WIP. Então, qual deve ser esse limite? Não sei, experimente.

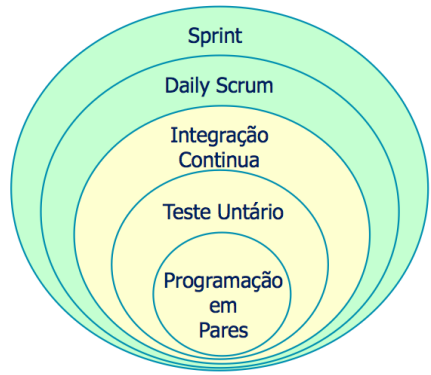
Como eu mencionei antes, Kanban impõe menos restrições que Scrum. Isso significa que você tem mais parâmetros para pensar, mais botões para apertar. Isso pode ser tanto uma desvantagem quanto uma vantagem, dependendo do seu contexto. Quando você abre a janela de configuração de uma ferramenta de software, prefere ter 3 opções para ajustar, ou 100 opções para ajustar? Provavelmente algo no meio disso. Depende de quanto você precisa ajustar e de quanto você conhece da ferramenta.

Então, vamos imaginar que nós reduzimos o limite de WIP baseados na hipótese de que isso irá melhorar nosso processo. Vamos, então, observar algumas coisas como capacidade, *lead time*, qualidade e previsibilidade de mudanças. Iremos tirar conclusões dos resultados e daí mudar mais algumas coisas, melhorando continuamente nosso processo.

Há muitos nomes para isso. *Kaizen* (melhoria contínua no jargão do *Lean*), Inspecionar e Adaptar (no jargão do Scrum), Controle de Processo Empírico, ou, por que não, o Método Científico.

O elemento mais importante disso é o *ciclo de feedback*. Mude alguma coisa => Saiba como foi => Aprenda com isso => Mude alguma coisa novamente. Falando de um modo geral, você quer o ciclo de *feedback* o mais rápido possível, para que seja possa adaptar o processo rapidamente.

Em Scrum, a iteração básica de um *feedback* é o *sprint*. Há mais, porém, especialmente se você combinar com o XP (*eXtreme Programming*):



Quando feito corretamente, Scrum + XP lhe oferece vários ciclos de *feedback* extremamente valiosos.

O ciclo interno de *feedbacks*, programação em par, é um ciclo de *feedbacks* de poucos segundos. Defeitos são encontrados e corrigidos nos segundos da criação (“Ei, essa variável não deveria ser um 3?"). Isso é um “nós estamos construindo o programa corretamente?" ciclo de *feedbacks*.

O ciclo externo de *feedbacks*, o *sprint*, oferece um ciclo de *feedbacks* de poucas semanas. Isso é um “nós estamos construindo o programa corretamente?" ciclo de *feedbacks*.

Então, e sobre o Kanban? Bem, primeiramente você pode (e provavelmente deve) colocar todos os ciclos de *feedback* acima no seu processo, usando ou não Kanban. Kanban, então, lhe oferece algumas métricas, em tempo real, muito úteis.

- Tempo médio de execução. Atualizado a cada vez que um item atinge o "*Done*" (ou o que vocês chamam de sua coluna mais à direita).

- Gargalos. Um sintoma típico é que a coluna X está repleta de itens, enquanto a coluna X+1 está vazia. Procure por "bolhas de ar" em seu quadro.

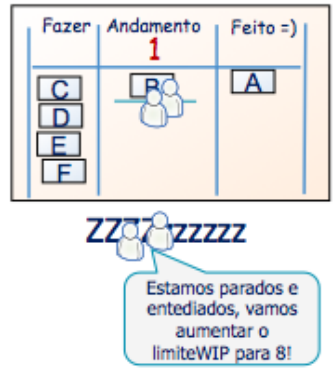
A coisa agradável sobre métricas em tempo real é que você pode escolher o comprimento do seu ciclo de *feedback*, com base na freqüência com que você está disposto a analisar as métricas e fazer mudanças. Ciclos muito longos significam que melhorias no seu processo se darão de forma lenta. Ciclos muito curtos podem fazer com que o seu processo não tenha tempo para estabilizar entre cada mudança, o que pode causar perda.

Na verdade, o comprimento do ciclo de *feedback* em si é uma das coisas com as quais você pode experimentar... como uma espécie de meta-ciclo de *feedback*. OK, parando por aqui.

Exemplo: Experimentando com WIP limites em Kanban.

Um dos pontos de ajuste (*tweak points*) típicos de Kanban é o limite de trabalho em progresso (WIP). Então, como vamos saber se o nosso está correto?

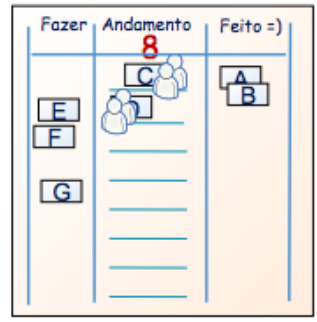
Vamos dizer que temos uma equipe de 4 pessoas, e nós decidimos começar com um limite de WIP, de 1.



Sempre que começarmos a trabalhar em um item, não poderemos iniciar um novo item até que o primeiro esteja concluído. Dessa maneira, o item em que estamos trabalhando tende a ser concluído mais rapidamente.

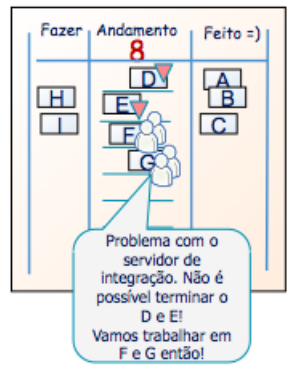
Ótimo! Mas então fica claro que normalmente não é possível as quatro pessoas trabalharem no mesmo item (no contexto deste exemplo), temos então gente sem nada para fazer. Se isso acontece de vez em quando não é problema, mas se acontece freqüentemente o tempo de execução médio vai acabar aumentando. Resumindo, um WIP de 1 significa que uma vez que entrem os itens vão sair de "Iniciado" rapidamente, mas antes disso vão ficar presos em "Não Iniciado" mais tempo do que o necessário, portanto o tempo de execução do fluxo todo vai ficar desnecessariamente alto.

Então, se WIP de 1 era pouco, que tal aumentar para 8?

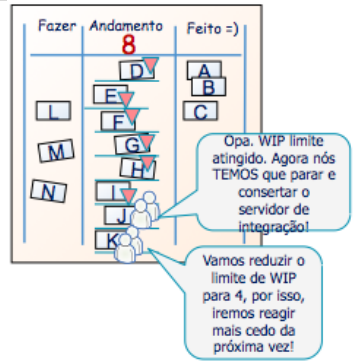


Funciona bem por algum tempo. Descobrimos que, na média, trabalhar em pares faz com que o trabalho acabe mais rápido. Então, numa equipe de quatro pessoas, normalmente teremos dois itens em andamento ao mesmo tempo. O WIP de 8 é um limite superior, então não há problema em ter menos itens em andamento.

Imagine agora, que nos deparamos com um problema com o servidor de integração, portanto não podemos concluir completamente qualquer um dos itens (nossa definição de "*Done*" inclui a integração). Esse tipo de coisa acontece algumas vezes corretamente?



Já que não podemos completar o item D ou E, começaremos a trabalhar no item F. Não podemos integrar nenhum, então nós começamos o novo item G. Depois de um tempo nós atingimos nosso limite do Kanban - 8 itens em "Em Desenvolvimento".



Nesse ponto não podemos começar mais itens. Ei, é melhor nós concertarmos esse maldito servidor de integração! O limite das atividades em andamento (WIP) levou-nos a reagir e corrigir o gargalo em vez de apenas aumentar um monte de atividade inacabada.

Isso é bom. Mas se o limite das atividades em andamento fosse 4, nós teríamos reagido muito mais cedo, assim fornecendo-nos um melhor tempo médio de execução (*lead time*).

Portanto, isso é um equilíbrio. Medimos o tempo médio de execução e ficamos otimizando nosso limite das atividades em andamento para aperfeiçoar o tempo de execução.

Depois de um tempo podemos encontrar os itens acumulando na coluna "Não Iniciado". Talvez seja hora de adicionar um limite das atividades em andamento nessa coluna também.

Por que precisamos de uma coluna “Não Iniciado”? Bem, se o cliente estivesse sempre disponível para falar com a equipe o que fazer na seqüência quando perguntarem.

Então a coluna “Não Iniciado” não seria necessária.

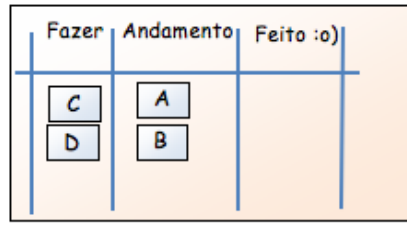
Mas nesse caso o cliente às vezes não está disponível, então a coluna “Não Iniciado” fornece a equipe uma pequena reserva de trabalho nesse item.

Experimente! Ou, como dizem os Evangelistas de Scrum (*Scrumologists*), Inspecione & Adapte!

7

Scrum resiste a mudanças dentro de uma iteração

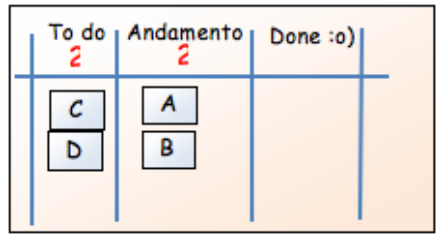
Digamos que nosso quadro do Scrum se pareça com este:



E se alguém aparecer querendo adicionar E ao quadro?

Uma equipe de Scrum tipicamente diria algo como "Não, desculpe, estamos comprometidos com A+B+C+D nesse *sprint*. Mas sinta-se livre para adicionar E ao *product backlog*. Se o *Product Owner* considerar que isto tem alta prioridade, ele adicionará isso no próximo *sprint"*. *Sprints* do tamanho certo dão ao time tempo focado suficiente para conseguir fazer alguma coisa, enquanto ainda permitem que o *Product Owner* gerencie e atualize prioridades com periodicidade regular.

Mas então, o que uma equipe de Kanban diria?



Um Kanban poderia dizer ”Fique à vontade para adicionar E na coluna Não Iniciado. Entretanto, o limite é 2 para aquela coluna, então você terá de remover C ou D nesse caso. Estamos trabalhando em A e B neste momento, mas assim que nós tivermos capacidade, vamos pegar o item do topo do Não Iniciado”.

Assim, o tempo de resposta (quanto demora a responder a uma mudança de prioridades) de uma equipe de Kanban é tão longo quanto à capacidade de se tornar disponível, seguindo o princípio geral de ”um item fora = um item dentro” (controlado pelos limites de trabalho em andamento).

Em Scrum, o tempo de resposta é em média a metade da duração do *sprint*.

Em Scrum, o *Product Owner* não pode alterar o quadro Scrum quando a equipe já estiver comprometida com determinados de itens na iteração. Já em Kanban, você precisa definir suas próprias regras básicas sobre quem tem permissão de modificar o que no quadro. Normalmente para o *Product Owner* é reservada uma coluna do tipo “A Fazer” ou “*Done*” ou “*Backlog*” ou ainda “Proposta” bem à esquerda, onde ele pode fazer as modificações que bem entender.

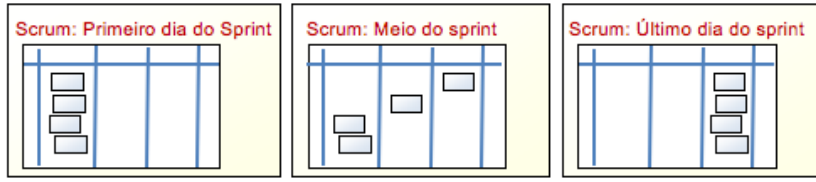
No entanto, estas duas abordagens não são mutuamente exclusivas. Uma equipe Scrum *pode* decidir por deixar o *Product Owner* modificar as prioridades no meio do *sprint* (ainda que isto devesse ser considerado normalmente uma exceção).

E uma equipe Kanban *pode* decidir incluir restrições sobre quando as prioridades podem ser modificadas. Uma equipe Kanban pode ainda decidir usar iterações com duração fixa e compromissos predefinidos, tal como em Scrum.

8

Quadro Scrum é limpo entre cada iteração

Um quadro do Scrum se parece algo como isto durante as diferentes fases de um *sprint*.

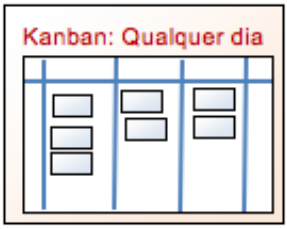


Quando o *sprint* acaba, o quadro é limpo – todos os itens são removidos. O *sprint* novo é iniciado e depois da reunião de planejamento do *sprint*, temos um novo quadro do Scrum, com novos itens na coluna mais à esquerda.

Tecnicamente isso é desperdício, mas para equipes Scrum experientes isso não costuma demorar muito, e o processo de limpeza do quadro pode dar um bom senso de realização e fechamento.

Algo como lavar os pratos depois do jantar - fazê-lo dói, mas nos sentimos bem depois.

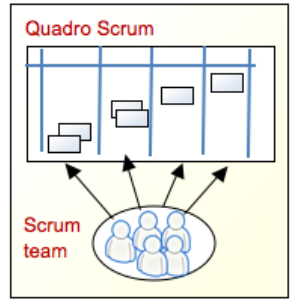
Em Kanban, o quadro é normalmente uma coisa de persistência - você não precisa limpá-lo e começar de novo.



9

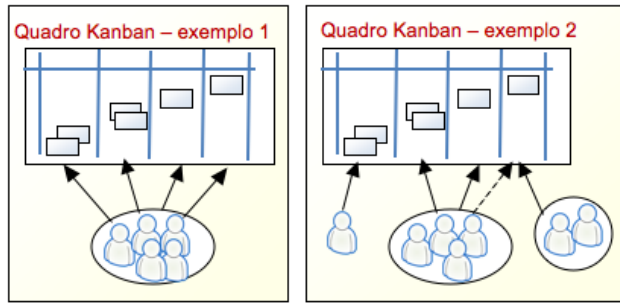
Scrum prescreve equipes multifuncionais

Um quadro de Scrum pertence a exatamente uma equipe Scrum. Uma equipe Scrum é multifuncional, ela contém todas as habilidades necessárias para completar todos os itens contidos na iteração. Um quadro de Scrum é geralmente visível a qualquer pessoa que esteja interessada, mas somente aqueles que fazem parte da equipe Scrum podem editá-lo – é a sua ferramenta para gerenciar o seu compromisso para esta iteração.



Em Kanban, equipes multifuncionais são opcionais, e o quadro de atividades não precisa pertencer exclusivamente a uma única equipe. Um quadro de atividades está relacionado a um fluxo de trabalho, não necessariamente a uma equipe.

Aqui estão dois exemplos:



Exemplo 1: Todo o quadro de atividades é usado por uma única equipe multifuncional. Assim como no Scrum.

Exemplo 2: O *Product Owner* define prioridades na coluna 1. Uma equipe multifuncional de desenvolvedores faz o desenvolvimento (coluna 2) e testa (coluna 3). A entrega (coluna 4) é feita por uma equipe especializada. Existe certa sobreposição de competências, dessa maneira se a equipe que faz a entrega acaba se tornando um gargalo, um membro da equipe de desenvolvimento irá ajudá-los na atividade de entrega.

Assim, em Kanban é necessário estabelecer algumas regras básicas definindo quem usa e como deve ser usado o quadro de atividades, sendo que você pode experimentar com essas regras até aperfeiçoar o fluxo.

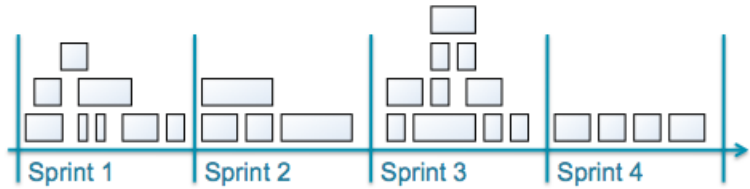
10

Itens do Scrum *backlog* devem caber num *sprint*

Ambos Scrum e Kanban são baseados no desenvolvimento incremental, isto é, quebrar o trabalho em pedaços menores.

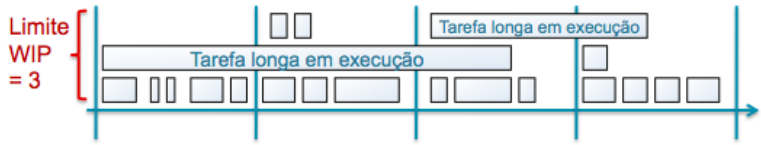
Uma equipe Scrum irá comprometer-se apenas aos itens que julguem poder terminar dentro de uma iteração (baseado no conceito de "*Done*"). Caso um item seja grande demais para caber em um *sprint*, a equipe e o *Product Owner* terão que encontrar formas de quebrá-lo em pedaços menores até que ele se encaixe.

Se os itens tendem a ser grandes, as iterações serão maiores (embora geralmente não maiores do que 4 semanas).



Equipes Kanban tentam minimizar o tempo de execução e equilibra o fluxo, de modo que indiretamente criam um incentivo para quebrar itens em pedaços relativamente menores. Porém não há nenhuma regra explícita indicando que os itens devam ser suficientemente pequenos para caber em um determinado período de tempo.

Em um mesmo quadro nós podemos ter um item que irá levar 1 mês para ser concluído e outro item que levará 1 dia.



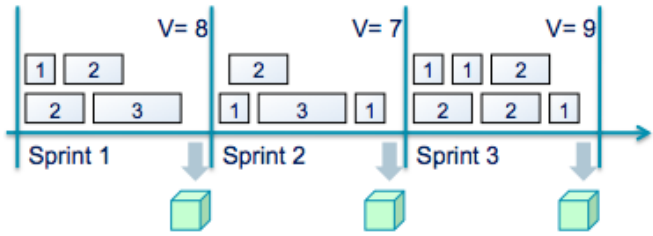
11

Scrum prescreve estimativas e velocidade

No Scrum, equipes devem estimar o tamanho (= quantidade de trabalho) de cada item aos quais eles se comprometeram.

Somando o tamanho de cada item concluído ao final de cada *sprint*, nós teremos a velocidade. A velocidade é uma medida da capacidade – a quantidade de coisas que nós podemos entregar por *sprint*.

Aqui temos um exemplo de uma equipe caso esta tenha uma velocidade média igual a 8.



Saber que a velocidade média é 8 é bom porque nos permite fazer previsões mais realistas sobre quais itens podemos entregar nos futuros *sprints*, e portanto, fazer planos de entrega cada vez mais próximos da realidade.

Já no Kanban, estimativa não é obrigatório. Portanto, se você precisar ter comprometimentos você precisará decidir como fornecer previsões.

Algumas equipes optam por fazer estimativas e medir a velocidade assim como no Scrum. Outras equipes escolhem pular as estimativas, mas tentam quebrar cada item em itens menores de aproximadamente do mesmo tamanho – desta forma eles podem medir a velocidade simplesmente em termos de quantos itens foram concluídos por unidade de tempo (por exemplo, requisitos ou funcionalidades por semana).

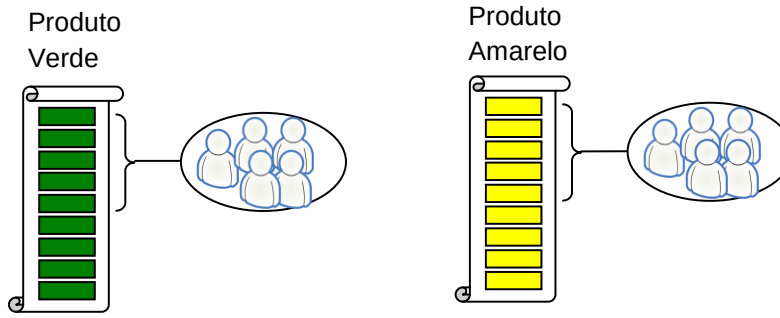
Algumas equipes agrupam itens em níveis mínimos de funcionalidades (do inglês *MMFs (minimum marketable features*) e medem a média de tempo de execução por MMF, e usam esta medição para estabelecer o ANS (Acordos de Nível de Serviço) – por exemplo, "quando nós nos comprometemos a um MMF, este será sempre entregue no prazo de 15 dias".

Existem inúmeras técnicas interessantes para o modelo Kanban de planejamento de entregas e gerenciamento de compromissos – porém não há técnica específica ou mandatória a ser seguida então vá em frente e pesquise novas formas e técnicas até encontrar a que se encaixe melhor ao seu contexto. Provavelmente veremos algumas "boas práticas" surgir ao longo do tempo.

12

Ambos permitem trabalhar em múltiplos produtos simultaneamente

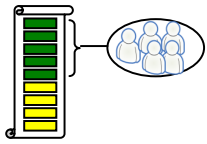
No Scrum, "*Product Backlog*" é um nome muito infeliz, pois implica que todos os itens devem ser do mesmo produto. Aqui temos dois produtos, verde e amarelo, cada um com seu próprio *product backlog* e sua própria equipe:



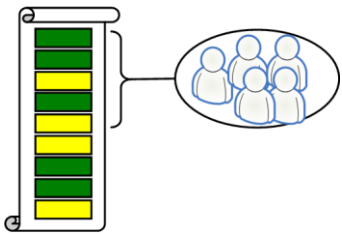
Mas, e se você só tem uma equipe? Bem, pense no *product backlog* mais como um *backlog* da equipe. Ele lista as prioridades das iterações programadas para uma equipe em particular (ou um grupo de equipes). Então, se essa equipe está mantendo múltiplos produtos, basta juntar os produtos em uma única lista. Isso nos força a priorizar os produtos, o

que é útil em alguns casos. Existem muitas maneiras de fazer isso na prática:

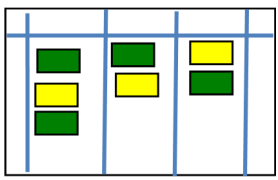
Uma estratégia seria ter a equipe focada em um produto por *sprint*:



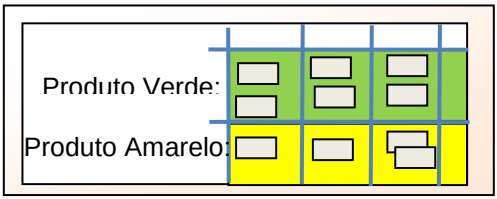
Outra estratégia seria ter a equipe trabalhando em características dos dois produtos em cada *sprint*:



É a mesma coisa no Kanban. Podemos ter diversos produtos fluindo através do mesmo quadro branco. Podemos distingui-los utilizando diferentes cartões coloridos:



... ou por raias:



13

Ambos são Lean e Ágeis

Eu não vou explorar o Pensamento Lean e o Manifesto Ágil aqui, mas em termos gerais tanto o Scrum quanto o Kanban são bem alinhados com estes valores e princípios. Por exemplo:

- Scrum e Kanban são sistemas puxados, que correspondem ao princípio de gestão de inventário de *Just In Time* (JIT) do Lean. Isto significa que a equipe escolhe quando e quanto de trabalho irá se comprometer para então "puxar" o trabalho quando estão prontos para começar, ao invés de ter que empurrar o trabalho de algum lugar. Assim como uma impressora puxa para a próxima página somente quando está pronta para imprimi-la (embora exista um número pequeno e limitado de papel que pode ser puxado).

- Scrum e Kanban são baseados em otimização empírica e continua de processo, que corresponde ao princípio de Kaizen do Lean.

- Scrum e Kanban dão ênfase à resposta a mudança ao invés de seguir um plano pré-estabelecido (embora o Kanban tipicamente permita uma resposta mais rápida do que o Scrum), um dos quatro valores do manifesto ágil.

... e mais.

Sob certa perspectiva, o Scrum pode ser visto como não tão enxuto porque ele prescreve itens em série em iterações de duração fixa. Mas isso depende do tamanho da sua iteração e com o que você está comparando. Comparado com um processo mais tradicional, no qual talvez integramos e entregamos alguma coisa 2 a 4 vezes por ano, um time Scrum produzindo código entregável a cada 2 semanas é extremamente enxuto.

Mas então, se você fizer a iteração cada vez mais curta, você está essencialmente aproximando ao Kanban. Quando você começa a falar sobre deixar a iteração mais curta que uma semana, você pode pensar em livrar-se inteiramente de iterações de duração fixa.

Eu disse isso antes e vou continuar dizendo: experimente até você encontrar algo que funcione para você! E depois continue experimentando :o)

14

Diferenças menores

Aqui estão algumas diferenças que parecem ser menos relevantes, em comparação com as outras mencionadas antes. Entretanto, é bom estar ciente delas.

Scrum prescreve um *product backlog* priorizado

Em Scrum, a priorização é sempre feita por triagem do *product backlog*, e as mudanças de prioridades realizam-se no próximo *sprint* (não no *sprint* atual). Em Kanban você pode escolher qualquer plano de prioridade (ou mesmo nenhum), e as mudanças realizam-se assim que a equipe estiver disponível (ao invés de em horários fixos). Pode-se ou não ter um *product backlog*, e pode-se ou não ser priorizado.

Na prática, isso faz pouca diferença. Em um quadro de Kanban a coluna mais à esquerda desempenha, tipicamente, o mesmo objetivo do *product backlog* de Scrum. Seja a lista ordenada ou não por prioridade, a equipe precisa de algum tipo de regra para a decisão de quais itens devem ser realizados primeiro. Exemplos de regras para a decisão:

- Sempre pegue o primeiro item do topo da lista.
- Sempre pegue o item mais antigo (então cada item tem um *timestamp*).
- Pegar qualquer item.

- Gastar aproximadamente 20% em itens de manutenção e 80% em novas funcionalidades.
- Dividir a capacidade da equipe aproximadamente de modo uniforme entre o produto A e o produto B.
- Sempre pegue os itens em vermelho primeiro, se houver algum.

Em Scrum, o *product backlog* também pode ser usado na forma de Kanban (eu acho). Podemos limitar o tamanho dele, e criar regras de decisão de como ele deve ser priorizado.

Em Scrum, são prescritas reuniões diárias

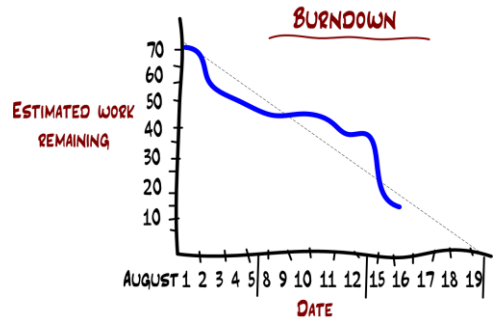
Uma equipe Scrum tem uma pequena reunião (de no máximo 15 minutos) todos os dias na mesma hora e local. O objetivo da reunião é difundir informação sobre o que está acontecendo, planejar o trabalho do dia em curso, e identificar os problemas mais importantes. Isso às vezes é chamado de reunião diária em pé, porque geralmente é realizada em pé (para mantê-la curta e manter um alto nível de energia).

Reuniões diárias não são prescritas no Kanban, mas, de qualquer maneira, a maioria das equipes de Kanban costuma fazê-las. É uma excelente técnica, independentemente do processo utilizado.

Em Scrum, a reunião diária é orientada à pessoa – cada pessoa passa a informação, um a um. Muitas equipes Kanban usam um formato mais orientado ao quadro, concentradas nos gargalos e outros problemas visíveis. Essa abordagem é mais escalável.

Se tivermos 4 equipes compartilhando o mesmo quadro e realizando suas reuniões diárias juntas, não necessariamente teremos que ouvir todo mundo falar enquanto nos concentramos nas partes de gargalo que estão no quadro.

No Scrum, são prescritos gráficos de *burndown*



Um gráfico de *burndown* do *sprint* mostra, em uma base diária, o quanto de trabalho resta na iteração atual.

A unidade do eixo Y é a mesma unidade utilizada nas tarefas de *sprint*. Normalmente horas ou dias (se a equipe divide itens de *backlog* em tarefas) ou pontos de história (se a equipe não os divide). Entretanto, há muitas variações disso.

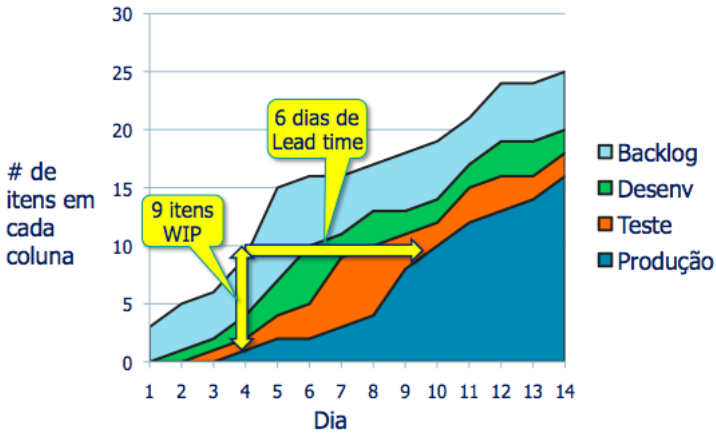
Em Scrum, gráficos de *burndown* do *sprint* são usados como uma das principais ferramentas para acompanhar o andamento de uma iteração.

Algumas equipes também usam gráficos de *burndown* de *release*, que segue o mesmo formato, mas em nível de *release* - ele normalmente mostra quantos pontos de história restam no *product backlog* depois de cada *sprint*.

O principal objetivo de um gráfico de *burndown* é perceber facilmente, o mais cedo possível, se estamos atrás ou à frente do cronograma, para que possamos adaptar.

Em Kanban, gráficos de *burndown* não são prescritos. Na verdade, não é prescrito nenhum gráfico em especial. Mas você está, logicamente, autorizado a usar qualquer tipo de gráfico que quiser (incluindo *burndowns*).

Este é um exemplo de um diagrama de Fluxo Cumulativo. Este tipo de gráfico mostra o quanto seu fluxo é regular e como as atividades em andamento afetam o seu *lead time*.



Funciona da seguinte maneira: Todos os dias, totalize o número de itens em cada coluna do quadro Kanban e coloque os valores no eixo Y. Assim, temos que no dia 4 havia 9 itens no quadro. Começando pela coluna mais à direita, havia 1 item em Produção, 1 item em Teste, 2 itens em Desenvolvimento e 5 itens no *backlog*. Se nós plotarmos estes pontos todos os dias e conectarmos os pontos teremos um bom gráfico como o do exemplo acima. As setas vertical e horizontal mostram a relação entre as atividades em andamento e o *lead time*.

A seta horizontal nos mostra que os itens adicionados ao *backlog* no dia 4 levaram, em média, 6 dias para alcançar a produção. Aproximadamente metade deste tempo foi em Teste. Podemos observar que limitando as atividades em andamento em Teste e em *backlog* poderíamos reduzir significativamente o tempo de espera total.

A inclinação da área azul-escuro nos mostra a velocidade (ou seja, número de itens implantados por dia). Ao longo do tempo, podemos ver como a maior velocidade reduz o tempo de espera, enquanto maior número de atividades em andamento aumenta o tempo de espera.

A maioria das organizações quer as coisas prontas mais rapidamente (reduzir o *lead time*). Infelizmente muitas caem

na armadilha de pensar que isso significa ter mais gente ou trabalhar horas-extras. Normalmente, a maneira mais eficaz de conseguir que as coisas sejam feitas mais rapidamente é liberar o fluxo e limitar o trabalho à capacidade, *não* adicionar mais gente ou trabalhar mais. Esse tipo de diagrama mostra porque e por isso aumenta a probabilidade de que a equipe e a gerência colaborem efetivamente.

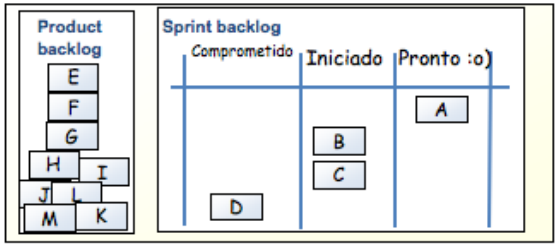
Isto se torna ainda mais claro se distinguirmos entre estados enfileirados (como "aguardando testes") e estados de trabalho (como "testando"). Queremos minimizar absolutamente o número de itens esperando em filas, e um diagrama de fluxo cumulativo ajuda a fornecer os incentivos corretos para isso.

15

Quadro Scrum vs. quadro Kanban – um exemplo menos trivial

Em Scrum, o *sprint backlog* é apenas uma parte de algo maior – a parte que mostra o que a equipe está fazendo durante o *sprint* atual. A outra parte é o *product backlog* – a lista de coisas que o *Product Owner* quer que sejam feitas nos *sprints* futuros.

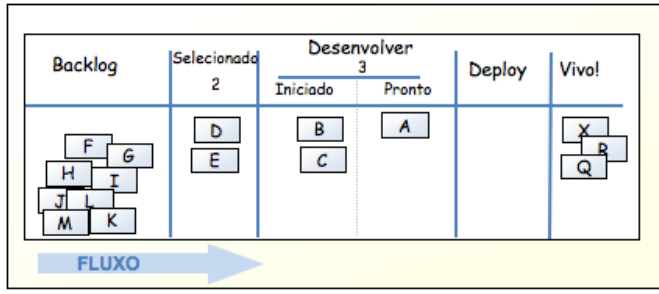
O *Product Owner* pode visualizar, mas não pode tocar no *sprint backlog*. Ele pode modificar o *product backlog* a qualquer momento, mas tais mudanças não terão efeito (i.e., não afetam o trabalho que está sendo feito no momento) até o próximo *sprint*.



Quando o *sprint* está pronto, a equipe "entrega código potencialmente entregável" para o *Product Owner*. Então a equipe termina o *sprint*, faz sua revisão e orgulhosamente demonstra as funcionalidades A, B, C, e D ao *Product* O*wner*. O *Product Owner* agora pode decidir se vai produzir.

A última parte – colocar realmente em produção – usualmente não é incluída no *sprint*, portanto, não é visível no *backlog* do *sprint*.

Neste cenário, o quadro Kanban deve preferencialmente se parecer com isto:



Agora todo o fluxo de trabalho está no mesmo quadro – não estamos mais simplesmente olhando para o que uma equipe Scrum está fazendo em uma iteração.

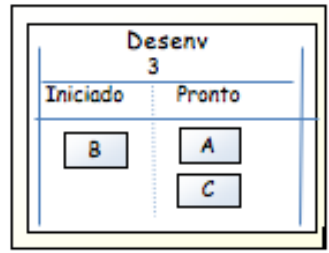
No exemplo acima, a coluna "*Backlog*" é, apenas, uma lista de coisas desejáveis, sem qualquer ordem especial. A coluna "Selecionados" contém os itens com maiores prioridades, com o limite Kanban igual a 2. Desta forma, deve conter apenas 2 itens com alta prioridade ao mesmo tempo. Sempre que a equipe estiver pronta para começar a trabalhar em um novo item, ela irá pegar o primeiro item da coluna "Selecionados". A qualquer momento o *Product Owner* poderá fazer mudanças nas colunas "*Backlog*" e "Selecionados", mas não poderá mudar as demais colunas.

A coluna "Desenvolver" (dividida em duas sub-colunas) mostra o que está sendo desenvolvido no momento, com um

limite Kanban de 3. Em termos de infra-estrutura de rede, o limite Kanban corresponde a "largura de banda" e o tempo de execução corresponde ao "*ping*" (ou tempo de resposta).

Por que nós dividimos a coluna "Desenvolver" em duas sub-colunas "Iniciado" e "*Pronto*"? Para dar à equipe de produção a oportunidade de saber quais itens eles podem mover para a produção.

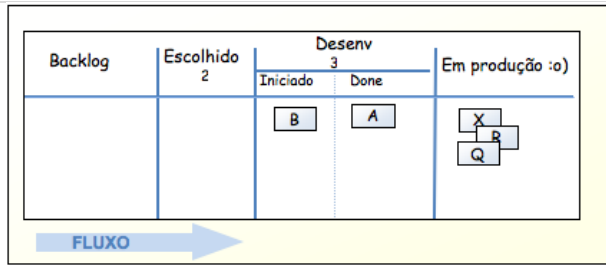
O limite de 3 é compartilhado pelas duas sub-colunas. Por quê? Vamos imaginar que temos dois itens em "*Pronto*":



Isto significa que só podemos ter 1 item em "Iniciado". Isto significa que teremos capacidade em excesso. Desenvolvedores que *poderiam* iniciar um novo item, mas não estão liberados em virtude do limite do Kanban. Isto lhes dá forte incentivo para concentrar esforços e ajudar a colocar as coisas em produção, limpar a coluna "*Pronto*" e maximizar o fluxo. Este efeito é bom e gradual – quanto mais coisas em "Pronto", menos coisas serão permitidas no "Iniciado" – o que ajuda a equipe a concentrar-se nas coisas certas.

Fluxo contínuo

Um fluxo contínuo é um tipo de cenário com um "fluxo perfeito", em que um item flui através do quadro sem ficar preso em nenhuma coluna. Isso quer dizer que a todo o momento existe alguém trabalhando naquele item. Aqui está um exemplo de como o quadro pode parecer nesse caso:

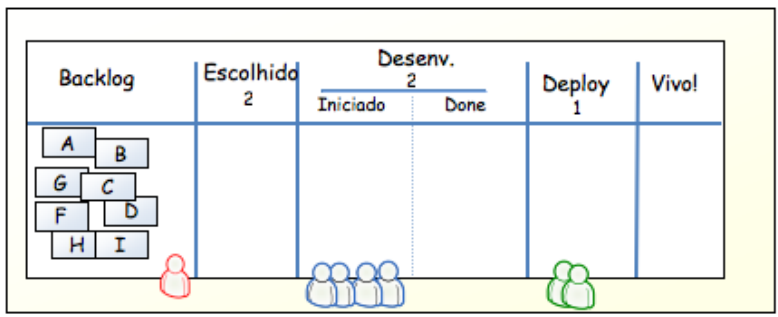


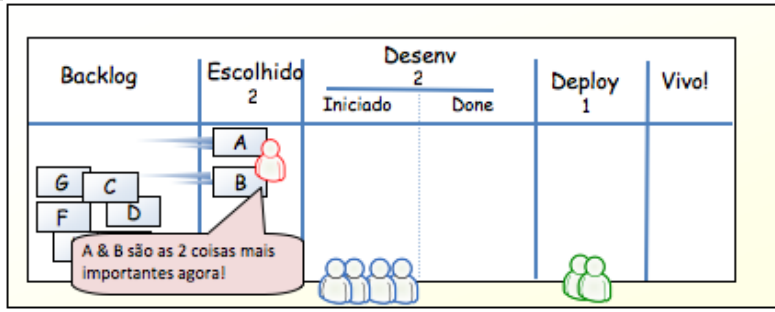
B está sendo desenvolvido nesse momento, A está sendo colocado em produção nesse momento. Sempre que a equipe estiver pronta para o próximo item, ela irá perguntar para o *Product Owner* qual é o item mais importante, e ela terá uma resposta na hora. Se este cenário ideal continuar, podemos nos livrar das duas colunas "*Backlog*" e "Escolhido" e assim ter um tempo de resposta realmente curto!

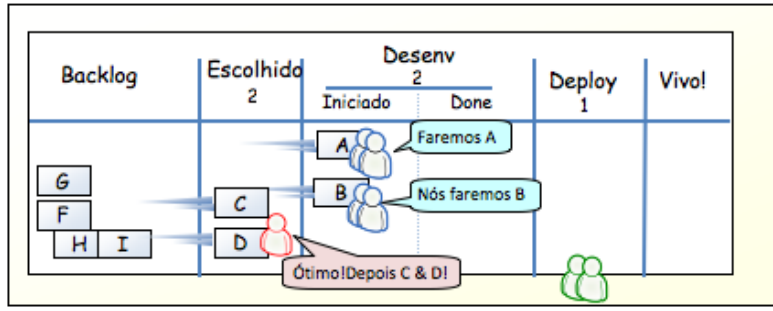
Cory Ladas expressou isso muito bem: "O processo ideal de planejamento do trabalho deve sempre fornecer à equipe de desenvolvimento a melhor coisa para trabalhar em seguida, nem mais e nem menos".

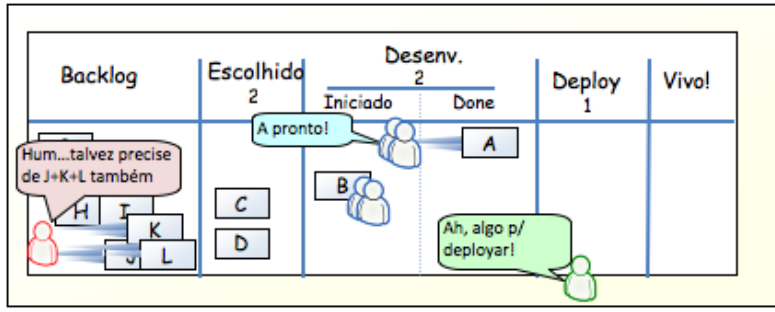
O limite das atividades em andamento existe para impedir que problemas fujam do controle. Então, se as coisas estão fluindo bem, os limites das atividades em andamento não são realmente usados.

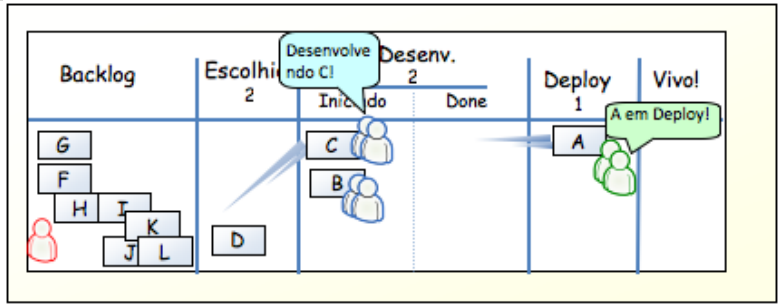
Um dia na terra do Kanban

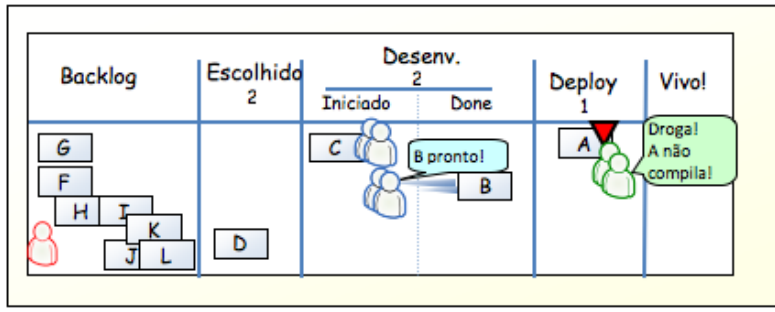


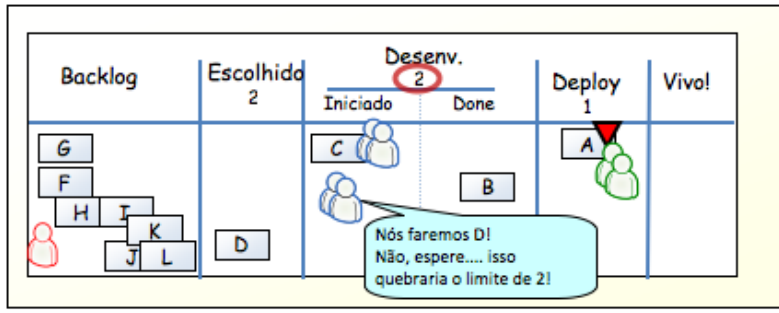

Backlog
Escolhido
2
Desenv
2
Iniciado
Done
Deploy
1
Vivo!
A
B
G
C
F
D
A & B são as 2 coisas mais importantes agora!

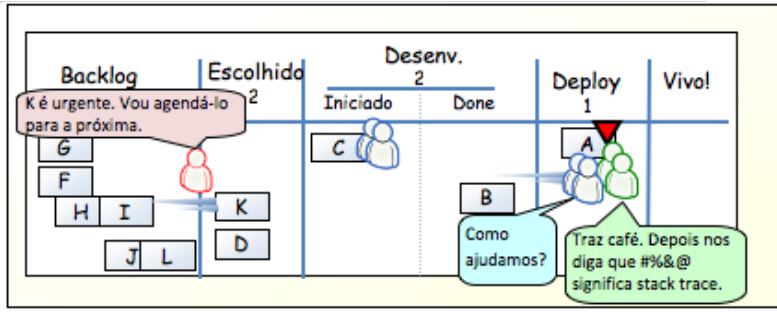

Backlog
Escolhido
2
Desenv
2
Iniciado
Done
Deploy
1
Vivo!
A
Faremos A
B
Nós faremos B
G
F
C
H
I
D
Ótimo!Depois C & D!

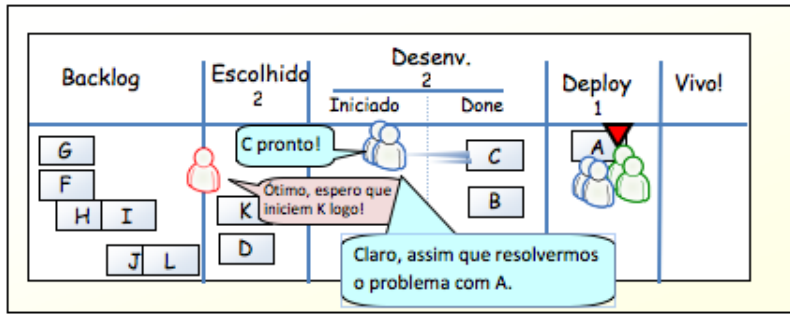

Backlog
Escolhido
2
Desenv.
2
Iniciado
Done
Deploy
1
Vivo!
A pronto!
A
Hum...talvez precise de J+K+L também
B
H
I
C
K
D
L
Ah, algo p/ deployar!

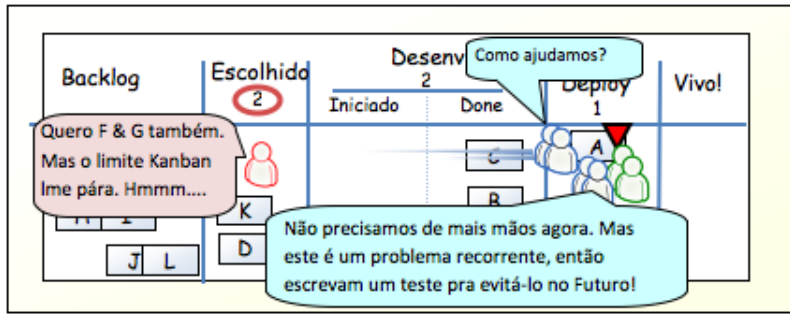

Backlog
Escolhi
2
Desenvolve
ndo C!
Desenv.
2
Ini do
Done
Deploy
1
Vivo!
A em Deploy!
G
F
H
I
K
J
L
D
C
B
A

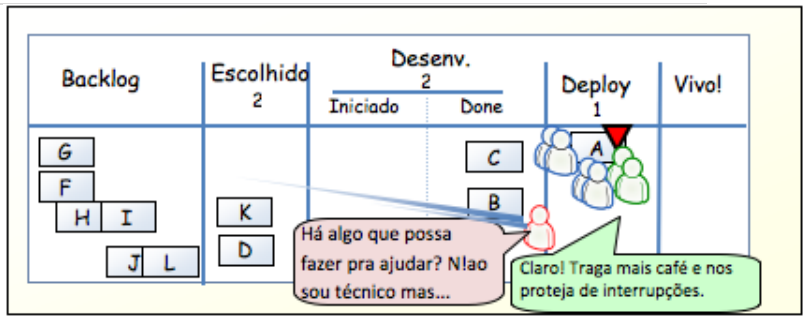

Backlog
Escolhido
2
Desenv.
2
Iniciado
Done
Deploy
1
Vivo!
B pronto!
Droga!
A não
compila!
G
F
H
I
K
J
L
D
C
B
A

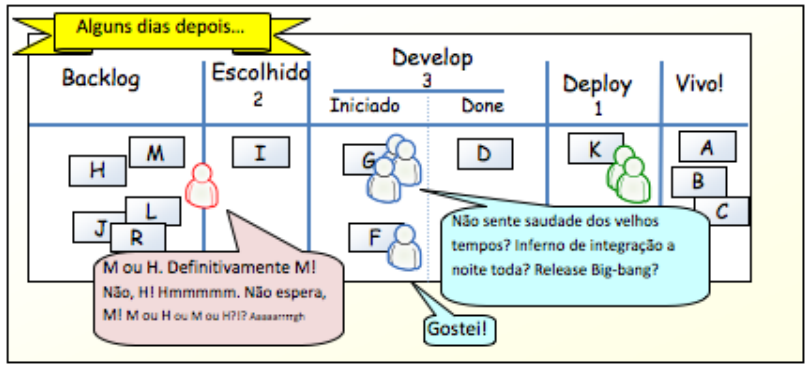

Backlog
Escolhido
2
Desenv.
2
Iniciado
Done
Deploy
1
Vivo!
Nós faremos D!
Não, espere.... isso
quebraria o limite de 2!
G
F
H
I
K
J
L
D
C
B
A


Backlog
Escolhido
2
Desenv.
2
Iniciado
Done
Deploy
1
Vivo!
K é urgente. Vou agendá-lo para a próxima.
G
F
H
I
J
L
K
D
C
A
B
Como ajudamos?
Traz café. Depois nos diga que #%&@ significa stack trace.


Backlog
Escolhido
2
Desenv.
2
Iniciado
Done
Deploy
1
Vivo!
G
F
H
I
J
L
C pronto!
Ótimo, espero que iniciem K logo!
K
D
C
B
A
Claro, assim que resolvermos o problema com A.


Backlog
Escolhido
2
Desenv
2
Iniciado
Done
Como ajudamos?
Deploy
1
Vivo!
Quero F & G também. Mas o limite Kanban me pára. Hmmm....
K
D
J
L
C
B
A
Não precisamos de mais mãos agora. Mas este é um problema recorrente, então escrevam um teste pra evitá-lo no Futuro!





O quadro de Kanban precisa se parecer com este?

Não, o quadro acima foi apenas um exemplo!

A única coisa que o Kanban prescreve é que o fluxo de trabalho deve ser visual, e que o WIP deve ser limitado.

O objetivo é criar um bom fluxo através do sistema e minimizar o *lead time*. Então você precisa regularmente levantar questões como:

Que colunas devemos ter?

Cada coluna representa um estado de fluxo de trabalho, ou uma pilha (fila) entre dois estados de fluxo de trabalho. Comece simples e adicione colunas conforme o necessário.

Quais devem ser os limites do Kanban?

Quando o limite de Kanban para as "suas" colunas foi atingido e você não tem nada para fazer, comece a procurar por um gargalo (pontos empilhando acima, à direita no quadro) e ajudar a corrigir o gargalo.

Se não houver nenhum gargalo, é uma indicação de que o limite do Kanban pode estar muito baixo, uma vez que a razão para ter o limite era reduzir o risco de se alimentar gargalos.

Se você notar que muitos itens continuam por um longo tempo sem serem trabalhados, é uma indicação que o limite de Kanban pode estar muito alto.

- Limite de Kanban muito baixo => pessoas ociosas => má produtividade.

- Limite de Kanban muito alto => tarefas inativas => *lead time* ruim.

Quão estritos são os limites de Kanban?

Algumas equipes as tratam como regras estritas (isto é, a equipe não pode exceder um limite), algumas equipes as tratam como diretrizes ou disparadores de discussão (isto é, quebrar um limite de kanban é permitido, mas deve ser uma decisão intencional com uma razão concreta). Então, mais uma vez, cabe a você. Eu lhe disse que o Kanban não era muito prescritivo, certo?

16

Resumo de Scrum vs. Kanban

Similaridades

- Ambos são Lean e *Agile*.
- Ambos usam controle de cronograma.
- Ambos limitam atividades em andamento.
- Ambos usam transparência para direcionar a melhoria do processo.
- Ambos concentram-se na entrega de software que funcione, o mais rápido possível e freqüentemente.
- Ambos são baseados em equipes auto-organizáveis.
- Ambos exigem que o trabalho seja dividido em partes.
- Em ambos, o planejamento de *release* é continuamente otimizado, baseado em dados (velocidade / tempo de execução) empíricos.

- **Diferenças**

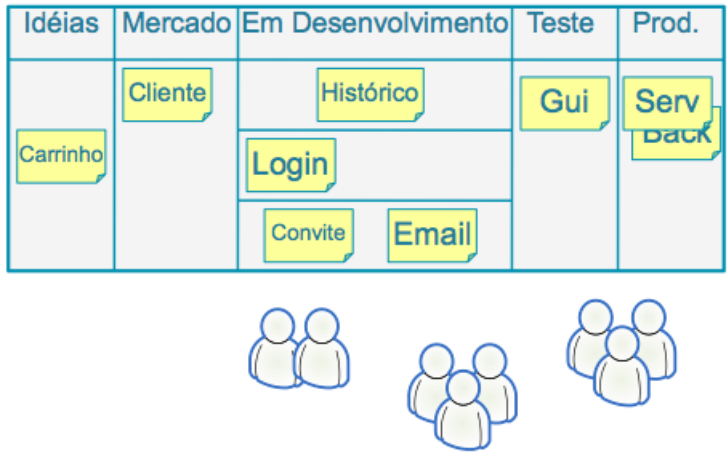
Scrum	**Kanban**
Iterações Timeboxed prescritas.	**Iterações Timeboxed opcionais.** Pode ter cadência separada para o planejamento, release e melhoria de processos. Pode ser orientada para eventos, em vez de timeboxed.
Equipe compromete-se a uma quantidade específica de trabalho para esta iteração.	**Compromisso opcional.**
Usa a **velocidade** como padrão métrico para o planejamento e melhoria de processos.	Usa o ***Lead time*** como padrão métrico para o planejamento e melhoria de processos.
Equipes multifuncionais prescritas.	Equipes multifuncionais opcionais. **Equipes de Especialistas autorizada**.
Os itens devem ser divididos para que possam ser concluídos dentro de 1 *sprint*	Nenhum tamanho especial de item é prescrito
Gráfico Burndown	Nenhum tipo específico de diagrama é prescrito
WIP limitado indiretamente (por sprint)	**WIP limitado diretamente** (por situação do fluxo de trabalho)
Estimativa prescrita	**Estimativa opcional**
Não poderá adicionar itens à iteração em uso	**Pode adicionar novos itens sempre que houver capacidade disponível**
O ***sprint backlog* é de uma equipe especifica**	**Quadros kanban podem ser compartilhados por várias**

	equipes ou individualmente
Possui 3 papéis (*Product Owner/ScrumMaster/Team*)	**Não discrimina nenhum papel**
Um quadro Scrum é reiniciado entre cada *sprint*	Um **quadro kanban continua**
Prescreve um *product backlog* priorizado	**Priorização é opcional.**

- Então é isso. Agora você sabe as diferenças.
- Mas ainda não acabou, agora é a melhor parte! Prepare-se, é hora de "colocar a mão na massa" com Mattias e ver como é isso na prática!

Parte II – Estudo de caso

Kanban na vida real



Esta é a história de como nós aprendemos a melhorar usando Kanban. Quando começamos, não havia muita informação disponível e o Dr. Google naquela época nos deixou de mãos vazias. Hoje, Kanban está evoluindo com sucesso e há um emergente campo de conhecimento. Eu claramente recomendo que dêem uma olhada no trabalho de David Anderson, por exemplo, em 'classes de serviço'. Então, esta é a primeira (e última) concessão (eu prometo!). Independentemente da solução que você implementar, certifique-se de que ela resolve seus problemas específicos. E pronto. Vamos chegar lá. Então, essa é nossa história.

Mattias Skarin

17

A natureza das operações técnicas

Se você nunca ficou disponível 24/7, você tem apenas uma leve idéia do sentido de responsabilidade ao gerenciar um ambiente de produção.

Esperam que você resolva a situação no meio da noite, independentemente de você ser ou não a origem do problema. Ninguém sabe, é por isso que eles te ligam. É um grande desafio porque você não criou o hardware, os *drivers*, o S.O ou o software customizado.

Suas opções muitas vezes limitam-se a reduzir o problema, reduzindo o impacto e salvando as evidências necessárias para recriar o problema, e esperar que a pessoa responsável por causar aquilo consiga reproduzir e resolver o que você acabou de testemunhar.

Para as operações técnicas, a capacidade de resposta e resolução de problemas com rapidez e precisão são fundamentais.

18

Por que diabos mudar?

Em 2008, um de nossos clientes, uma organização de desenvolvimento de jogos escandinava, trabalhou em uma série de melhorias de processo. Uma delas incluiu escalar Scrum para a organização de desenvolvimento e, gradualmente, remover os impedimentos que não permitiam que as equipes de desenvolvimento entregassem software.

À medida que o software começou a fluir e o desempenho melhorou, esta pressão aumentada voltou-se para as equipes técnicas de operações.

Antes, as equipes de operações estavam normalmente à margem; agora, elas estavam se tornando cada vez mais envolvidas, como parte ativa do processo de desenvolvimento.

Figura 1. A organização das operações técnicas incluiu três equipes, os engenheiros de bancos de dados (DBAs), os administradores de sistemas e o suporte de segundo nível.

Portanto, ajudar a equipe de desenvolvimento não foi suficiente.

Se nós tivéssemos nos concentrado apenas nas equipes de desenvolvimento isso teria causado atrasos em importantes melhorias de infra-estrutura que eram executadas pelas equipes de operações técnicas. Melhorias nas duas áreas eram necessárias.

Além disso, o progresso nas equipes de desenvolvimento significou que os gerentes fossem cada vez mais requisitados a ajudar com análises e sugestões sobre as idéias. Significou que eles tiveram menos tempo para a priorização das tarefas em tempo real e para solução de problemas.

A equipe de gerenciamento percebeu que eles precisavam agir antes que a situação ficasse fora de controle.

19

Por onde começamos?

Um bom ponto de partida foi perguntar às equipes de desenvolvimento, quem eram os clientes de operações técnicas.

Opinião dos Desenvolvedores sobre Operações

Perguntei "Quais são as três coisas mais importantes que vem na sua cabeça quando você pensa em 'Operações'"? As repostas mais comuns foram:

"Conhecimento Variável"

"O sistema de workflow *deles é péssimo"*

"Muito competentes quando o assunto é infra-estrutura"

"O que os caras estão fazendo?"

"Eles querem ajudar, mas, na verdade, conseguir ajuda é difícil"

"São necessários muitos e-mails para coisas simples"

"Projetos demoram muito"

"Dificuldade para entrar em contato"

Em resumo, essa era a opinião dos desenvolvedores sobre operações. Agora, vamos comparar isso com a opinião de operações sobre desenvolvimento.

Opinião de Operações sobre Desenvolvimento

"Porque você não está usando as vantagens da plataforma?"

"Vamos fazer dos releases um assunto mais leve!"

"A sua péssima qualidade nos prejudica!"

"Eles têm que mudar" – foi um tema comum na argumentação de ambos os lados. Obviamente o espírito precisaria mudar se quiséssemos nos mexer para solucionar problemas comuns. Do lado positivo; "*Muito competentes quando o assunto é infra-estrutura*" (indica confiança no núcleo da confiança) me fez acreditar que a mentalidade "nós VS eles" podia ser resolvida se criássemos as condições de trabalho certas. Eliminar excesso de trabalho e concentrar-nos em qualidade foi uma opção viável.

20

Começando

Então nós precisávamos começar, mas por onde? A única coisa que sabíamos ao certo era: onde quer que comecemos, não será onde iremos terminar.

Meu conhecimento é o de um desenvolvedor então eu certamente sabia pouco sobre a natureza das operações. Eu não estava a ponto de "explodir e começar a mudar tudo". Eu precisava de uma abordagem menos confrontadora que, entretanto, pudesse nos ensinar coisas relevantes, descartar as irrelevantes, e que fosse fácil de aprender.

Os candidatos eram:

1. Scrum – que vinha funcionando bem com as equipes de desenvolvimento.

2. Kanban – novo & não testado, mas com boa adaptabilidade aos princípios Lean que nos faltavam.

Nas discussões com os gerentes, Kanban e os princípios de Lean pareciam se encaixar nos problemas que estávamos tentando resolver. Do ponto de vista deles, *sprints* não se encaixariam muito bem porque estavam tendo que fazer novas priorizações diariamente. Então, Kanban foi o ponto de partida mais lógico, mesmo sendo algo completamente novo para nós.

21

Iniciando os times

Como iniciamos os times? Não existia manual sobre como fazer isto acontecer.

Fazer de forma errada era arriscado demais. Fora perder as oportunidades de melhoria, nós éramos nós lidando com uma plataforma de produção com pessoal altamente especializado e habilidoso; difíceis de serem substituídas. Deixar eles alienados era uma idéia ruim.

- Nós deveríamos iniciar? E aceitar as conseqüências conforme elas aparecem?

- Ou deveríamos executar um treinamento primeiro?

Era óbvio para nós – deveríamos realizar o treinamento certo? Mas como? Era um desafio conseguir todo o time técnico de operação participando do treinamento (quem iria atender se alguém ligasse?).

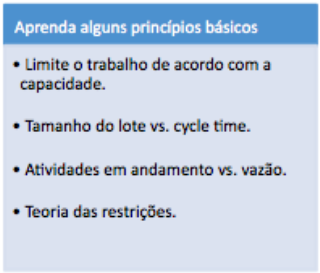
No final decidimos fazer um treinamento com meio dia de duração, mantendo simples e baseado em exercícios.

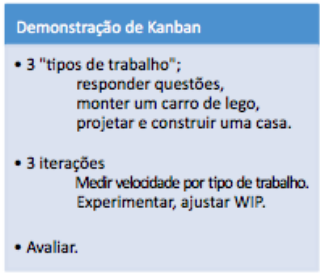
O workshop

Um dos benefícios do workshop foi que ajudou a trazer à tona nossos problemas cedo. Também proporcionou um ambiente de alta confiança onde as complicações puderam ser discutidas diretamente com os membros do time.

Quero dizer, vamos enfrentar isso – nem todos estavam muito interessados em mudar a atual forma de trabalhar. Mas a maioria dos membros do time estava aberta a tentar.

Então executamos a oficina demonstrando os princípios mais importantes e fizemos uma simulação de Kanban em escala reduzida.





No final do workshop fizemos uma votação para verificar se os times estavam dispostos a tentar isso de verdade. Sem objeções neste ponto, pudemos seguir em frente.

22

Lidando com partes interessadas

Era de se supor que as partes interessadas também seriam igualmente afetadas por uma implementação Kanban. Entretanto as mudanças devem ser para melhor – isso significa que a equipe deve começar a dizer "não" a trabalhos que não tem como concluir, a atentar para a qualidade e remover todas as coisas não prioritárias do *backlog* da equipe. Apesar disso, ter uma discussão antes é sempre uma boa idéia.

As partes interessadas mais próximas são o pessoal do suporte de primeiro nível e os gerentes de departamento. Uma vez que eles estejam participando de todo o processo, eles já estão dando seu aval para o desenvolvimento do produto. Mesmo para as equipes de desenvolvimento (que já estavam mesmo esperando melhorias de qualquer maneira). Mas, para uma equipe em particular – a equipe de suporte – as coisas são diferentes. O problema mais significativo era que eles estavam sobrecarregados de trabalho. Além disso, eles lidam diretamente com demandas de problemas do cliente e a empresa tem o compromisso de responder a todas as demandas. Agora isto estará prestes a mudar se implementarmos Kanban e começarmos a impor limites para a quantidade de trabalho em andamento.

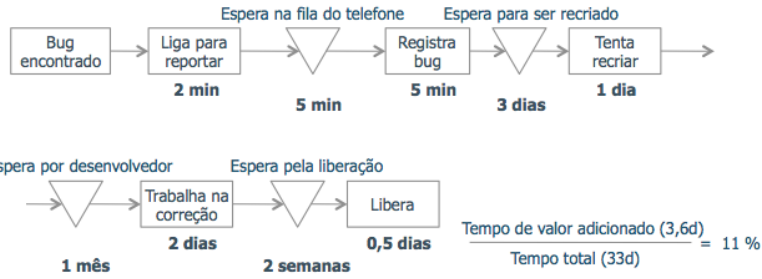
Então, fizemos uma jornada com as principais partes interessadas e lhes apresentamos as nossas intenções, os benefícios esperados e as possíveis conseqüências. Para meu alívio, nossas idéias em geral foram bem recebidas, algumas vezes com uma observação do tipo "seria ótimo se finalmente pudéssemos dar um tempo com esse monte de demandas".

23

Construindo o primeiro quadro

Uma boa forma de iniciar a construção de um quadro Kanban é fazendo o mapa de fluxo de valor.

É basicamente uma visualização da cadeia de valor e proporciona compreensão sobre os estágios de trabalho, fluxo e tempo através do sistema (tempo de ciclo).



Mas iniciamos de modo muito mais simples; um quadro Kanban desenhado e papel junto com o gerente. Revisamos umas duas vezes e então seguimos em frente.

As questões que surgiram nessa fase foram:

- Quais tipos de trabalho têm aqui?
- Quem cuida disso?
- Devemos dividir as responsabilidades conforme os diferentes tipos de trabalho?
- Como lidamos com responsabilidades compartilhadas tendo habilidades específicas?

Já que os diferentes tipos de trabalho tinham diferentes acordos de nível de serviço, foi natural deixar que cada equipe controlasse a elaboração de seu quadro. Eles mesmos fizeram as colunas e as linhas.

A próxima grande decisão foi se usaríamos responsabilidade compartilhada conforme os diferentes tipos de trabalho. "Deveríamos deixar uma parte fixa de a equipe lidar com questões diretas (trabalho reativo) e deixar o resto da equipe concentrada nos projetos (trabalho proativo)" Decidimos tentar, inicialmente, a responsabilidade compartilhada.

A razão principal foi identificarmos que auto-organização, aprendizagem constante e transferência de conhecimento pelos membros da equipe seriam essenciais para manter o crescimento.

A desvantagem dessa decisão seria potenciais problemas para todos, mas esta foi à melhor solução que pudemos encontrar para começar.

Uma pequena nota: quando fizemos o workshop, as equipes de fato se auto-organizaram a partir deste problema. Eles deixaram uma pessoa lidar com as requisições imediatas e o resto tratou das questões maiores.

O primeiro modelo Kanban

Abaixo está o modelo básico que usamos para Kanban. Note que a equipe decidiu deixar os itens em fluxo ascendente (como bolhas na água) em vez do modelo mais típico, de fluxo da esquerda para a direita.

Figura 2. Este é o primeiro modelo para Kanban. Prioridades seguem da esquerda para a direita e o fluxo de baixo para cima. Atividades em andamento, contadas como número total de tarefas na linha atividades em andamento (circulada em vermelho). Modelo influenciado por experiências relatadas por Linda Cook.

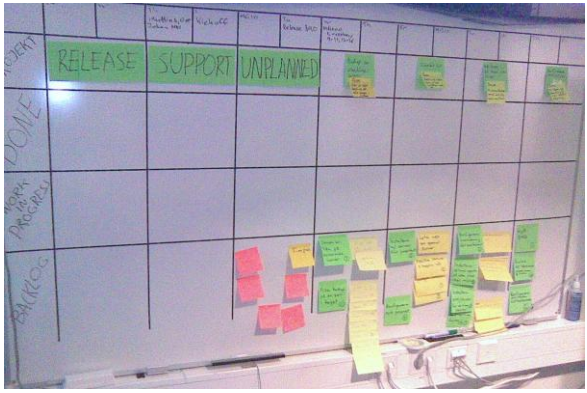


Figura 3. Primeiro quadro Kanban para administração de equipe.

- Linhas usadas

Estado de *workflow* (linha)	**Como definimos**
Backlog	Histórias que o gerente decide como necessárias.
Atividades prontas para serem iniciadas	Histórias são estimadas e divididas em tarefas de até, no máximo, 8 horas.
Atividades em andamento	A linha que contém a quantidade de tarefas em andamento. Começamos com um limite de 2x o tamanho da equipe -1 (um a menos para colaboração). Então, para uma equipe de 4 pessoas, a quantidade de tarefas em andamento seria de 7.
Done	Executável pelo usuário.

- Colunas utilizadas

Tipo de trabalho	**Como definimos**
Lançamento de versão	Ajudar as equipes de desenvolvimento a fazer lançamentos de versões do software
Suporte	Requisições menores vindas de outras equipes.
Não planejado	Trabalho não antecipado que precisa ser feito, mas que não tem um responsável definido. Por exemplo, melhorias tópicas em infra-estrutura.
Projeto A	Projeto de grande enfoque tecnológico, por exemplo, mudanças de hardware em um ambiente de teste.
Projeto B	Outro grande projeto.

Nem todos os quadros Kanban se parecem. Tudo começa com um desenho simples que evolui ao longo do tempo.

24

Definindo o primeiro limite de WIP

Nossa primeira definição para o limite de atividades em andamento (WIP) foi bem generoso. Decidimos isso ao visualizar o fluxo que teríamos e fomos experimentando a partir dali. Era pouco provável que tivéssemos como prever o limite ideal desde o início. Com o passar do tempo, iríamos ajustando os limites da quantidade de atividades em andamento (WIP) toda vez que encontrássemos uma boa razão para isso (tudo o que precisávamos fazer era indicar o valor no quadro).

A primeira quantidade de atividades em andamento (WIP) que utilizamos foi 2n-1 (n = número de membros na equipe, -1 para encorajar a cooperação). Por quê? Simplesmente porque não tínhamos uma idéia melhor ☺. Também não havia por que não começar desta forma. A fórmula oferecia uma explicação simples e lógica a todos que tentavam empurrar uma tarefa para a equipe: "... então, dado que cada membro da equipe só podia trabalhar em, no máximo, duas tarefas ao mesmo tempo, uma ativa e outra aguardando, como poderíamos esperar que pegassem *mais* tarefas *ainda*?" Agora, olhando para trás, qualquer limite generoso teria funcionado para os iniciantes. Ao monitorar o quadro Kanban fica fácil descobrir os limites adequados ao longo do tempo.

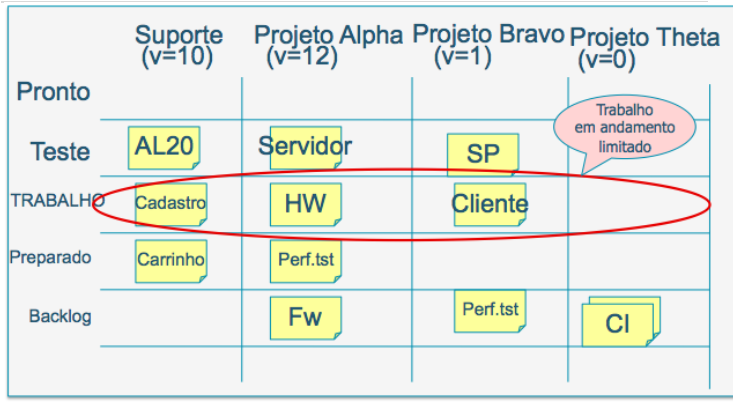


Figura 4. Como aplicamos o limite da quantidade de atividades em andamento para as equipes de DBA e administração de sistemas, um limite conforme o tipo de trabalho.

Uma observação que fizemos foi que era inútil definir o limite da quantidade de atividades em andamento em pontos da história.

Era muito difícil de acompanhar o andamento das atividades. O único limite suficientemente fácil de acompanhar era simplesmente contar a quantidade de itens (= tarefas simultâneas).

Para a equipe de suporte utilizamos atividades em andamento (WIP) definidas por coluna. Isto porque nós precisávamos reagir mais rápido caso o limite estivesse sendo excedido.

25

Honrando o limite de WIP

Enquanto o respeito ao limite de WIP parece fácil na teoria, é um acordo difícil de honrar na prática. Significa dizer "não" em algum momento. Tentamos várias abordagens para lidar com isso.

Discussão no quadro

Se um descumprimento do acordo fosse descoberto, deveríamos trazer a parte interessada até o quadro e perguntar o que ele quis obter. A princípio, a razão mais freqüente para o descumprimento era simplesmente a inexperiência. Em alguns casos encontramos diferentes visões na priorização. Um caso típico seria o membro de uma equipe especialista trabalhando em uma área específica. Essas foram as únicas vezes em que tivemos atrito. Na maioria das vezes os problemas foram resolvidos imediatamente conversando em frente ao quadro.

Designando uma seção de transbordo

Se dizer "não" fosse muito conflitante, e remover os itens fosse muito difícil, nós movíamos os itens de baixa prioridade para uma seção de transbordo[4] no quadro quando os limites

[4] N.T.: *Overflow*

de WIP eram ultrapassados. Duas regras são aplicadas às tarefas da seção de transbordo:

1. Elas não foram esquecidas, quando tivermos tempo iremos lidar com elas.

2. Se nós as cortarmos, você será informado.

Depois de apenas duas semanas ficou óbvio que os itens excedentes não seriam tratados. Então, com o apoio do gerente da equipe, eles puderam finalmente ser removidos.

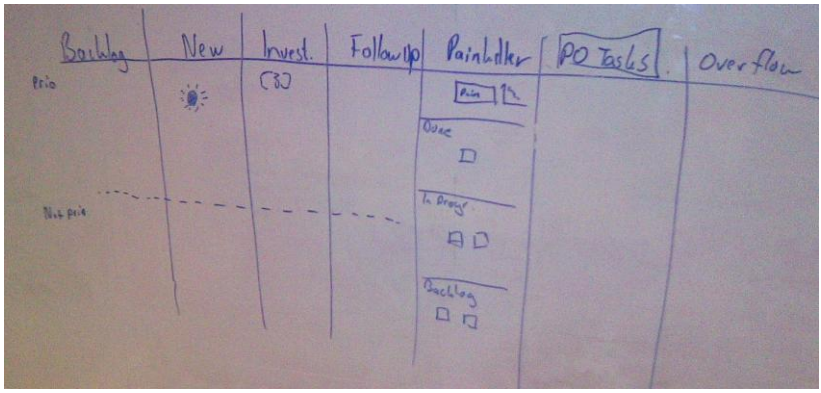


Figura 5. Um esboço do quadro do kanban da equipe de suporte, com a seção de transbordo bem à direita.

26

Quais tarefas vão para o quadro?

Havíamos decidido não colocar no quadro *todo* o trabalho feito pela equipe. Monitorar coisas como uma ligação telefônica ou pegar café deixaria o quadro Kanban um monstro administrativo. Nós estávamos aqui para *resolver* problemas, não criar problemas ☺. Então decidimos colocar no quadro somente tarefas com tamanho > 1 hora; qualquer coisa menor seria considerada "ruído branco". O limite de 1h realmente funcionou muito bem e foi uma das poucas coisas que não foram modificadas. (Tivemos que revisar a premissa sobre o impacto que o ruído de fundo tinha, mas mais posteriormente).

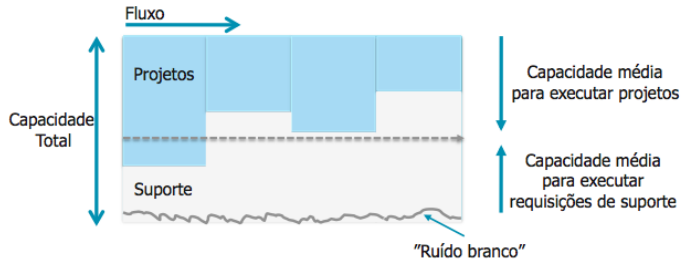


Figura 6. Nós começamos assumindo que a capacidade total era consumida principalmente por dois tipos de trabalho: maiores (projetos) e menores (suporte). Acompanhar a velocidade nos projetos poderia nos dar uma indicação da data de entrega, caso isso fosse necessário. "Ruído branco" (suporte pequeno < 1h, reuniões, pegar café, auxiliar os colegas) sempre era esperado.

27

Como estimar?

Este é um tópico em discussão; e certamente há mais de uma resposta:

- Estimar regularmente.
- Estimar quando necessário.
- Usar estimativas em dias / pontos da história.
- Estimativas são incertas. Use os tamanhos de camiseta (pequena, média, grande)
- Não faça estimativas ou as faça só quando existir um custo, devido a atrasos, que justifique isso.

Ligeiramente influenciados pelos Scrum (já que foi onde tudo começou, afinal) nós decidimos começar com pontos da História. Mas, na prática, as equipes tratavam os pontos da história de forma equivalente a homem-hora (parecia mais natural para eles). Inicialmente todas as histórias eram estimadas. Com o passar do tempo, os gerentes aprenderam que, se mantivessem um número baixo de projetos, simultaneamente, eles não deixariam as partes interessadas esperando. Aprenderam também que, no caso de uma mudança repentina, eles poderiam priorizá-la novamente e solucionar o problema.

A necessidade de projetar uma data de entrega já não era mais um grande problema. Isto levou os gerentes a pararem de

pedir estimativas antecipadas. Eles só as pediam se temessem que manteriam as pessoas esperando.

Há algum tempo atrás, um gerente, estressado com um telefonema, prometeu entregar o projeto "até o final desta semana". Sendo este um projeto com quadro Kanban, foi fácil estimar o andamento do trabalho (contando as histórias prontas) e concluir que ele só estaria cerca de 25% concluído após uma semana. Desta forma, mais três semanas seriam necessárias. Confrontado com os fatos, o gerente mudou a prioridade do projeto, interrompeu os trabalhos simultâneos e tornou a entrega possível. Sempre verifique o quadro ☺

O que o tamanho estimado significa? *Lead time* ou tempo de trabalho?

Nossos pontos da história refletem tempo de trabalho, i.e., quantas horas de esforço ininterrupto esperamos que esta história leve; e não tempo de planejamento (*lead time* - ou tempo de cronograma, ou quantas horas de espera). Ao medir o número de pontos por história que ficam "prontos" a cada semana, (velocidade) pudemos deduzir o tempo gasto com planejamento (*lead time*).

Nós estimamos cada nova história apenas uma vez, não revisamos as estimativas de histórias durante sua execução. Isto nos permitiu minimizar o tempo da equipe gasto com estimativas.

28

Então, como é que trabalhamos, realmente?

Na realidade, o Kanban é muito livre. Você pode trabalhar de diversas maneiras. Pode deixar a equipe trabalhar de acordo com as atividades agendadas, ou pode escolher trabalhar em atividades quando houver a motivação necessária que as justifique.

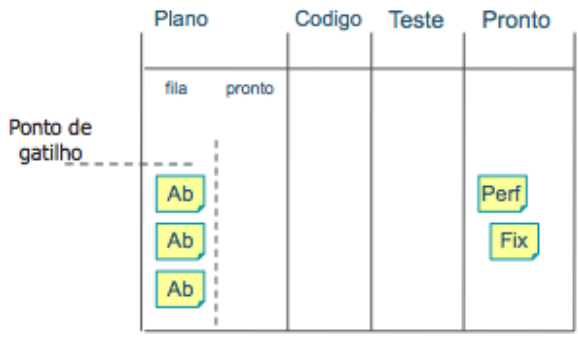


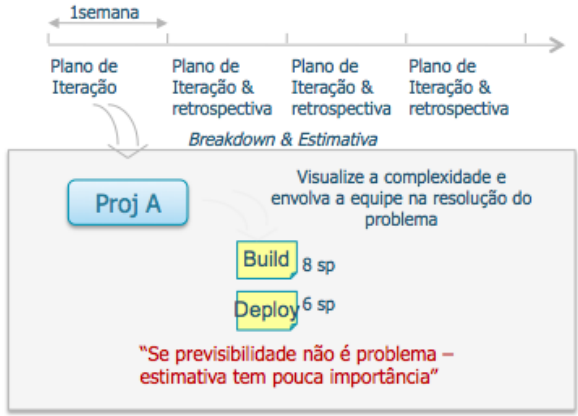
Figura 7. Quando três tarefas chegam ao *backlog*, isso provoca um evento de planejamento/estimativa.

Nós optamos por agendar dois eventos recorrentes:

- Reunião diária em pé – com a equipe em frente ao quadro, para trazer problemas à tona e

ajudar a criar a "visão em partes" das tarefas de outros membros da equipe.

- Planejamento de iteração semanal, para fins de planejamento e melhoria contínua.



Isso funcionou bem para nós.

Reunião diária em pé

A reunião diária em pé era similar a *Daily Scrum*. Acontecia depois da reunião de "Scrum de Scrums", com participação de todas as equipes (Desenvolvimento, Testes, Operação). A reunião de Scrum de Scrums trazia informações importantes para as equipes Kanban como, por exemplo, que problemas deveriam ser encaminhados primeiro, ou qual equipe de desenvolvimento estava sofrendo mais no momento. Inicialmente, os gerentes participavam com freqüência destas reuniões, propondo soluções e priorizando decisões. Ao longo do tempo, conforme as equipes foram melhorando a auto-organização, os gerentes passaram a freqüentá-las menos (mas não estavam longe, caso a equipe precisasse).

Planejamento da iteração

Uma vez por semana, nós organizamos um planejamento de iteração. Nós o mantivemos, semanalmente, em um horário fixo, porque descobrimos que, se não fosse planejado, o horário seria consumido por outras prioridades ☺. E nós precisávamos de mais conversas entre as equipes. Uma agenda típica era:

- Atualizar quadro e gráficos. (Projetos Prontos eram movidos para um "Quadro dos Prontos".)
- Olhe para a semana passada. O que aconteceu? Porque aconteceu assim? O que pode ser feito para melhorar?
- Reajustes do limite de atividades em andamento (WIP), se necessário.
- Quebra de tarefas e estimativas de novos projetos [se houver necessidade para isto].

Basicamente, o planejamento da iteração foi uma combinação de estimativa e impulso por melhoria contínua. Problemas qualificados como pequenos e médios foram resolvidos na hora, com o suporte dos gerentes de linha. Mas, mantendo o foco nos mais complexos, questões de infra-estrutura foram um martírio. Para lidar com isso nós habilitamos as equipes a apontarem até 2 "impedimentos da equipe" aos seus gerentes.

As regras eram:

1. Os gerentes podem trabalhar em até 2 *slots* ao mesmo tempo.
2. Caso ambos estejam cheios, você pode adicionar um novo, desde que

remova o menos importante.

3. As equipes decidem quando o problema está resolvido.

Essa mudança foi positiva. Rapidamente as equipes puderam ver que os gerentes estavam trabalhando para ajudá-las, até mesmo nas questões mais difíceis. Elas podiam apontar para os impedimentos e perguntar "qual o andamento deste assunto?" Eles não seriam esquecidos ou anulados por uma nova estratégia de alta prioridade.

Um exemplo de impedimento grave era que os operadores não tinham o suporte necessário que precisavam dos desenvolvedores quando eles suspeitavam de uma falha (*bug*). Eles precisavam de ajuda para descobrir qual parte do sistema havia causado o problema, mas como os desenvolvedores estavam ocupados desenvolvendo coisas novas do *sprint*, os problemas continuavam a se acumular. Não causou surpresa os operadores sentirem que os desenvolvedores não se importavam suficientemente com a qualidade.

Quando esse impedimento apareceu, ele foi escalado primeiramente para o gerente de linha e, em seguida, para o gerente de departamento. Este agendou uma reunião juntamente com o líder de desenvolvimento. Nas discussões seguintes, os gerentes concordaram em priorizar a qualidade. Eles criaram uma solução de suporte rotativo – a cada *sprint*, uma equipe de desenvolvimento deveria ficar de plantão e disponível, instantaneamente, para ajudar nas operações. Depois de garantir o suporte de seus gerentes, o líder de desenvolvimento pegou uma lista de contatos das pessoas das equipes de suporte. Imediatamente, eles testaram a solução, pensando que o plano não funcionaria. Mas, desta vez, o dever de casa foi levado a sério e o impedimento considerado resolvido. Esse fato trouxe grande segurança às equipes de operações.

29

Encontrando um conceito de planejamento que funcione

Uma História

Lembro-me de um momento importante para uma das equipes. Eu estava sentado com eles na segunda sessão de estimativa. A equipe estava travada com um projeto sem saber como estimar. Havia muitas dúvidas e toda a sessão de estimativa ficou parada. Ao invés de interferir e assumir o controle, eu pedi a eles que refinassem o processo para encontrar uma solução melhor. Liderados pelo gerente, eles aceitaram o desafio e começaram a projetar uma solução própria. Esse evento foi um momento decisivo, uma vitória importante a partir do qual eles se transformaram em uma equipe com confiança. Depois disso eles começaram a evoluir tão rapidamente que nós tivemos que sair do caminho.

Dois meses depois, o gerente me abordou após uma retrospectiva. "Eu tenho um problema", disse ele, apontando para o quadro do Kanban da equipe. "Nós não temos problemas reais, o que eu devo fazer?"

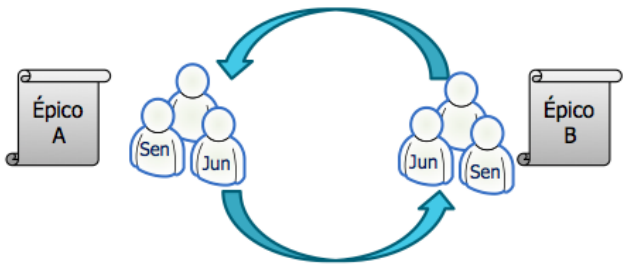
Reinventando o planejamento

As sessões de estimativa com *Planning Poker* envolvendo todos os membros da equipe não funciona bem para todos as equipes de operações. Eis algumas razões:

- O conhecimento foi difundido muito uniformemente na equipe.
- Freqüentemente apenas uma pessoa estava falando.
- Os membros da equipe queriam atacar problemas urgentes que pipocavam em suas mesas.

Mas, por experimentação, as equipes vieram, independentemente, com dois processos de estimativa diferentes. Cada um funcionou bem para a respectiva equipe.

Abordagem 1 - Troca e revisão



- Para cada projeto/história, associe um par sênior + júnior para fazer as estimativas (i.e., uma pessoa que conheça bem a história e uma que não a conheça). Isso ajuda a difundir o conhecimento.
- Os demais membros da equipe escolhem qual história eles querem ajudar a estimar (mas com o limite de quatro pessoas por história, para manter as discussões efetivas).

- Cada equipe de estimativa quebra suas histórias em tarefas e, se necessário, estima também as tarefas.

- Então as equipes trocam histórias e fazem revisão umas das outras (uma pessoa de cada equipe permanece "nos bastidores" para explicar o trabalho de sua equipe para os revisores).

- Pronto!

Normalmente todo um ciclo de uma sessão de planejamento levou cerca de 45 minutos e o nível de energia permaneceu alto ao longo da reunião. 1 ou 2 ajustes ainda foram feitos a partir desta troca e revisão de histórias.

Abordagem 2 – avaliação sênior em alto nível, e então a estimativa

Dois membros seniores do time fazem uma revisão de alto nível da história/projeto antes do planejamento. Eles devem analisar soluções de arquitetura e escolher uma delas para o problema. Uma vez em acordo, o time deve entrar e quebrar a história/projeto em tarefas, usando a solução proposta como um ponto de partida.

Figura 8. Quebra de tarefas com *peer-review* por outro time no planejamento da iteração.

30

O que medir?

Há muitas coisas que podem ser medidas - o ciclo de tempo (tempo que leva da descoberta da necessidade até que ela seja satisfeita), velocidade, filas, *burndowns*. A questão importante é que as métricas podem ser usadas para melhorar o processo. Meu conselho é experimentar e ver o que funciona para você. Aprendemos que gráficos *burndown* foram um exagero para alguns projetos mais curtos do que 1-4 semanas. A base do progresso poderia ainda ser avaliada com uma olhada para o quadro Kanban (quantas histórias estavam no *backlog* e quantas foram feitas).

Métrica candidata	**Pró**	**Contra**
Tempo de Ciclo	Fácil de medir. Não necessita de estimativa. Começa e termina com cliente.	Não leva tamanho em consideração.
Velocidade Total (agregada através de todos os tipos de trabalho)	Um fácil e grosseiro indicador para direcionar melhoria e variação.	Não ajuda a prever datas de entrega para tipos de trabalho específicos.

Velocidade por tipo	Mais precisa que velocidade total.	Para ser útil, deve iniciar de uma necessidade de cliente até a entrega. Toma um pouco mais de tempo para rastrear comparada à velocidade total.
Comprimento de Fila	Um rápido indicador de flutuação de demanda. Fácil de visualizar.	Não informa se a causa é demanda desigual ou capacidade desigual. Uma fila zerada pode indicar na realidade uma sobrecarga.

Nós começamos medindo "a velocidade por tipo de trabalho" e "o tamanho das filas". A velocidade por tipo de trabalho é simples de medir e faz o trabalho. Os tamanhos das filas são bons indicadores desde que sejam mostrados instantaneamente (se você souber onde procurá-los).

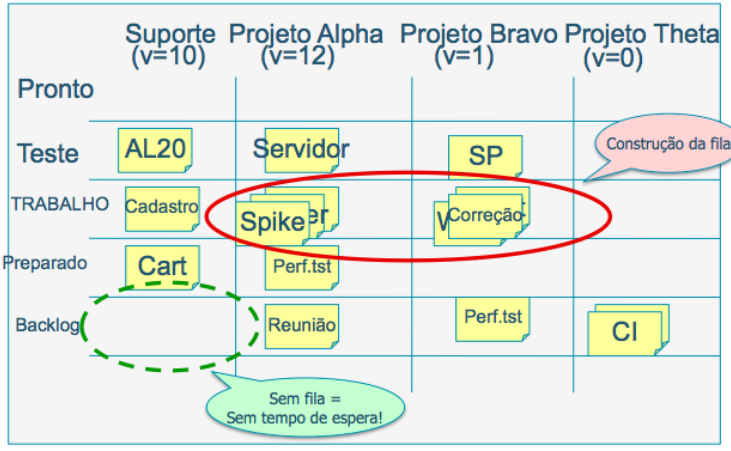
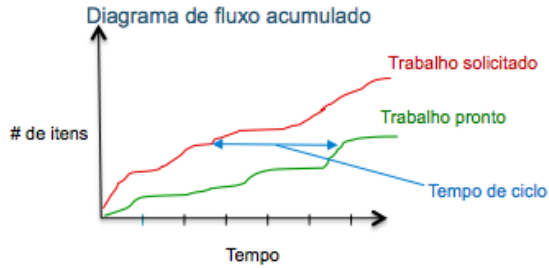


Figura 9. Gargalos e oportunidades. A área vermelha mostra como as filas se desenvolveram, mostrando um gargalo de teste. A *falta* de uma fila na coluna de *backlog* de suporte indica que novos trabalhos de suporte não podem esperar. Isso é um bom sinal para um alto nível de serviço.

Nós não usamos um diagrama de fluxo cumulativo, mas seria interessante.



Nós não usamos diagramas de fluxo cumulativos porque o quadro do Kanban e o gráfico de velocidade já nos deram informações suficientes, pelo menos em nossas fases iniciais de maturidade. Gargalos, irregularidades e excesso de trabalho poderiam também ser facilmente identificados e resolvê-los nos manteria ocupados por mais seis meses.

31

Como as coisas começaram a mudar

Três meses depois de introduzir o Kanban, a equipe do sistema de administração ganhou da Direção o título de "equipe com a melhor performance" no departamento de TI. Ao mesmo tempo, a equipe foi votada como umas das 3 melhores experiências na retrospectiva da companhia. A "retrospectiva da companhia" é um evento de toda a empresa que acontece a cada 6 semanas, e esta foi a primeira vez que uma equipe entrou na lista das 3 melhores! E, apenas 3 meses antes, essa equipe havia recebido diversos tipos de reclamações devido a pontos de gargalo.

A qualidade do serviço claramente melhorou. E como isso aconteceu?

O momento crucial foi quando todos começaram a trabalhar em conjunto. A Direção forneceu um foco claro e protegia a equipe de coisas externas aos seus trabalhos, e a equipe assumiu a responsabilidade sobre a qualidade e os prazos. Isto levou uns três ou quatro meses até funcionar, mas depois fluía sem problemas. Não é que todos os problemas tinham ido embora (isso colocaria a todos para fora do trabalho, certo? ☺) – mas nós encontramos novos desafios do tipo "como nós iremos manter a equipe motivada para melhorar (quando eles não forem mais o gargalo)?".

Uma peça importante no quebra-cabeças da auto-organização foi a introdução do conceito “uma pessoa de contato de operação por time”. Cada time de desenvolvimento tinha seu próprio contato no time de operações. O Kanban tornou isso possível permitindo aos membros do time se auto-organizarem ao redor do trabalho, prevenindo trabalho em excesso e habilitando melhoria contínua. Antes, uma pessoa qualquer do time poderia puxar trabalho da fila, e resolver, da melhor forma, um chamado, conforme sua habilidade, e depois iniciar o próximo. Qualquer falha de comunicação significava iniciar tudo novamente, com um novo chamado de suporte. Quando o conceito “um ponto de contato para cada time de desenvolvimento” foi lançado, o time de suporte teve a oportunidade de responder mais rapidamente a falhas na entrada ou a problemas de qualidade que ameaçavam o sistema.

Curiosidade – depois de algum tempo, protocolos de comunicação evoluíram; pessoas do time de operações começarem a usar mensagens instantâneas para desenvolvedores que eles conheciam bem, e-mails para aqueles que escrevem melhor do que falam, e telefones, se essa fosse a maneira mais rápida de resolver o problema ☺.

Antes

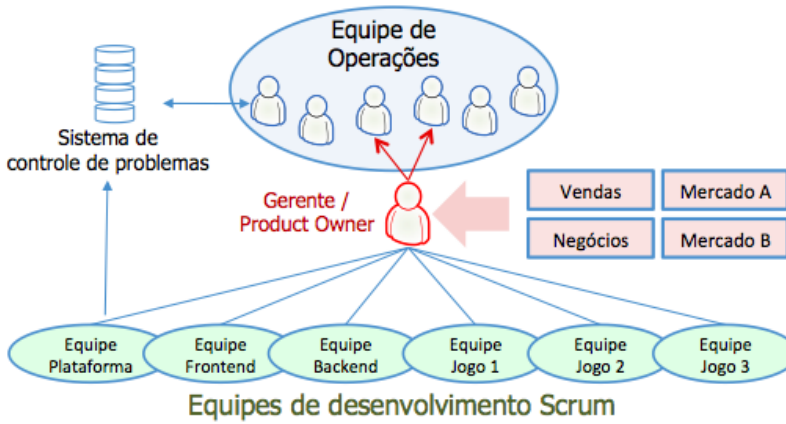


Figura 10. Antes: Gerente de primeiro nível é o ponto de contato com o time. Qualquer coisa importante que precise ser feita, passará por ele. Pequenas situações, como problemas de desenvolvedores, são recebidas por um sistema de controle de chamados. Pouca iteração entre as pessoas ocorrendo.

Depois

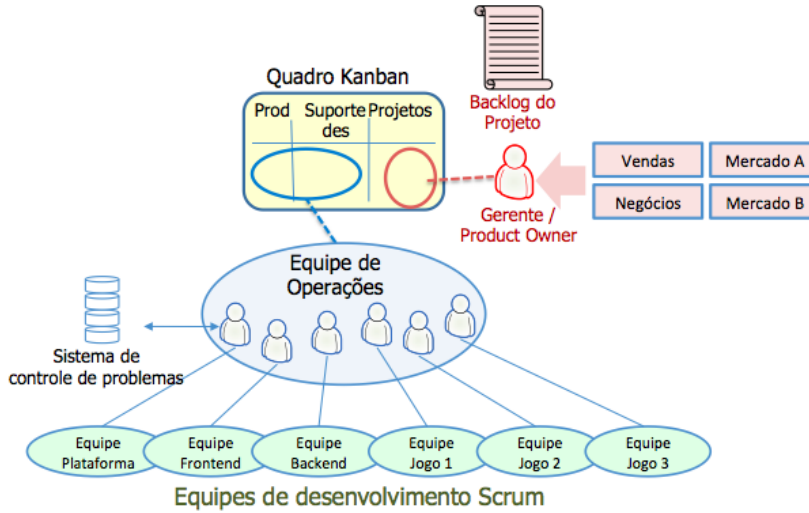


Figura 11. Depois: "um contato das operações por equipe" implantado. As equipes de desenvolvimento falam diretamente com o contato das operações definido. Muita interação pessoa-a-pessoa. Os membros da equipe de operações auto-organizam o seu trabalho utilizando o quadro Kanban. O gerente volta o foco para a priorização de projetos maiores e fornece apoio quando problemas maiores surgem.

Então, qual o impacto no desempenho da equipe?

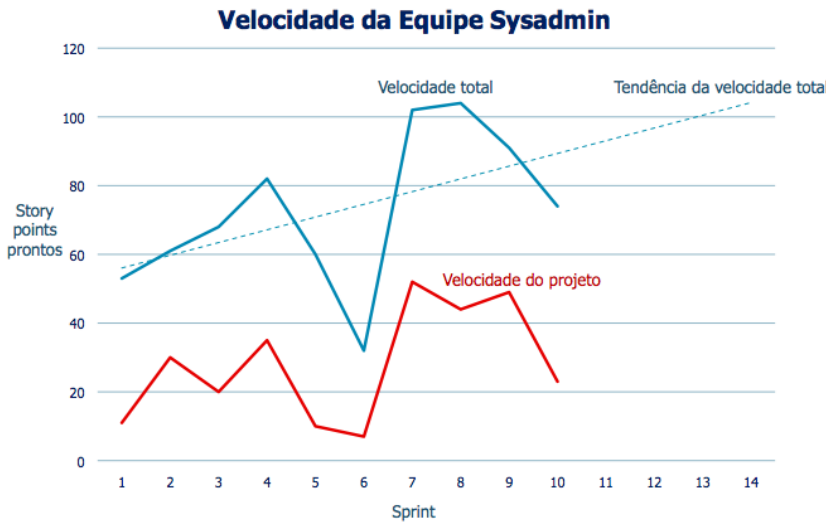


Figura 12. Velocidade total e velocidade do projeto, medidos como pontos da história "prontos" por semana. O total é a soma de todas as colunas, a velocidade do projeto representa a parte dedicada a "projetos" (grandes esforços de trabalho, por exemplo, fazer *upgrade* na plataforma de hardware). As duas quedas relacionam-se a 1) uma semana em que quase todos os membros de equipes estavam viajando e 2) uma grande release do desenvolvimento.

Assim, o time demonstrou uma tendência positiva de forma geral. Ao mesmo tempo, a equipe investiu pesadamente em compartilhamento de conhecimento usando programação em par.

Enquanto isso, vamos dar uma olhada no desempenho da equipe;

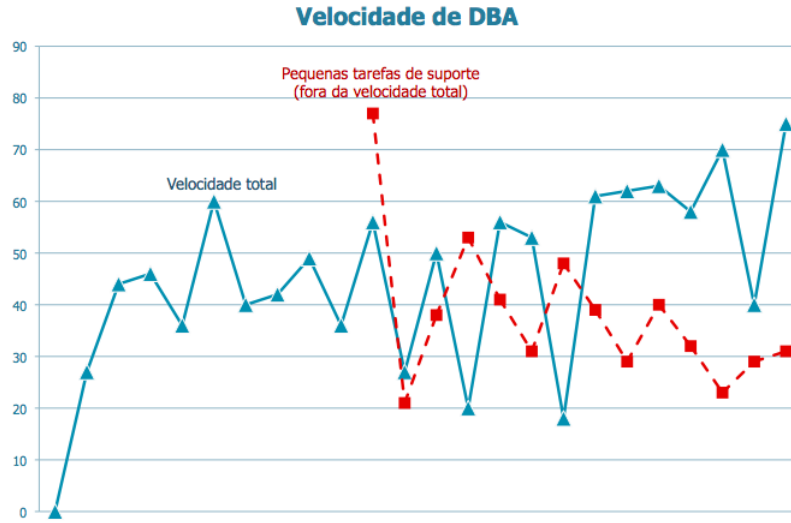


Figura 13. Velocidade Total e pequenas tarefas de suporte. A queda bem no meio corresponde ao Natal.

A velocidade total tende para cima embora a variação seja significativa. O tamanho da variação inspirou a equipe a monitorar o número de pequenas tarefas de suporte (tarefas muito pequenas para serem adicionadas ao quadro Kanban). Como você pode notar, o gráfico indica a clara correlação inversa entre o número de pequenas tarefas de suporte e a velocidade total.

A equipe de suporte começou a usar Kanban mais tarde do que as outras duas equipes. Então, nós ainda não temos tantos dados confiáveis.

Aumento de maturidade

Quando começamos, encontrar áreas problemáticas era fácil. Mas achar a grande oportunidade para melhorias era difícil. O quadro Kanban nos propiciou todo um novo nível de transparência. Não apenas ficou fácil apontar os problemas, como também levantar questões importantes sobre fluxo,

variações e etapas. Nós começamos utilizando as etapas como uma maneira para identificar problemas. Quatro meses depois de começar a usar Kanban, os gerentes estavam procurando as origens das variações que prejudicavam suas equipes.

Conforme as equipes evoluíam de grupos de indivíduos para unidades auto-organizáveis, os gerentes perceberam que eles estavam se deparando com um novo conjunto de desafios em termos de liderança. Eles tinham de tratar mais com questões em nível pessoal – lidar com reclamações, definir objetivos compartilhados, resolver conflitos e negociar acordos. Não foi uma transição indolor – eles disseram abertamente que para aprender a lidar com tudo isso foi necessário muito jogo de cintura e muita paciência. Mas eles aceitaram o desafio e, no fim das contas, acabaram por se tornar líderes melhores.

32

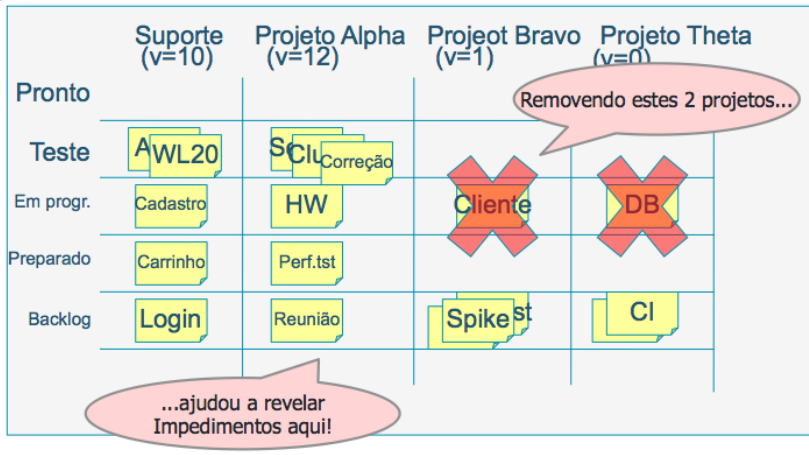
Lições gerais aprendidas

À medida que o trabalho em andamento progride, limitações surgem

Todas as equipes começam com limites de trabalho em andamento bem generosos. Nesse momento, a maior parte da energia é consumida tentando criar fluxo e garantindo que a organização esteja obtendo o suporte de que necessita.

No começo, os gerentes queriam ter vários projetos em execução simultaneamente, mas, após algumas semanas, tornou-se evidente que não havia capacidade suficiente para lidar com os projetos de baixa prioridade. Bastava uma breve olhada no quadro para ver que nenhum trabalho sequer era iniciado nas coisas de baixa prioridade. Isso alertou os gerentes para reduzir o número de projetos por equipe.

Com o tempo, como o fluxo se tornou mais estável para o trabalho de alta prioridade, começamos a estreitar os limites do trabalho em andamento. Isso foi feito reduzindo-se o número de projetos em andamento (colunas) de três para dois e, então, para um. À medida que isso ocorria, restrições de fora da equipe começaram a aparecer. Membros da equipe começaram a reportar que não estavam conseguindo ajuda de outros na equipe, de modo que os gerentes mudaram sua atenção para lidar com isso.



Outra coisa que ficou claro foi o quanto a entrada de outras equipes pode prejudicar o desempenho. Foi difícil manter um fluxo adequado quando novos itens de entrada precisam constantemente de correção.

Estes problemas não eram visíveis antes de começar. A questão foi então "que problemas deveríamos atacar primeiro" - e como chegar a um consenso sobre isso. Com o quadro Kanban, todo mundo pôde ver como um problema específico impacta no *fluxo*, o que facilita obtermos o momento certo de se lidar com as questões ao longo das fronteiras organizacionais.

O quadro vai mudar ao longo do tempo, não fixe o *layout* em ferro

Todos os quadros Kanban mudam ao longo do tempo. Normalmente demora dois ou três *designs* até que um time ache um quadro que funcione bem. Então, investir uma grande quantidade de tempo no primeiro quadro é, provavelmente, desperdício. Tenha certeza de que você pode modificar a organização do quadro de forma fácil. Nós usamos fita preta para desenhar o *layout*. Estas eram fáceis de serem modificadas e poderiam funcionam bem em paredes e quadros brancos. Outra forma de organização que eu vi é desenhar no quadro linhas de guia com marcadores finos (mas tenha certeza que o desenho pode ser apagado! ☺).

Abaixo é mostrado um típico exemplo de organização de *layout*. Prioridades mudaram freqüentemente no início então, para evitar ter que mover colunas inteiras de notas adesivas para frente e para trás, o time colocou um número de prioridade acima de cada coluna.

Figura 14. Kanban inicial com notas para a prioridade corrente.

Não tenha medo de experimentar e falhar

A lição que eu tirei dessa aventura é que não há um ponto final. Nós falhamos no momento em que pensamos existir um. Há somente experimentações e aprendizado sem fim. Nunca falhar significa não aprender. Nós falhamos inúmeras vezes no caminho (desenho ruim do quadro, estimativas, *burndowns* redundantes...) – mas cada vez nós aprendemos algo novo e importante. Se tivéssemos parado de tentar, como poderíamos então estar aprendendo?

O sucesso do Kanban tem agora inspirado as equipes de gerência e de desenvolvimento Scrum a iniciarem experimentações com quadros Kanban. Talvez esse livro ajude!

Considerações finais

Comece com as retrospectivas!

Muitas opções e coisas para pensar, hein? Espero que esse livro tenha ajudado e esclarecer um pouco do nevoeiro. Pelo menos funcionou para nós. :o)

Se você está interessado em mudar e melhorar o seu processo, vamos tomar uma decisão para você agora mesmo. Se você não estiver fazendo retrospectivas regularmente, comece por elas! E tenha certeza de que elas levem a uma mudança de verdade. Traga um facilitador se necessário. Assim que você tiver estabelecido retrospectivas eficazes, você terá começado sua jornada de evolução rumo ao processo certo para o seu contexto – seja ele baseado em Scrum, XP, Kanban, uma combinação deles ou qualquer outra.

Nunca pare de experimentar!

Kanban ou Scrum não são o objetivo; aprendizado contínuo é. Uma das melhores coisas sobre software é o ciclo curto de *feedback*, que é a chave para o aprendizado. Portanto, use o ciclo de *feedback*! Questione tudo, experimente, falhe, aprenda e então experimente novamente. Não se preocupe em acertar desde o começo, porque você não acertará! Apenas comece em algum ponto e evolua a partir de lá.

A única falha *real* é a falha em aprender com a falha.

Mas espere, você pode aprender com isso também

Boa sorte e curta a viagem!

/Henrik & Mattias, Estocolmo 24/06/2009

H: Isso é tudo o que temos?

M: Eu acho que sim. Vamos parar por aqui.

H: Talvez devamos contar a eles que somos?

M: Boa observação. Se fizermos parecer que somos caras legais, talvez possamos fazer consultorias.

H: Vamos fazer isso, então! E então paramos por aqui.

M: Sim, nós temos trabalho a fazer, e os leitores também.

H: Na verdade, minhas férias começam agora :o)

M: Ei, não esnoba.

Sobre os autores

Henrik Kniberg e Mattias Skarin são consultores da Crisp em Estocolmo. Eles gostam de ajudar empresas a terem sucesso tanto no lado técnico quanto no lado humano do desenvolvimento de software e já ajudaram dúzias de empresas a colocar em funcionamento princípios de Lean e metodologias ágeis.

Henrik Kniberg

Durante a última década Henrik foi o CTO de 3 empresas de TI suecas e ajudou muitas mais a melhorar seus processos. Ele é *Certified Scrum Trainer* e trabalha regularmente com pioneiros de Lean e *Agile* como Jeff Sutherland, Mary Poppendieck e David Anderson.

O livro anterior de Henrik ("Scrum e XP direto das Trincheiras") tem mais de 150.000 leitores e é um dos livros mais populares sobre o tema. Ele recebeu o prêmio de melhor palestrante em várias conferências internacionais.

Henrik cresceu em Tóquio e agora vive em Estocolmo com sua esposa Sophia e três filhos. Ele é músico nas horas vagas, compondo músicas e tocando baixo e teclado em bandas locais.

henrik.kniberg<at>crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin

Mattias trabalha como consultor Lean ajudando empresas de Software a aproveitarem os benefícios do Lean e do *Agile*. Ele assessora todas as camadas da organização, dos desenvolvedores aos gerentes. Ele ajudou uma empresa de jogos a reduzir o tempo de desenvolvimento de um jogo de 24 para 4 meses, recuperou a confiança de todo o departamento de desenvolvimento e foi um dos primeiros pioneiros do Kanban.

Como empreendedor, ele é co-fundador e administra duas empresas.

Mattias é graduado Mestre da Ciência em Gestão da Qualidade e trabalhou como desenvolvedor por 10 anos em sistemas de missão crítica.

Ele vive em Estocolmo e gosta de dançar *rock and roll*, correr e esquiar.

mattias.skarin<at>crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin

Glossário

Nesta tradução, mantivemos alguns termos em inglês, pois estes já fazem parte do vocabulário dos trabalhadores da área. Caso não tenha familiaridade com os termos e expressões, pode consultar as breves definições apresentadas abaixo. Em caso de mais alguma dúvida, não exite em nos enviar um e-mail com sua requisição para que ela seja acrescentada nas próximas revisões.

Termo em Inglês	**Significado**
Acceptance Test	Teste de aceitação ou teste funcional
Burndown chart	Gráfico de *burndown*: gráfico que apresenta referência de trabalho estimado para uma iteração, visualmente informando se os objetivos estimados têm tendência de serem cumpridos ou não. "Taxa de "queima".
Daily Standup / Daily Scrum	Reunião diária, muitas vezes feita de pé para não se alongar muito.
Deploy	Montado / Instalado
Done	Pronto, forte conceito proveniente do Scrum.
Story Points	Pontos da história (*User Story*), unidade usada para estimar histórias de usuário.
Kanban	Quadro de sinalização, também é uma metodologia ágil.
Lean	Enxuto, *Lean Manufacturing* é proveniente do Modelo Toyota de

	Produção, dele foi originada a *Lean Software Development*, uma metodologia Ágil
Live!	Vivo! Em produção! Funcional!
MMF (Minimum marketable feature set)	Conjunto mínimo aceitável de funcionalidades.
Ongoing	Iniciado
Peer-Review	Revisão por pares
Product Backlog	Pilha priorizada de funcionalidades, um artefato do Scrum.
Release	Entrega
Scrum	Metodologia Ágil
ScrumMaster	Líder da equipe Scrum
Sprint	Iteração Scrum

Sprint Backlog	Pilha de funcionalidades priorizada, proveniente do *product backlog*, mas visando o escopo apenas de um *sprint*.
Stakeholder	Parte interessada
Sticky Note	Nota Adesiva, Nota Colante, Post-It
Team	Time ou Equipe
Timebox	Duração fixa, determinada, forte conceito Scrum
Value Stream Map	Mapa de fluxo de valor, ferramenta muito utilizada no Lean
Work-in-Progress (WIP)	WIP – Atividades (ou trabalho) em andamento
Workflow	Fluxo de trabalho
Workshop	Oficina

Sobre a tradução

Este livro foi traduzido voluntariamente por brasileiros, nas mais diversas funções, com os mais diversos conhecimentos das línguas cujo único interesse foi o de disseminar e popularizar as metodologias ágeis

Coordenação

Renato Willi	renato.willi@gmail.com
Vinicius Assef	viniciusban@gmail.com

Revisão

Caso o leitor encontre algum erro ou falha, ou queira fazer uma revisão completa da tradução do livro, entre em contato com um dos coordenadores para enviar sua revisão e ter seu nome registrado aqui.

Versão 1.0

Adriana Luppi	dirsluppi@hotmail.com
Daniel Wildt	dwildt@gmail.com
Rafael Fuchs	rafaelfuchs@gmail.com
Vinicius Assef	viniciusban@gmail.com
Paulo Victor G. Gross de Souza	pvggds@gmail.com

Ilustrações

Eduardo Bobsin Machado	eduardo.bobsin@gmail.com
Renato Willi	renato.willi@gmail.com
Vitor Machel	invasao@gmail.com

Tradução

Juliana Berossa Steffen	julianaberossa@gmail.com
Marcelo Andrade	mfandrade@gmail.com
Eduardo Bobsin	eduardo.bobsin@gmail.com
Rodrigo Russo	rodrigozrusso@gmail.com
Daniel Wildt	dwildt@gmail.com
Luciano Costa	lucianocosta.info@gmail.com
Renato Willi	renato.willi@gmail.com
Marcos Vinícius Guimarães	marcosvafg@gmail.com
Adam Brandizzi	brandizzi@gmail.com
André Pantalião	andre0610@gmail.com
Cássio Marques	cassiommc@gmail.com
Ismael Stahelin	nilehats@gmail.com
Rafael Fuchs	rafaelfuchs@gmail.com
Gerson Dias	gerson.dias.job@gmail.com
Ian Gallina	ian.gallina@gmail.com

Rafael Dantas	rafaeldantas@gmail.com
Vitor Machel	invasao@gmail.com
Vinicius Assef	viniciusban@gmail.com
Bruno Pedroso	brunopedroso@gmail.com
Cassiano Alves	cassiano.alves@gmail.com
Gustavo Grillo	gustavogrillo@gmail.com
Igo Coelho	igocoelho@gmail.com
Rafael Prikladnicki	rafael.prikladnicki@gmail.com
Adriana Luppi	drisluppi@hotmail.com

Apoio

SEA Tecnologia	www.seatecnologia.com.br
Caelum	www.caelum.com.br/