Jesper Boeg

Kanban em 10 Passos

Otimizando o fluxo de trabalho em sistemas de entrega de software

InfoQ BRASIL
InfoQ queue
TRIFORK. driving software innovations

Kanban em 10 Passos

Jesper Boeg

Tradução para português e revisão
Leonardo Campos
Marcelo Costa
Lúcio Camilo
Rafael Buzon
Paulo Rebelo
Eric Fer
Ivo La Puma
Leonardo Galvão
Thiago Vespa
Manoel Pimentel
Daniel Wildt

Coordenação de tradução
Leonardo Campos

Edição e revisão final
Leonardo Galvão

http://infoq.com/br

Sobre o autor

Jesper Boeg trabalha como coach de Agile e Lean desde o início de 2006. Hoje é Vice-Presidente do departamento de Excelência em Agile da empresa dinamarquesa Trifork (parceira do InfoQ nos eventos QCon San Francisco e QCon London). Tem Mestrado na área de Sistemas de Informação na Universidade Aalborg, com uma tese sobre como gerenciar com sucesso equipes distribuídas de desenvolvimento de software. Jesper apoia empresas e organizações na adoção de princípios Lean e Agile, com enfoque em deixar claro o porquê de cada princípio. Jesper realiza com frequência apresentações em conferências Agile e Lean. É membro da comissão de organização do evento GOTO Aarhus e atuou como coordenador de trilhas em várias conferências QCon e GOTO.

Conteúdo

Contexto e Fundamentos

Antes de mergulhar no guia passo a passo de implementação do Kanban, vamos gastar alguns minutos apresentando os conceitos para que se possa reconhecer onde cada passo se encaixa no framework Kanban de gestão de mudanças. O escopo deste minilivro não é descrever a fundo os conceitos do Kanban; para isto, recomendo a leitura do excelente livro "Kanban" de David J. Anderson.

Ao invés disso, faço uma pequena introdução seguida de conselhos no estilo passo a passo sobre como iniciar no Kanban.

O Kanban, ou mais precisamente o "sistema Kanban para desenvolvimento de software" representa uma implementação mais direta dos princípios de Desenvolvimento Lean de Produtos para o desenvolvimento de software que os métodos ágeis tradicionais. Com foco consistente no fluxo e no contexto, o Kanban oferece uma abordagem menos prescritiva comparada ao Agile, e tem se tornado uma extensão popular dos métodos ágeis tradicionais como Scrum e XP.

A palavra "Kanban" vem do japonês e significa "Cartão Visual". Uma pesquisa pela palavra no Google retorna mais de 5 milhões de resultados; isto porque a palavra também é utilizada para descrever o sistema que vem sendo utilizado há décadas pela Toyota para visualmente controlar e equilibrar a linha de produção. O termo tem se tornado quase sinônimo da implementação dos princípios Lean. Então, embora sistemas kanban sejam conceito relativamente novo em TI, vêm sendo utilizados por mais de 50 anos no sistema de produção Lean na Toyota.

O pioneiro no uso do Kanban em software foi David J. Anderson, que em conjunto com Don Reinerstsen, tem se empenhado para expandir o conhecimento do Lean e o uso do Kanban para visualizar e otimizar o fluxo de trabalho no desenvolvimento de software, nas áreas de manutenção e de operações.

Quando devo considerar trabalhar com o Kanban?

Se a resposta for “sim” para uma ou mais das questões abaixo, existe uma boa chance de que você se beneficie da leitura deste livro:

- Você tem se esforçado para implementar o Agile em sua organização, mas sem muito sucesso?
- Você tem usado o Agile por algum tempo, mas as melhorias de desempenho começaram a estagnar?
- Você está utilizando tempo precioso em práticas ágeis que já não parecem mais se encaixar no contexto em que está trabalhando?
- Tem usado Agile como um checklist, mas sem compreender completamente os princípios básicos?
- Sente a necessidade de flexibilidade que vá além da oferecida por iterações fixas e planejadas?

- Suas prioridades mudam diariamente?
- Você está utilizando processos criados para o desenvolvimento ágil num contexto em que não se adaptam facilmente, como por exemplo em manutenção e operações?
- Precisa de uma transição gradual da execução em cascata para o Agile, para evitar altos níveis de resistência organizacional?

Se seu objetivo é trabalhar em um contexto estritamente voltado ao Scrum, ou se atualmente está utilizando um processo cascata, ou se está tentando encontrar um caminho para otimizar sua implementação Agile atual – independente do caso, a maioria pode se beneficiar de conhecimento mais profundo em Lean – e o Kanban provou ser um excelente catalisador nessa direção.

O que é Kanban?

Existem diversas abordagens para o Kanban, mas a maioria dos especialistas concorda que o Kanban é um método de gestão de mudanças, que dá ênfase aos seguintes princípios:

- Visualizar o trabalho em andamento;
- Visualizar cada passo em sua cadeia de valor, do conceito geral até software que se possa lançar;
- Limitar o Trabalho em Progresso (WIP – Work in Progress), restringindo o total de trabalho permitido para cada estágio;
- Tornar explícitas as políticas sendo seguidas;
- Medir e gerenciar o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências dessas decisões;
- Identificar oportunidades de melhorias, criando uma cultura Kaizen, na qual a melhoria contínua é responsabilidade de todos.

Tudo isso, com a filosofia subjacente de que se deve:

- Começar com o que se está fazendo agora;
- Concordar em buscar mudanças incrementais e evolucionárias;
- Respeitar o processo atual, com seus papéis, responsabilidades e cargos.

Quem está familiarizado com o Lean irá reconhecer muitos desses princípios como sendo a fundação para um sistema "puxado"(pull) Lean; forma também as bases para uma cultura de melhoria contínua (Kaizen). O que o Kanban faz, em primeiro lugar, é servir como um catalisador para introduzir ideias Lean na entrega de sistemas de software. Isso também é declarado por David Anderson em seu livro:

> O **Kanban** (com K maiúsculo) é o método de mudança evolucionária que utiliza um sistema **kanban** (com k minúsculo), além da visualização e outras ferramentas, para catalisar a introdução das ideias Lean nas áreas de desenvolvimento e operações de TI.

Mostraremos diversos exemplos de quadros kanban nos capítulos a seguir, além de explicar a mecânica da técnica. Para se ter uma ideia dos conceitos fundamentais, a **Figura 1** mostra um exemplo.

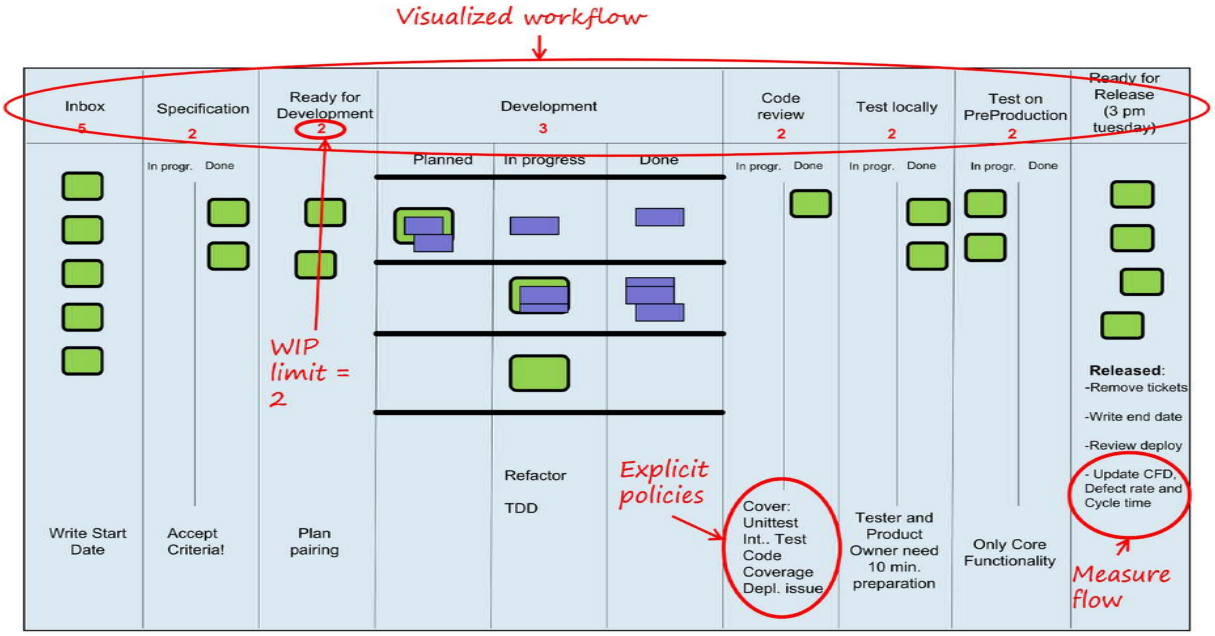

Figura 1. Princípios do Kanban em ação

Como se vê, todo o trabalho se torna visível com o Kanban. Os limites de WIP são estabelecidos (escritos na parte de cima de cada coluna). As políticas se tornam explícitas e o fluxo passa a ser medido. Note que a última coluna não possui limite de WIP, devido à equipe ter optado especificamente por um ritmo semanal de entregas (3 da tarde na terça-feira. Isso significa que todo o trabalho finalizado é liberado neste momento, semanalmente.

O foco do Kanban é conduzir mudanças evolucionárias, e estes passos simples têm-se provado extremamente úteis para esse objetivo. A razão pela qual nos referimos a esse sistema como um "Sistema Puxado Kanban" (Kanban Pull System) é que, ao visualizar o fluxo e estabelecer os limites de WIP, garantimos que nunca se pode introduzir mais trabalho no sistema que a capacidade do sistema de processar esse trabalho. Está disponível apenas uma quantidade limitada de permissões de trabalho ("kanbans"); então é necessário que se complete o trabalho existente antes que um novo trabalho possa ser iniciado. Isto resulta em funcionalidades sendo *puxadas* pelo sistema, com base na capacidade deste, em vez de empurradas com base em previsões ou demandas.

Não há regras quanto à aparência específica do quadro. As únicas limitações são a imaginação, a criatividade e as restrições de um sistema eletrônico, ou o espaço na parede. Como este minilivro é um guia para principiantes, iremos utilizar exemplos bastante simples de quadros para demonstrar os fundamentos.

Como começar com o Kanban?

Espera-se que as 10 etapas apresentadas neste texto auxiliem o leitor a começar bem com o Kanban – mas antes de chegarmos nesse ponto, é importante compreender que o Kanban aborda a gestão de mudanças de forma diferente que a maioria dos métodos ágeis.

O Kanban é construído sobre os conceitos da mudança evolucionária. Uma abordagem é começar a entender como funciona atualmente seu sistema de entrega de software. Quando conseguir visualizar, medir e gerenciar o fluxo sendo utilizado, melhore-o um passo de cada vez, aliviando o seu maior gargalo. Isso é muito diferente do que ocorre, por exemplo, no Scrum – onde se começa redefinindo papéis, processos e artefatos. Isso faz do Kanban um método ideal para utilização em conjunto com processos existentes, que podem ser desde o Scrum até Cascata. O Kanban também é excelente em situações em que estruturas organizacionais inibem mudanças radicais. Lembre-se desse princípio fundamental: "Respeite o processo atual, seus papéis, responsabilidades e cargos"

Em termos do Lean, isto significa que o Kanban é construído principalmente sobre o conceito de Kaizen (melhoria contínua). O Kanban apenas usa Kaikaku (mudança dramática) em situações especiais, nas quais mudanças estruturais são necessárias ou quando sérias melhorias de desempenho precisam ser realizadas.

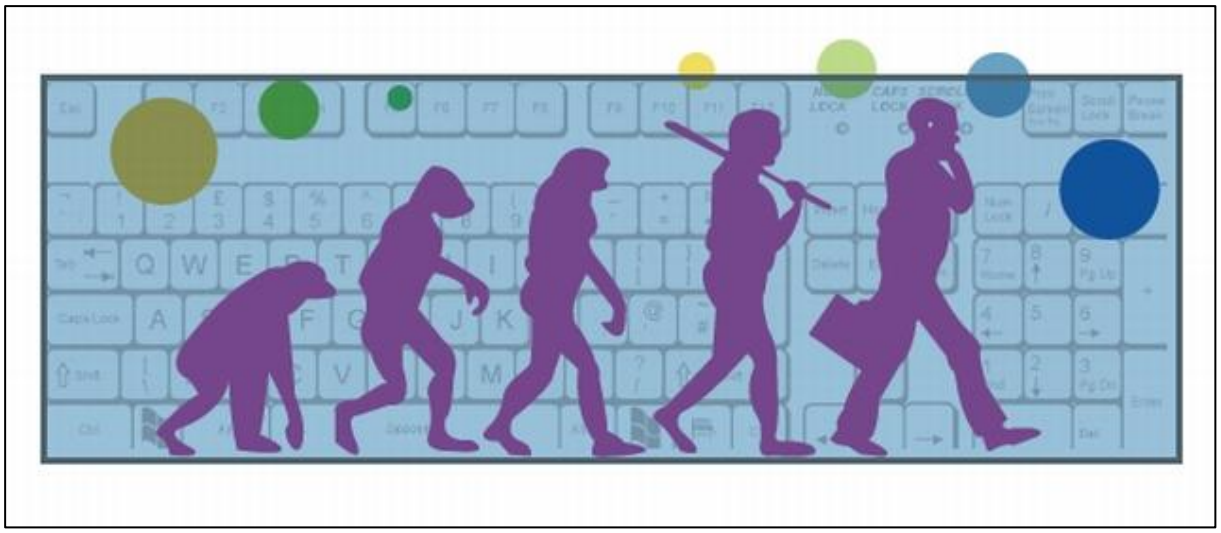

Figura 2. O Kanban segue uma abordagem evolucionária para a otimização de processos

Onde o Kanban pode ser utilizado?

Agora estamos quase prontos para iniciar. Porém, antes de mergulhar nos detalhes da implementação, vamos falar rapidamente sobre os mitos do Kanban, para garantir que as seções a seguir sejam lidas com os conceitos corretos em mente.

Na Trifork, ajudamos várias empresas e equipes a aumentar a efetividade por meio da adoção do Kanban. No início, parecia que os principais grupos alvo eram as equipes de manutenção e operações, mas o Kanban se provou útil também para o desenvolvimento de software. Equipes e organizações trabalhando com o modelo cascata também perceberam nessa abordagem evolucionária a possibilidade de uma transição gradual para o desenvolvimento ágil de produtos.

Mitos e fatos do Kanban

Mito: O Kanban é adequado apenas para equipes trabalhando com tarefas pequenas e padronizadas, como as de operações e manutenção.

- **Fato:** O Kanban é altamente inspirado pelo trabalho de Don Reinertsen, sobre o Desenvolvimento Lean de Produtos. E se provou ser tão boa opção para o desenvolvimento de software quanto é para operações e manutenção.

Mito: O Kanban e o Scrum são opostos.

- **Fato:** Nenhum dos princípios do Kanban restringe o uso do Scrum. O Kanban funciona como um agente de mudanças, e os princípios do Scrum devem, portanto, ser usados apenas nos casos em que ajudam a otimizar o fluxo de trabalho. Nada impede de se começar com o Scrum e utilizar o Kanban para impulsionar futuras mudanças – muitos projetos têm sido muito bem sucedidos com essa estratégia. Pode-se até argumentar que esta era essa a intenção original com o Scrum também, mas de alguma forma se perdeu com o foco em cerimônias, papéis e artefatos.

Mito: Por não insistir no compromisso com iterações planejadas, o Kanban torna-se vítima da Lei de Parkinson, em que "O trabalho se expande de modo a preencher o tempo disponível para a sua conclusão".

- **Fato:** Apesar de ser uma preocupação válida, os projetos com Kanban raramente apresentam esse comportamento, pois as cadências fixas, a visualização permanente, a medição do tempo de ciclo, e os ciclos de feedback curtos com o negócio, mantêm o foco controlado e as tarefas fluindo.

Mito: As equipes que utilizam Kanban não utilizam timeboxes (períodos fixos de tempo).

- **Fato:** Os timeboxes não são exigidos no Kanban, mas devem ser utilizados quando ajudarem a otimizar o fluxo, o nível de feedback e a qualidade. A maioria das equipes Kanban praticam uma cadência fixa de planejamento, revisão e entregas – mas sem vincular estritamente essas cadências. Desse modo, dispensa-se o modelo tradicional de iterações, mas preservando o valor gerado.

Mito: As equipes Kanban não fazem estimativas

- **Fato:** As estimativas não são obrigatórias, mas devem ser usadas quando apropriadas. A maioria dos projetos Kanban de desenvolvimento usam algum grau de dimensionamento inicial – ou para assegurar o gerenciamento, a priorização e o alinhamento ideal de portfólio, ou para os trabalhos considerados mais sensíveis à análise de custo/benefício ou ao desempenho relacionado a prazos.

Mito: O Kanban é melhor do que Crystal/Scrum/XP/FDD etc.

- **Fato:** O Kanban é antes de tudo um catalisador para a condução de mudanças e precisa de um ponto de partida. Assim, apesar de a maioria dos projetos poder se beneficiar do uso do Kanban, o Kanban não é um substituto para, por exemplo, o Scrum, que é, na maioria dos casos, um ponto de partida perfeito para a adoção do Kanban.

Embora a comunidade Kanban esteja constantemente lutando contra esses e outros mitos, o Kanban ainda continua até hoje sendo um dos conceitos mais mal compreendidos na comunidade ágil, principalmente pelas duas razões abaixo:

Boa parte dessa confusão se devem ao fato de o Kanban ser um método de mudanças e de existirem nele poucas partes descritivas dizendo como se trabalhar, quais os papéis existentes etc. Como o conceito de um método de mudanças é mal compreendido, as pessoas tentam compará-lo com métodos prescritivos como o Scrum e o XP.

Exemplos locais e comportamentos emergentes do Kanban, em projetos do mundo real, passaram a representar um "método Kanban" para algumas pessoas. É fácil ver porque há tanta incompreensão, pois muitos projetos Kanban apresentam as mesmas práticas emergentes. No entanto, a realidade é que o Kanban se apoia no uso dos princípios Lean para otimizar processos existentes, de forma evolucionária. Assim, não pode e não deve ser comparado com Scrum, XP, Crystal, FDD ou outro método que se esteja utilizando.

A segunda razão possível é que a palavra "Kanban" remete a muitos conceitos, que vêm da sua origem nos sistemas de produção Lean. O termo portanto talvez não seja palavra ideal para descrever um método de mudança para o desenvolvimento de software e operações de TI.

Embora os sistemas puxados kanban impulsionem mudanças na forma que são usados em sistemas de produção, o Kanban em software baseia-se em um conjunto muito mais amplo dos princípios Lean. Isso cria uma dissonância que dificulta o entendimento daqueles que trabalharam com Lean no passado.

Você vai perceber que muito da rejeição ao Kanban por partes da comunidade ágil tem como fonte esse mal-entendido.

A boa notícia é que a maioria dos projetos, ágeis ou não, podem ser beneficiados pelo uso dos princípios do Kanban, para impulsionar as mudanças e a melhoria contínua.

O leitor atento pode ter notado que o título deste livro não está alinhado exatamente aos princípios do Kanban. Como já disse David J. Anderson "o design do sistema Kanban é um processo de pensamento; não uma cópia ou modelo de implementação de processo". No entanto, que estiver familiarizado com o "Modelo de Dreyfus para aquisição de competências" irá reconhecer que para se adquirir uma nova competência, sempre são necessárias prescrições iniciais.

Minha esperança é que este texto introdutório ajude o leitor a fazer a transição mais rápida e suavemente. O mais importante é saber que as receitas aqui apresentadas servem apenas como uma forma de se adquirir conhecimento para avançar, e não como um ponto final ou uma lista de verificação a ser seguida cegamente. Os modelos e práticas sugeridos nos dez passos deste livro são comportamentos emergentes, constatados com experiência prática em projetos utilizando Kanban. O que é sugerido aqui não se trata do método de mudança Kanban em si.

Agora que temos uma ideia geral, vejamos como aplicar os conceitos na prática. Cada passo a seguir consiste de uma explicação concisa dos motivos, seguidos pelas práticas.

Passo 1: Visualizar o fluxo de trabalho

O primeiro passo para visualizar o seu fluxo de trabalho é entender como seu sistema atual funciona.

Compreendendo seu sistema de entrega de software

Para ser capaz de tomar decisões bem embasadas sobre a melhor forma de otimizar o fluxo de trabalho, o primeiro passo é entender o que está sendo feito atualmente – mas se deve resistir a tentação de já realizar mudanças. Além disso, deve-se mapear todo o fluxo de trabalho para a entrega de software, sem focar apenas na parte de "desenvolvimento".

Existem várias formas de se mapear o fluxo de trabalho. A mais popular é a utilização do conceito Lean de Mapeamento da Cadeia de Valor (Value Stream Mapping – VSM). Recentemente, esta prática Lean tem sido atacada por comunidades ágeis – o principal argumento é que o de que o trabalho com conhecimento não é um processo linear, diferentemente de outras atividades em sistemas de produção. Isso levou à criação da técnica de Redes de Criação de Conhecimento (Knowledge Creation Networks), que são mais indicadas para atividades não lineares.

Para o exemplo simples apresentado aqui, porém, utilizaremos a técnica VSM, que é mais simples, e que considero extremamente útil. Deve-se, contudo, explorar a opção que melhor se adapte ao contexto do leitor.

Em sua forma mais simples, um Mapa de Cadeia de Valor é uma visualização dos estágios pelos quais passa o trabalho, desde a matéria prima até o produto final – em software, de uma ideia vaga até uma funcionalidade implantada em produção. Para o trabalho de conhecimento (como o desenvolvimento de software), o mais importante é pensar em cada estágio como sendo a representação de um grupo de atividades.

Em um estágio chamado "Testes", por exemplo, há mais trabalho do que simplesmente testar, incluindo corrigir, refatorar, discutir, atualizar critérios de aceite etc. Mas como o termo mais representativo é "Testes", iremos definir nosso trabalho como estando neste estágio, enquanto todas essas atividades estiverem acontecendo. O espaço entre os estágios em que nenhuma informação está sendo adicionada é definido como "tempo de espera" (*wait time*). A **Figura 3** mostra um exemplo retirado de um projeto real.

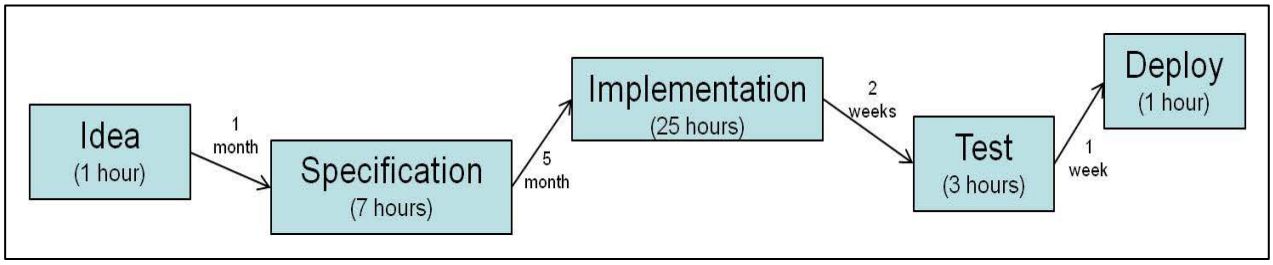

Figura 3. Exemplo de mapa de cadeia de fluxo de valor

Mais tarde, poderíamos perceber que a implementação, na verdade, consiste em Detalhamento das histórias de usuário e Desenvolvimento, seguido de Revisão de código – mas vamos focar

nessa versão simplificada por enquanto. Outras ideias de melhoria começam a surgir: Por que existe um tempo médio de cinco meses da especificação até a implementação? Por que estamos esperando duas semanas para testar? Resista à tentação de lidar com essas questões por enquanto, pois haverá tempo de sobra para isso mais tarde. O foco agora é entender como nosso sistema atual funciona.

Inicialmente, é uma boa ideia limitar o número de estágios, pois se pode perder rapidamente a visão do todo ao dar muita ênfase na mecânica. Mais tarde, pode ser benéfico adicionar mais detalhes e estágios, mas procure manter o fluxo o mais simples possível por enquanto.

Visualização do seu sistema de entregas

Agora que já temos um entendimento melhor de nosso sistema de entrega de software, o próximo passo é tentar visualizá-lo através de um sistema eletrônico, ou simplesmente utilizando um quadro branco. A menos que se esteja trabalhando em uma equipe distribuída, geralmente é uma boa ideia começar apenas com o quadro branco. Nada torna seu trabalho mais visível do que tê-lo à sua frente o tempo todo e poder tocá-lo fisicamente. À medida que cresce a maturidade da equipe e se sente a necessidade de coletar mais dados, pode-se querer mover a visualização do trabalho para uma versão eletrônica. Mas vamos nos ater ao quadro físico por enquanto.

Muitas vezes, você acabará com pelo menos dois tipos de estágios: os de "Atividade", nos quais o trabalho está sendo de fato realizado, e os de "Buffer", nos quais o trabalho está aguardando para ser lançado/desenvolvido etc. Veremos mais sobre isso adiante.

A primeira versão do seu quadro pode parecer com a mostrada na **Figura 4**. Perceba que todo o trabalho necessário para se completar uma funcionalidade é representado no quadro, e não apenas o desenvolvimento.

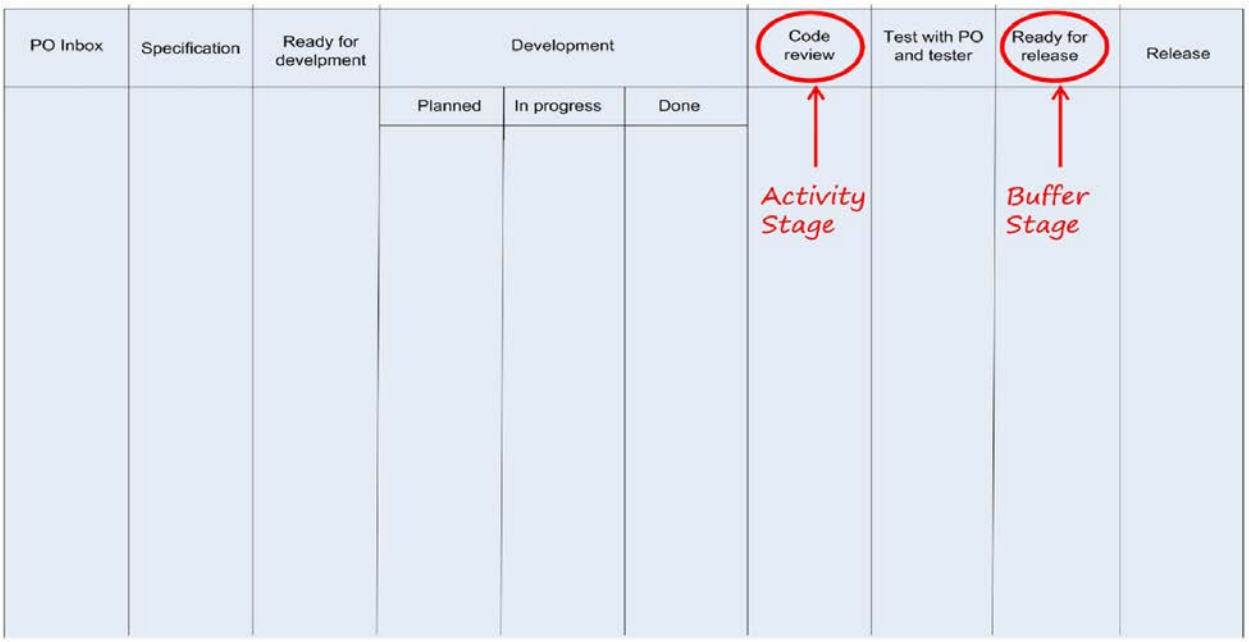

Figura 4. Estágio de atividade e estágio de buffer

Toda funcionalidade começa como uma ideia geral na "Caixa de entrada do PO" (*PO Inbox*, na imagem) e termina na coluna “Pronto para produção” (*Ready to Release*, na imagem). A funcionalidade é removida desta última coluna quando está liberada e em produção. Note que não fizemos mudança alguma – pode-se ainda estar usando uma implementação estrita do Scrum.

Caso visualizar seu trabalho seja difícil, pare. Não vá além, a não ser que tenha conseguido visualizar todo o trabalho que você faz. Se ainda existe informação escondida e algumas tarefas estiverem completamente fora do seu sistema de entregas, há pouca chance de que se consiga tomar decisões fundamentadas sobre a melhor forma de otimização. Uma regra geral é que se pode gerenciar apenas o trabalho que se pode ver.

Visualizar o trabalho é difícil na vida real. As razões para a dificuldade são das mais variadas. Por exemplo, as pessoas sabem que estão fazendo coisas que não deveriam; têm medo de serem punidas caso seus superiores saibam como as coisas realmente funcionam; e apesar de cansadas e sob pressão, sentem que deixarão seus colegas desamparados se não estiverem constantemente apagando incêndios. Antes de seguir em frente é necessário corrigir esses problemas, e fazer com que todos os envolvidos acreditem que ninguém será culpado ou responsabilizado por deixar claro o trabalho que está realmente sendo feito.

Visualizar seu fluxo de trabalho gera uma série de benefícios, sendo os mais importantes:

- **Foco no “todo”:** Fica visível exatamente como o seu trabalho afeta as outras pessoas e vice-versa;
- **Transparência:** Todos sabem exatamente o que está acontecendo e nenhuma informação é escondida;
- **Identificação de desperdícios:** Surgirá naturalmente um questionamento da razão pela qual as coisas são feitas como são (mais sobre isto adiante).

Passo 2: Limitar o trabalho em progresso

Uma vez que consiga visualizar o fluxo de trabalho, mova para o próximo passo – limitar o trabalho em progresso (WIP).

Compreendendo o WIP

Para entender porque faz sentido limitar o WIP, precisamos passar pela Lei de Little: *Tempo de Ciclo = WIP / Produção por unidade de tempo* (fórmula adaptada para a terminologia do desenvolvimento de produtos)

O tempo de ciclo é o tempo que se leva para um item de trabalho passar pelo nosso sistema kanban; em outras palavras, o tempo que se leva para uma funcionalidade selecionada para implementação chegar à produção. O que significa "selecionada para implementação" dependente de contexto. Para alguns, é a entrada de um item no backlog; para outros, trata-se do momento em que um item é selecionado para detalhamento. Pode-se também distinguir entre "Lead Time" e

"Tempo de ciclo": o primeiro seria o tempo desde a chegada do item até sua entrega; o segundo é o tempo desde a seleção para implementação até sua entrega.

O WIP descreve o total de trabalho em progresso no sistema kanban. Quantos “pontos de história”/”histórias de usuário”/”itens de backlog” estão atualmente em progresso em seu sistema? Mais uma vez, isso vai depender do contexto. Alguns especialistas incluem todos os itens de backlog no WIP; outros consideram apenas os itens selecionados para implementação.

Produção (*throughput*) por unidade de tempo é a média do número de itens produzidos em um determinado período de tempo. No Scrum, é comumente chamado de velocidade.

Isso significa que, dado um sistema com 100 histórias de usuário em progresso (WIP) e uma produção de duas histórias por semana, o tempo médio de ciclo é calculado como 100/2 = 50 semanas ou quase um ano. Reduzir esse tempo para 25 semanas poderia ser feito dobrando-se a produção, ou reduzindo o número de histórias de usuário em progresso para 50. Na maioria dos casos, é mais fácil inicialmente reduzir o WIP do que dobrar a produção.

Como se pode perceber, limitar o WIP está intimamente ligado à redução do tempo de ciclo para aumentar a produtividade e minimizar a quantidade de trabalho em que investimos tempo e recursos (mas que ainda não gerou nenhum valor de negócio). Ciclos de feedback rápidos são também uma excelente maneira de minimizar riscos, dado que as decisões são validadas continuamente e os problemas de qualidade são expostos imediatamente. Este é um assunto explorado mais detalhadamente no livro "Custos da Qualidade" de Capers Jones (1980).

Então, como fazer? Não há como saber o número exato de quantos itens deveriam ser permitidos em cada estágio de nosso quadro, então escolhemos um número da melhor forma possível. Uma boa ideia é deixar que esse exercício seja guiado pelas políticas em que sua equipe gostaria de dar ênfase. Caso a equipe ache uma boa ideia incentivar o trabalho em pares, então se pode escolher um limite de 3 para uma equipe de seis pessoas.

Veja que isto é verdade apenas para colunas de atividade, como "desenvolvimento", "testes" etc. Já para colunas de buffer como "pronto para desenvolvimento", a regra geral é que, se essas colunas estiverem vazias uma vez por ano, o limite de WIP está muito grande (pois durante 364 dias o buffer foi maior que o necessário). E se estiver vazia todo dia, então o limite está pequeno demais. É comum ouvir que o limite universal de WIP é 5: se estiver em dúvida, 5 é provavelmente um bom número para se começar.

Visualizando limites de WIP

Há várias formas de se visualizar o limite de trabalho em progresso. As **Figuras 5** e **6** mostram duas formas comuns de se fazer isso. Na **Figura 5**, apenas um item pode ser colocado em cada espaço desenhado no quadro, disparando um sinal visual quando houver uma "permissão" para começar/puxar novas atividades (o espaço está vazio).

Na **Figura 6**, é mais fácil dividir o estágio da atividade em "em progresso" e "feito", já que o limite de WIP é simplesmente escrito no topo de cada coluna. Isso pode ajudar a compreender melhor o funcionamento do seu sistema de entregas, e é a maneira mais comum de exibir o limite de WIP

em TI. As pessoas que já trabalharam com manufatura Lean tendem a optar pela primeira versão (**Figura 5**), pois se parece mais com o sinal para puxar mais trabalho, identificado pelo cartão visual.

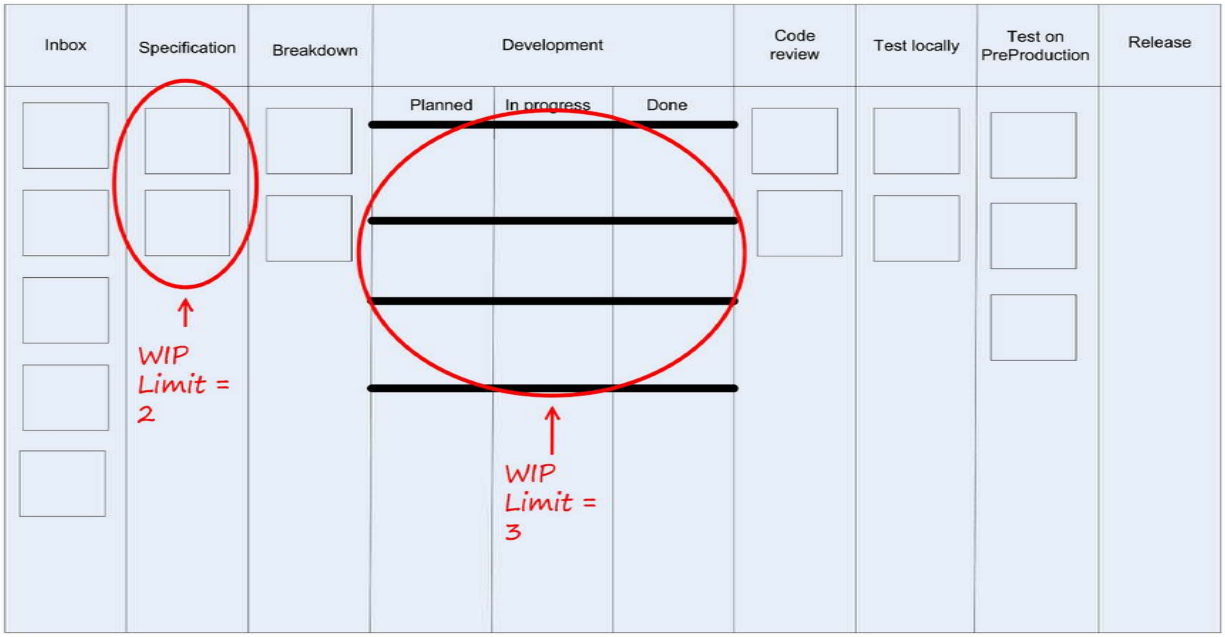

Figura 5. Limites de WIP visualizados com containers

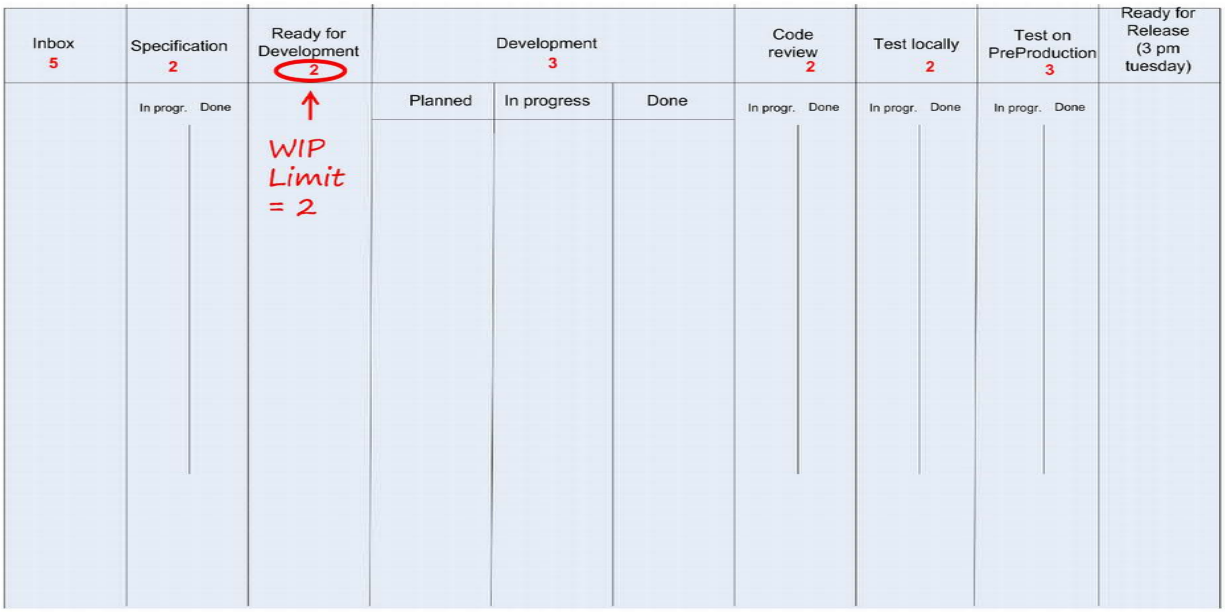

Figura 6. Limites de WIP visualizados usando números nos cabeçalhos de colunas

Identificação dos limites ideais para o WIP

Há várias "escolas de pensamento" sobre qual o tamanho ideal dos limites iniciais de trabalho em progresso (WIP), e estaria fora do escopo deste minilivro abordar esse tópico em detalhes. Uma das formas, entretanto, de se definir um limite inicial é observar o sistema e determinar valores com certa folga, para que o fluxo de trabalho atual continue desimpedido. Em seguida, ao se identificar o gargalo, começa-se a ajustar os limites, um por vez.

Uma abordagem mais radical seria determinar limites de trabalho em progresso muito pequenos; menores do que o seu sistema é capaz de lidar e colocar um buffer antes de cada estágio. Observa-se, então, onde o trabalho se acumula e gradualmente ajusta-se o limite do WIP até que o trabalho flua através do sistema. Ambas as abordagens requerem experiência e não se espera um acerto logo na primeira tentativa. Não há conclusão definitiva quanto à melhor abordagem, mas recomenda-se manter suas políticas de trabalho em mente, nas duas situações.

Em qualquer situação, é importante que se determine uma política explícita sobre como serão tomadas as decisões para alterar um limite. Para maximizar o aprendizado, o ideal é tomar essas decisões em conjunto com a equipe, garantindo que todos possam expor suas opiniões e entender as decisões. Não se trata apenas do fluxo, mas também de aprendizado.

Lembre-se sempre que os limites iniciais de trabalho em progresso são apenas bons palpites, dados em um momento em que se tem poucas informações. À medida que são adquiridas mais informações sobre o sistema, e encontradas melhores formas de se trabalhar, os limites devem ser ajustados. Se, depois de três meses, ainda se está trabalhando com os mesmos limites iniciais e os mesmos estágios, há boa chance de que tenha sido perdido o passo mais importante: o de melhoria contínua, que cobriremos mais tarde em mais detalhes.

Limites muito pequenos irão impedir o fluxo e deixar pessoas ociosas por tempo demais; ou então os limites serão simplesmente ignorados, sem discussões sérias. Limites muito soltos, por outro lado, aumentam o tempo de ciclo e mantêm o trabalho parado por mais tempo do que necessário.

O que se percebe rapidamente é que, com os limites de WIP estabelecidos, seu sistema só será capaz de trabalhar até a capacidade. Vai ser necessário finalizar uma atividade para conseguir permissão para começar outra. Apesar disso parecer trivial, trata-se de um conceito fundamental em um sistema puxado Lean, e uma ferramenta poderosa na busca por um sistema de entrega de software que seja eficaz, sustentável e previsível.

Pode-se imaginar o funcionamento de um sistema puxado como uma corrente de clipes de papel. Enquanto estiver puxando a corrente pela mesa por uma das pontas, os clipes seguem em uma fila organizada, mas caso se decida empurrá-los, há o caos (veja a **Figura 7**).

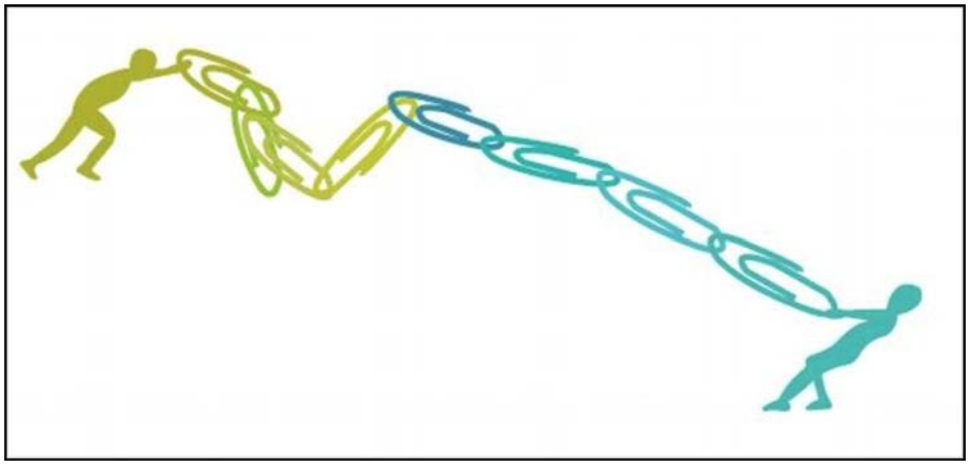

Figura 7. Pull contra Push: a metáfora da cadeia de clips

No início, haverá desconforto por não se poder começar algo novo, mesmo quando isso parece ser a "coisa certa" a fazer. Esse desconforto é sinal de que existe um impedimento no fluxo. O mais importante é não ignorar o problema e aproveitar a oportunidade para realizar melhorias.

Nos momentos de discussão sobre o WIP, é comum se focar demais no número de itens em progresso. Mas não devemos esquecer que o tamanho dos itens também é importante. Itens grandes bloqueiam recursos por longos períodos e perturbam o fluxo – ao passo que itens menores fluem muito mais rapidamente pelo sistema de entregas, fornecendo feedbacks mais rápidos. Porém, analisar os itens e quebrá-los em partes pequenas, em um conjunto "mínimo de funcionalidades vendáveis" (*minimal marketable features* – MMF) é uma tarefa que requer imaginação, habilidade e experiência.

Passo 3: Estabelecer políticas explícitas para garantia de qualidade

Caso esteja familiarizado com a filosofia Lean, o leitor pode ter ouvido falar do termo "Qualidade embutida" e pode se questionar porque a qualidade é tão importante. A resposta não está em encontrar e corrigir defeitos, mas no fato de problemas de qualidade serem muito mais caros do que se pensa.

No Kanban, concentra-se em criar políticas explícitas que otimizem a qualidade e deem consistência ao sistema de entrega de software – e usamos essas políticas como base para a melhoria contínua.

Entendendo a qualidade

Uma interface de baixa qualidade que impede a conclusão de uma tarefa, ou um defeito que atrapalha o fluxo de trabalho – ambos são problemas de qualidade que estressam o sistema Kanban, gerando ciclos de desperdício. Isso é conhecido em sistemas Lean como "demanda de falha", e descreve todas as atividades e o retrabalho ligados à baixa qualidade no projeto do produto.

Às vezes é aceitável lançar um produto com baixa qualidade para se obter mais feedback. Pode haver também vantagens financeiras em permitir que se encontre o defeito em produção, ao invés de em desenvolvimento – pois o custo e o tempo poderiam ser maiores sem o feedback dos usuários. Na maioria dos casos, porém, a baixa qualidade se trata apenas de um sintoma de processo imaturo.

Por que problemas de qualidade são tão caros? Verifiquemos essa questão em cenários comuns no desenvolvimento de software.

Um usuário (vamos chamá-lo de João) não consegue concluir sua tarefa devido a um defeito em nossa aplicação. João abre um chamado relatando o bug ou liga para o suporte de primeiro nível. Ele não sabe muito sobre o trabalho necessário para um desenvolvedor investigar o problema, ou talvez o suporte de primeiro nível não conheça a aplicação tão bem quanto deveria.

Em ambos os casos, informações erradas ou inadequadas são passadas ao desenvolvedor, que acaba resolvendo um problema diferente (que pode até não ser um problema e gerar um novo bug); ou o desenvolvedor pode simplesmente desistir. Até que o defeito seja corrigido, o usuário insiste nesse processo. Só que não foi apenas João que ficou impossibilitado de completar o que estava fazendo. Informações inconsistentes foram persistidas na base de dados, que agora requer o uso de um script de SQL complicado para reverter os dados a um estado consistente.

Não é incomum que situações como essa ocorram e de que 100 a mil vezes mais tempo e recursos sejam gastos com a solução – comparando-se com o custo de evitar a introdução do defeito.

Além disso, problemas de qualidade levam à mudança constante de contexto e ao apagamento de incêndios. Naturalmente colocamos de lado o trabalho que estamos fazendo para corrigir um defeito sério ou um problema de usabilidade no ambiente de produção. E quando, após metade do dia, o problema é finalmente resolvido, não conseguimos lembrar dos detalhes do outro problema complicado em que estávamos trabalhando – e temos que gastar meia hora apenas para se reambientar.

Por essas razões, o Lean dá grande ênfase na adoção de medidas para prevenir que falhas aconteçam no sistema de entrega, usando o Poka Yoke (pronuncia-se "Pocá loquê"). Em sistemas de produção, o Poka Yoke é feito utilizando-se padrões e checklists que devem ser seguidos para se completar uma tarefa. Por exemplo, até células fotossensíveis são usadas para registrar se uma chave de fenda específica está sendo utilizada a quantidade correta de vezes, ou se todas as partes necessárias foram removidas de uma pilha.

Isso pode parecer um ambiente desumano para se trabalhar, mas na realidade os trabalhadores de sistemas de produção Lean não se enxergam como robôs seguindo cegamente padrões e checklists. Eles se veem como especialistas que estão constantemente tentando melhorar o sistema no qual estão trabalhando, ao sugerirem aperfeiçoamentos e novas ideias. No livro "Toyota: A Fórmula da Inovação, 2006", de Matthew E. May, o autor refere-se a este fato como a principal razão pela qual a Toyota ainda consegue implementar um milhão de novas melhorias todos os anos.

Visualização das políticas

Mas como traduzimos, essas ideias para o desenvolvimento de software? Já temos boa parte do trabalho feito, pois já estamos visualizando o trabalho com o quadro Kanban e já limitamos o trabalho em progresso. Tudo o que se precisa fazer agora é adicionar no quadro as políticas sendo usadas atualmente para assegurar a qualidade e a consistência. Ao fazer isso, seu quadro pode ficar parecido com o da **Figura 8**.

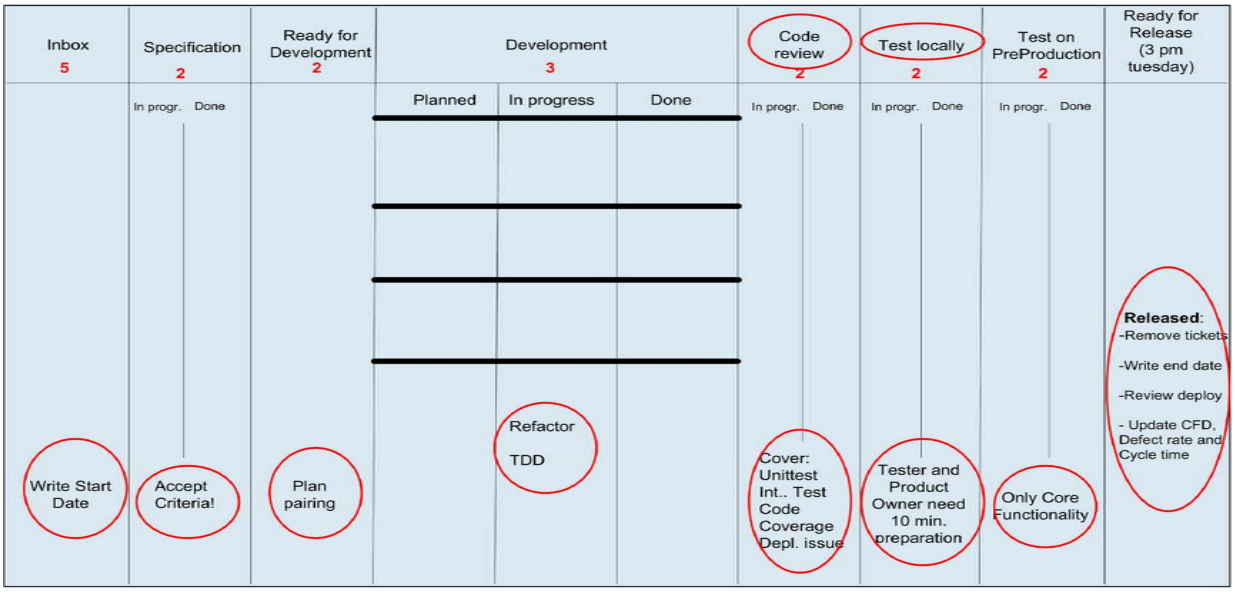

Figura 8. Políticas explícitas visualizadas no quadro

Repare que estágios inteiros podem ter políticas para garantia de qualidade, e que políticas também podem servir para a garantia da consistência e de qualidade do próprio processo (ex.: medindo o tempo de ciclo e o índice de defeitos). Tudo isso nos leva a outra prática fundamental do Kanban: tornar as políticas explícitas.

Várias equipes já atingiram níveis extraordinários nas práticas voltadas à prevenção de falhas nos últimos anos, e implementaram sistemas que eram impensáveis apenas alguns anos atrás. O movimento de Lean Startup tem mostrado que é possível lançar software em produção diversas vezes ao dia sem indisponibilidades ou altos índices de defeitos. Isto só funciona porque foram adotadas práticas como testes unitários, testes de integração, e testes de regressão e de desempenho, que ajudam a validar o código – assim como indicadores de desempenho que sinalizam sempre quando o sistema não se comportar como esperado, retornando-o automaticamente para sua última versão estável. Isso torna quase impossível introduzir quantidades grandes de erros.

Tenha em mente que você nunca deve se sentir dominado por suas próprias políticas e checklists. Você não é um autômato; é um profissional do conhecimento, observando constantemente e tentando melhorar o sistema em que está trabalhando. Comece adicionando os procedimentos que está utilizando no momento e adicione, remova ou altere esses procedimentos quando perceber maneiras novas e melhores de garantir a qualidade. Cada defeito é uma oportunidade de pensar sobre como evitar que outro defeito apareça.

No começo, pode parecer penoso analisar e aprender com cada defeito que surgir. Mas com o tempo, conforme a qualidade melhora, o processo se tornará mais natural. Esse é um preço que vale a pena pagar, e é conhecido como "Tolerância zero" nos círculos de testes ágeis.

É comum que essa ideia seja interpretada de forma incorreta, como "não podemos gastar o tempo que gostaríamos criando mecanismos para tornar os produtos à prova de falhas". O que vale no final é que teremos zero defeitos. "Tolerância zero" não significa que nunca mais possam existir defeitos, e sim que devemos conscientemente refletir sobre cada defeito e buscar a perfeição.

Ao implementar o Kanban, é essencial que todos se comprometam com as políticas adotadas (inicialmente essas políticas descrevem apenas o processo atual), e que aceitem que é necessária uma decisão da equipe para mudá-las. Quando alguém decide infringir as políticas por conta própria, sem o envolvimento da equipe, o processo rapidamente degenera. Lembre-se: quando você sentir a necessidade de desrespeitar uma política, será geralmente por uma boa razão – e este é um bom momento para iniciar as discussões necessárias. Em relação às políticas, "altere mas não quebre" é a frase que sempre repito quando forneço treinamentos em Kanban.

Passo 4: Ajustar cadências

Quando você tiver conseguido visualizar seu fluxo, limitado o trabalho em progresso e estabelecido políticas para assegurar a qualidade, a próxima coisa que se deve avaliar são as cadências ou ritmos de trabalho. Em um sistema de entrega de software há diversas atividades que se beneficiam de cadências regulares – e encontrar a cadência exata para cada tipo de atividade é essencial para melhorar o fluxo.

Repare que, em um sistema kanban, não somos obrigados a sincronizar tudo utilizando um mínimo denominador comum. Podemos ajustar as cadências de cada atividade para níveis ótimos individualmente. Cadências típicas que precisamos considerar são a de planejamento (entrada) e a de entrega (saída). Há muitas outras cadências, como a de revisão/retrospectiva e a de garantia de qualidade. Vamos focar em planejamento e entrega, por enquanto.

Entendendo cadências

Encontrar a cadência ideal de entrega é uma das tarefas mais importantes no Desenvolvimento Lean de Produtos, pois ajuda a otimizar os ciclos de feedback, reduzir os riscos e otimizar o seu processo de entrega. O feedback rápido faz parte do núcleo do Agile e do Lean. Quando faço o *coaching* de equipes, continuamente friso que o trabalho que ainda não entregou valor deve ser visto apenas como uma hipótese que precisamos validar o mais cedo possível.

Apesar de entregar cada funcionalidade diretamente para produção ser a solução ótima (considerando que se tenha um sistema otimizado para lidar com isso), na realidade, a maioria dos projetos trabalham com duas cadências de entrega.

- Uma cadência em que o código é disponibilizado em um ambiente de pré-produção para se obter feedback inicial (cadência de lançamentos internos);
- Outra cadência na qual a nova versão é lançada em ambiente de produção (cadência de lançamentos externos).

Especialmente em projetos novos, estas duas cadências podem estar muito distantes uma da outra. Pode levar até cinco meses para que o sistema tenha funcionalidades suficientes para ser colocado em produção. Ao mesmo tempo, o código pode estar sendo lançado e testado todos os dias em um ambiente de pré-produção.

O que é importante ao escolher uma cadência, tanto para as entregas internas quanto para as externas, é estar consciente dos impactos econômicos dessa escolha. Sempre há custos transacionais associados com a entrega (por exemplo, o custo de mover sua versão de um ambiente para outro); e sempre há um custo de espera (*holding cost*). O ponto ótimo entre esses dois custos determina a cadência ideal. Veja esse processo visualmente, na **Figura 9**, extraída do livro de Don Reinertsen sobre Desenvolvimento Lean de Produtos (utilizada com permissão).

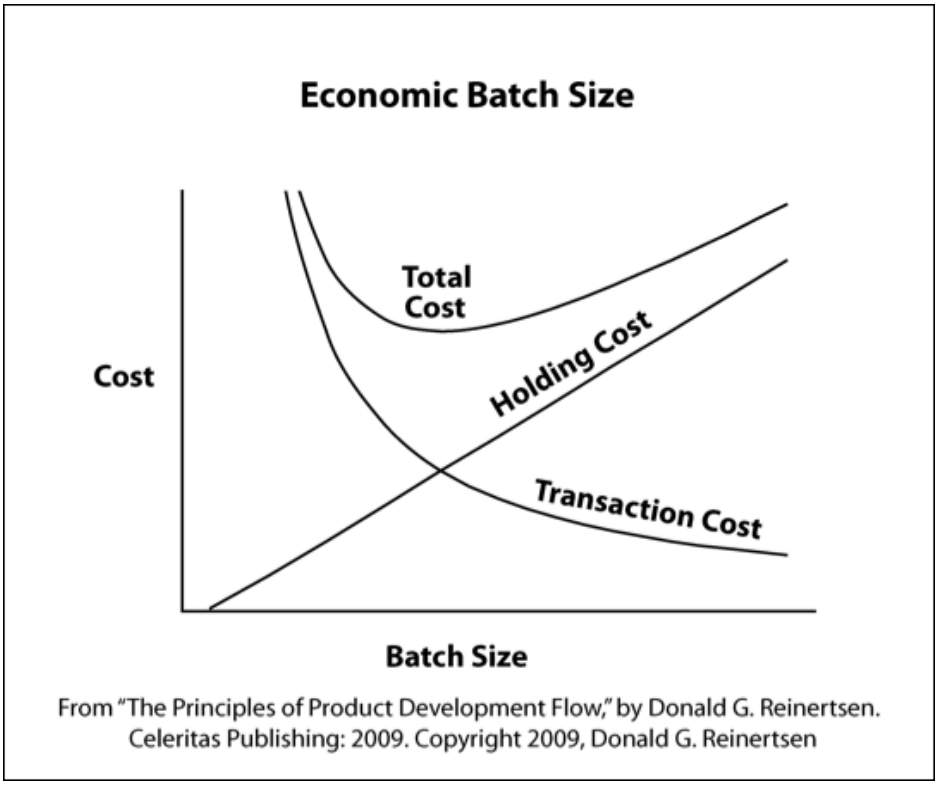

Figura 9. Otimização do tamanho dos lotes

Quanto mais funcionalidades são incluídas em uma entrega, mais barato será o custo por funcionalidade (menor custo transacional). Em contrapartida o custo de espera é maior, uma vez que cada funcionalidade terá de esperar mais tempo para ser lançada. Isso resulta em perda de valor de negócio, feedback desatualizado, decisões pouco embasadas e menor envolvimento com o usuário.

Custos de espera são um pouco diferentes para lançamentos externos e internos. Como um lançamento interno não expõe nenhum valor real de negócio para os clientes, os custos de espera representam se restringem a feedback desatualizado, decisões desinformadas e motivação do usuário/negócio diminuída devido ao baixo envolvimento. Mas qualquer pessoa que tenha trabalhado em um ambiente de negócios real reconhecerá que esses problemas podem ser tão prejudiciais quanto a perda de receita.

O gráfico da **Figura 9**, já citado, mostra uma curva do tipo "u" para a linha de custo total. A curva apresenta uma parte inferior bastante suave, significando que não é necessário encontrar o ponto ótimo exato para a cadência de entregas. Mesmo com um erro de 10 a 15%, haverá bons resultados.

Estabelecendo as cadências ideais

Muitos projetos, infelizmente, nem sequer consideram os aspectos acima com profundidade. Diversas equipes ágeis maduras são capazes de fazer entregas em ambientes de pré-produção com um clique de um botão – mas ainda assim demoram três semanas para receber o feedback dos usuários sobre uma funcionalidade. Quando os custos de transação estão na casa dos poucos reais, deve-se considerar seriamente o uso de lotes muito pequenos. Algumas vezes isso pode ser problemático, já que os usuários podem não estar disponíveis, mas na maioria dos casos o que acontece é que a possibilidade nem sequer tinha sido considerada.

Outro ponto importante que a Toyota nos ensinou é que os custos de transação não são fixos.

O movimento de implantação contínua (*continuous deployment*) tem mostrado que é possível lançar versões confiáveis para produção 50 vezes ao dia, para sistemas lidando com milhões de pessoas. Isto só pode ser atingido com um procedimento completamente automatizado de lançamento, e todo um conjunto de testes unitários, testes de integração e de regressão. É esse investimento que permite trabalhar em lotes de poucas linhas de código.

As mesmas considerações devem ser levadas em conta para se chegar a uma cadência de planejamento adequada. Quando o tempo entre os encontros de planejamento se torna mais longo, mais coisas precisam ser planejadas em um grande lote. Isso resulta em mais esforço de design em progresso e menos decisões fundamentadas – devido a um tempo de ciclo mais longo. Por outro lado, realizar encontros diários pode se mostrar custoso demais e aumentar o custo transacional.

Em alguns casos, equipes que utilizam Kanban escolhem realizar o planejamento sob demanda. Isso pode ser feito com o envio de um email para os interessados, convidando-os para um encontro, ou uma conferência. Isso sempre que houver espaço para que a fila de entrada seja preenchida com, por exemplo, os três itens de maior prioridade. Normalmente o planejamento "sob demanda" é utilizado por equipes mais avançadas e requer experiência na gerência dos fluxos. Portanto, pense bem antes de começar com essa estratégia.

Passo 5: Medir o fluxo

Medir o progresso de projetos é, infelizmente, um dos aspectos mais mal compreendidos e mal aplicados no desenvolvimento de software. É comum ver métricas sendo usadas para tornar os gerentes de projeto responsáveis por aspectos sobre os quais não têm controle; ou ainda como critérios de sucesso fixos, estabelecidos quando nem se conhecia o sistema a ser desenvolvido.

Entendendo métricas

Em métricas, uma regra básica a ser lembrada é que seu sistema de entrega de software tem capacidade limitada. Quando o sistema é pressionado além da capacidade, haverá queda na qualidade, ritmo insustentável, custos de manutenção mais altos, ou tudo isso ao mesmo tempo. Ainda assim, sempre vemos gerentes de projeto “quase orgulhosos” de terem feito suas equipes

trabalharem além do horário por três meses, ou que, por algum esforço heroico, conseguiram controlar as coisas no último momento, depois de semanas de caos.

Apesar de ser positivo comemorar grandes realizações, os projetos de desenvolvimento de software não precisam de heróis, e sim de gente capaz de fazer entregas com transparência e em ritmo sustentável. Tudo a mais é simplesmente muito caro. Aqui se aplica uma frase de Kent Beck: "se um problema precisa de mais trabalho que uma semana de horas extras, então o problema não deveria ser resolvido com horas extras".

Pode-se, claro, aumentar a capacidade no decorrer do tempo, contratando mais pessoas, otimizando processos, ou os dois (mas cuidado ao fazer ajustes visando resultados de curto prazo). Outra boa regra a ser usada é: "seu sistema nunca terá mais capacidade do que a capacidade que já *provou* ter". Seguindo essa regra, pode-se evitar os antipadrões de excesso de otimismo e comparações com o sucesso dos outros. Começando um projeto do zero, sua capacidade será apenas um palpite, baseado em desempenhos anteriores. O importante é medir o progresso desde o começo para validar todas as suposições. Veremos mais sobre esse assunto no Passo 8.

Então, pense em seu plano como uma ferramenta que serve como guia, não como critério de sucesso. E meça seu fluxo para determinar se você ainda está no caminho certo. O objetivo é obter um sistema de entrega de software estável e previsível, para que se possa tomar decisões bem fundamentadas sobre prazos, dependências, pessoal, escopo e custos.

O que medir?

Como medir o fluxo? Antes de escolher a forma de medir, deve-se sempre se perguntar o que será feito com a informação obtida. Se você não pretende mudar nada com uma métrica, é provável que se trate de algo que não se deveria estar sendo medido. Para começar, sugiro quatro técnicas: Diagramas de Fluxo Cumulativo, Tempo de Ciclo, Índices de Defeitos e Itens Bloqueados.

Diagramas de fluxo cumulativo (CFD)

Diagramas de fluxo cumulativo parecem estar substituindo os gráficos de burndown (comuns no Scrum), nas equipes e organizações ágeis mais maduras. São fáceis de atualizar e fornecem uma melhor perspectiva do estado do projeto.

Os CFDs mostram essencialmente um retrato da quantidade de trabalho em seu sistema, para cada estágio de sua cadeia de valor, no decorrer do tempo. O gráfico inclui o mesmo tipo de informações que gráficos de burndown tradicionais, mas vai além (veja um exemplo na **Figura 10**).

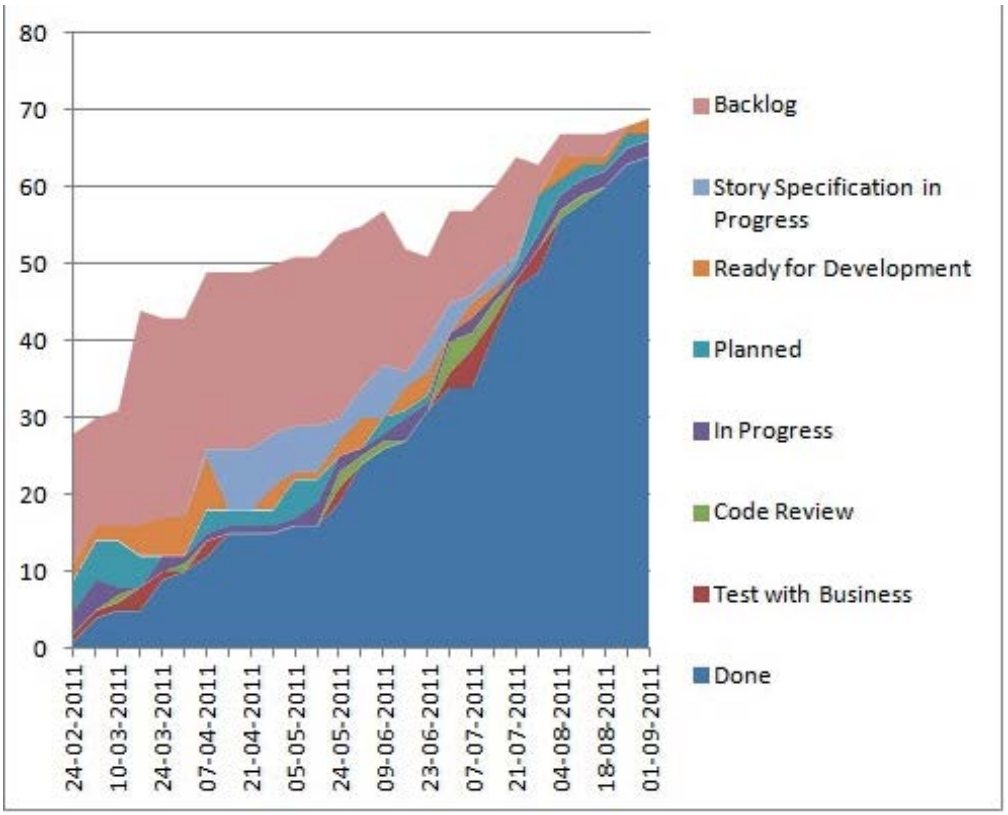

Figura 10. Exemplo de diagrama de fluxo cumulativo

Interpretando o CFD

A área que mostra os itens "done" (prontos) representa a velocidade no decorrer do tempo. A área entre as linhas “done” e "backlog" é o trabalho em progresso (WIP).

- Se a distância entre duas linhas na área do trabalho em progresso (WIP) aumentar, isso pode ser um sinal de gargalos;
- Se a linha de backlog estiver mais inclinada que a de done, há sinal claro de que se está adicionando mais trabalho ao sistema se comparado à sua capacidade atual de entrega;
- Projetar onde as linhas que representam os itens de backlog e de done irão se encontrar é a melhor maneira de estimar a data final de entrega;
- O tempo médio de ciclo e quantidade de itens na fila também podem ser extraídos do diagrama.

Aprender a interpretar um diagrama de fluxo cumulativo (CFD) é fácil. A **Figura 11**, do livro de Don Reinertsen, mostra uma excelente representação visual. Observe que a área em preto representa o trabalho em progresso (WIP).

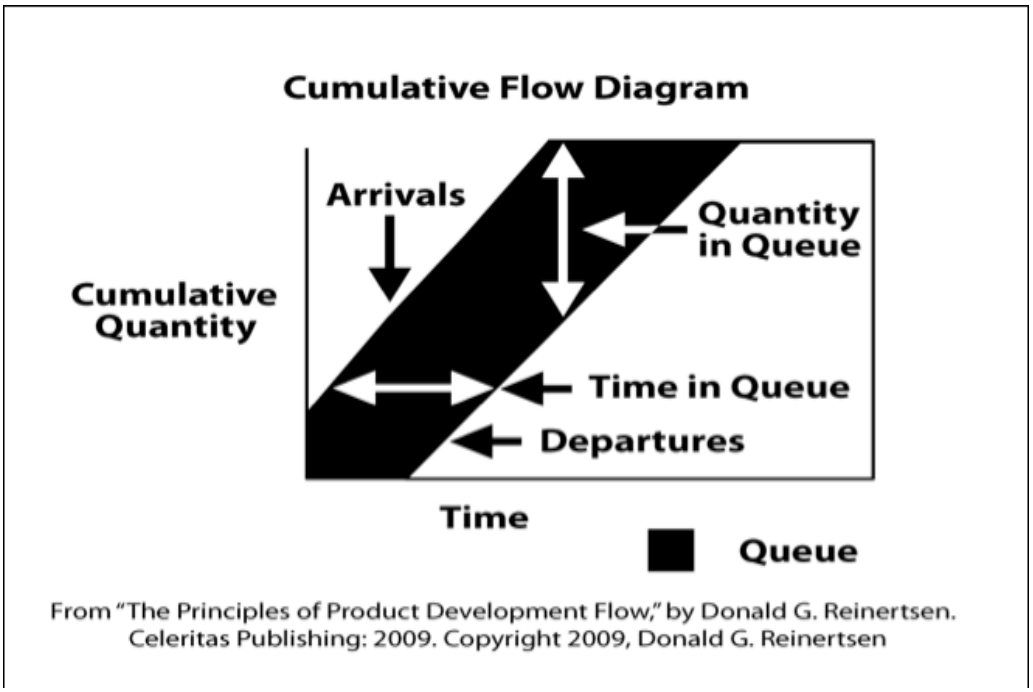

Figura 11. Como ler um Diagrama de Fluxo Cumulativo

Tempo de ciclo

Apesar de seu Diagrama de Fluxo Cumulativo mostrar o tempo médio de ciclo, medir os tempos de ciclo individuais vale muito a pena em termos de previsibilidade.

Médias podem ser enganosas. Uma representação visual trará informações mais detalhadas sobre a confiabilidade do seu sistema, além da oportunidade de atender demandas dos clientes com mais precisão (veremos mais sobre isso no Passo 9).

Medir o tempo de ciclo é ainda mais fácil que atualizar o gráfico CFD. Tudo que se tem de fazer é registrar o dia em que se começou a trabalhar em um item (lembre-se de tornar esta política explícita também). Quando o trabalho é concluído, basta marcar o número de dias gastos para completar o item. Assim, seu diagrama deverá ficar parecido com o mostrado na **Figura 12**.

Como cada "passo" no eixo X representa um item de trabalho completo, as equipes frequentemente escolhem deixar o eixo sem unidades.

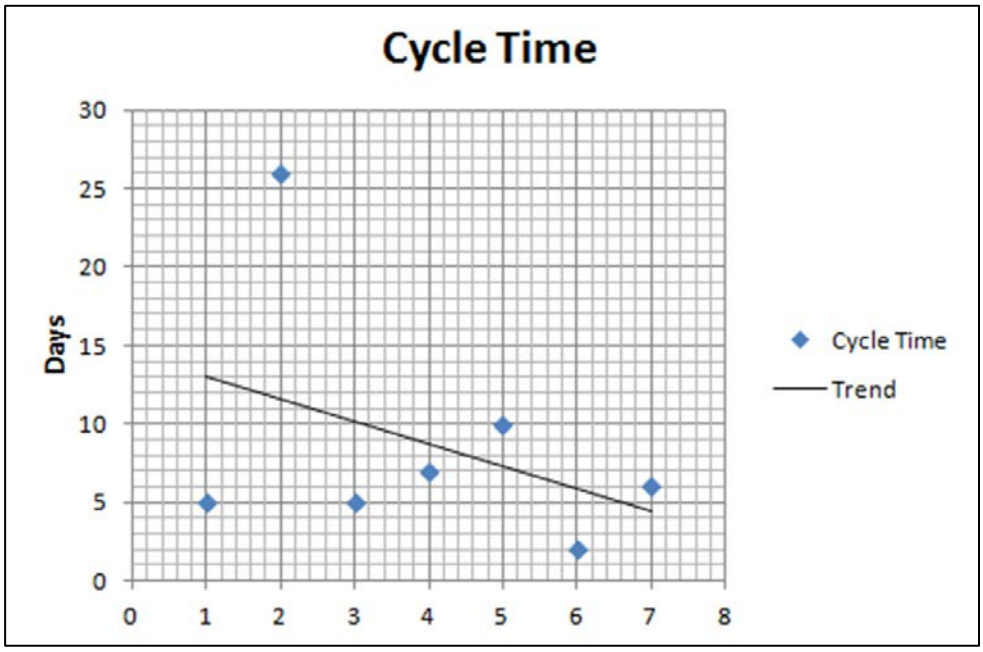

Figura 12. Exemplo de diagrama de tempo de ciclo

Apesar de simples, o diagrama de tempo de ciclo pode trazer muitas informações e responder várias perguntas sobre como seu sistema está funcionando:

- Os números indicam algum nível de consistência ou os números são incongruentes?
- Como está a tendência? Aponta na direção correta?
- O diagrama permite investigar os números muito discrepantes (positivos e negativos)
- O diagrama também permite verificar as consequências das decisões (tarefas grandes, resolução de incidentes, problemas de qualidade etc.)

Se 90% dos itens de trabalho levam menos de uma semana, é possível se comprometer com o cliente dizendo que ele/ela pode esperar que 9 em cada 10 itens sejam feitos neste prazo.

É importante lembrar que o tempo de ciclo é um indicador tardio, o que significa que veremos apenas os problemas depois que ocorrerem. Ou seja, quando é tarde demais para fazer algo a respeito. Portanto, devemos usar diagramas de tempo de ciclo em conjunto com um CFD, por exemplo, para poder agir proativamente.

Índice de defeitos

Como mencionado, problemas de qualidade são extremamente caros, portanto é importante ficar atento a eles. Medir o índice e o número total de defeitos em seu sistema é uma maneira fácil de evitar que problemas de qualidade fujam do controle.

É surpreendente ver tão poucas organizações tratarem índices de defeitos como um indicador de chave de desempenho (*Key Performance Indicator* – KPI). Indicadores de desempenho nos mostram informações valiosas sobre o estado do projeto, por exemplo:

- Por que o número de novos defeitos tem aumentado? Houve relaxamento de alguma política de Qualidade?
- Como o alto índice de defeitos na semana 20 afetou o tempo de ciclo?
- Qual o impacto no diagrama de fluxo cumulativo quando o número de defeitos aumentou?

Tome sempre cuidado para não tirar muitas conclusões com base em conjuntos individuais de informações. Uma semana ruim pode ser apenas uma coincidência. E não deixe de prestar atenção às tendências para saber se está tudo indo na direção correta. A **Figura 13** mostra um exemplo de um diagrama de índice de defeitos.

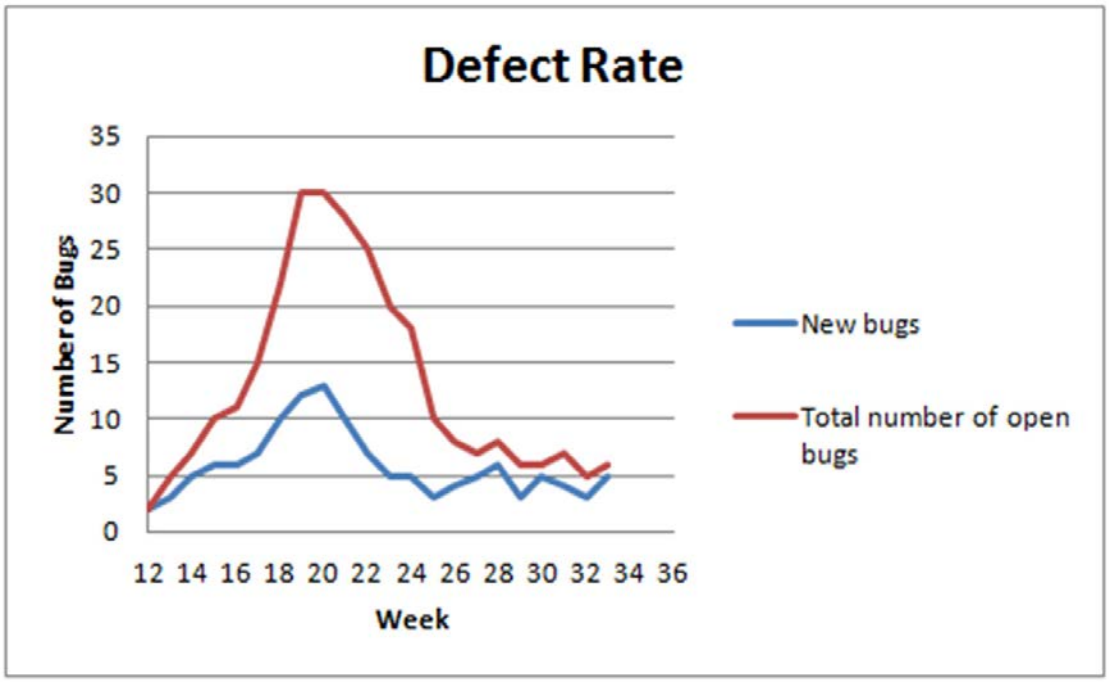

Figura 13. Exemplo de diagrama de taxa de defeitos

Manter o número total de defeitos abaixo de 20 é uma boa política para a maioria dos projetos. Se a lista se tornar maior que vinte itens, fica difícil administrá-la e se gasta muito tempo dando manutenção no backlog de defeitos – conferindo entradas duplicadas, defeitos desatualizados e itens corrigidos. E quando o número é alto, as pessoas parecem ficar mais preocupadas, demandando mais relatórios e métricas para acompanhar a lista de defeitos. Começam a surgir quadros gerenciais sobre defeitos e reuniões semanais de acompanhamento, que roubam tempo valioso. Mesmo defeitos cosméticos requerem atenção e exigem tempo adicional para administrar.

É comum que se acabe tentando simplesmente alterar a categorização da severidade dos defeitos para se livrar do problema, mas é uma solução inadequada, devido às razões discutidas acima.

Itens bloqueados

Espero que você já esteja convencido da importância do fluxo para que os sistemas se comportem com previsibilidade, e para que os processos individuais operem com eficácia. A maioria das pessoas, independentemente do contexto ser Agile ou não, já viu itens ficarem bloqueados por longos períodos e por várias razões. Apesar de o gráfico CFD e o diagrama de tempo de ciclo

mostrarem esse acúmulo de atividades paradas, a equipe pode achar vantajoso medir e visualizar explicitamente sua habilidade de lidar com esses problemas.

Algumas empresas utilizam o número de itens bloqueados e a habilidade de resolvê-los como indicador chave na medição do desempenho. Reconhecem que impedimentos têm sérios efeitos de longo prazo no sistema de entregas, e que a habilidade de resolvê-los rapidamente é bom indicativo do desempenho e da eficácia da equipe. Itens bloqueados devem sempre estar visíveis no quadro. E medi-los no decorrer do tempo é uma boa maneira de saber se a equipe está caminhando na direção certa. A **Figura 14** mostra um exemplo de diagrama de itens bloqueados.

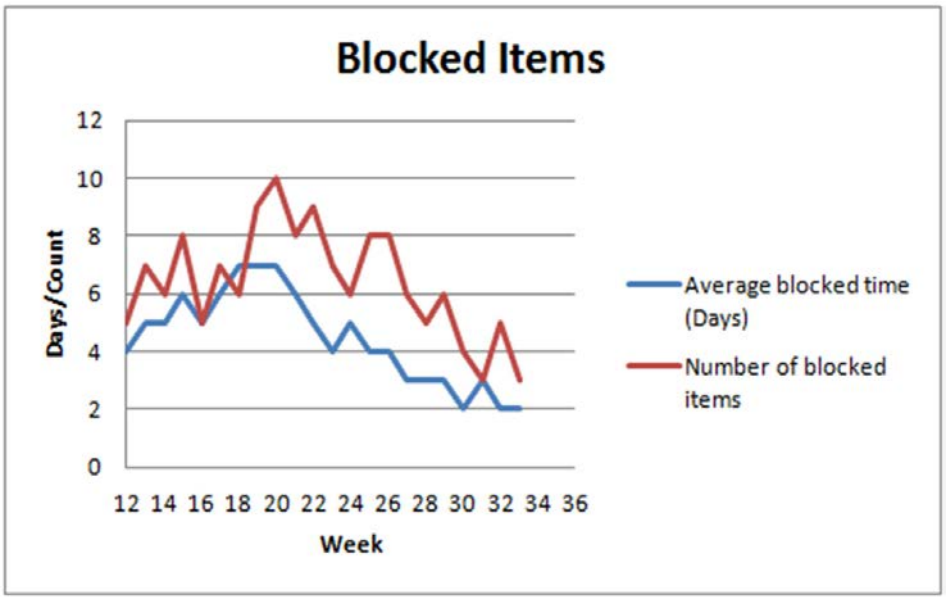

Figura 14. Exemplo de diagrama de itens bloqueados

A forma mais comum de visualizar itens bloqueados no quadro é simplesmente colar um post-it colorido a uma funcionalidade, com o problema que está causando o bloqueio e a data em que surgiu. A **Figura 15** mostra um exemplo de quadro em que é marcado um item bloqueado.

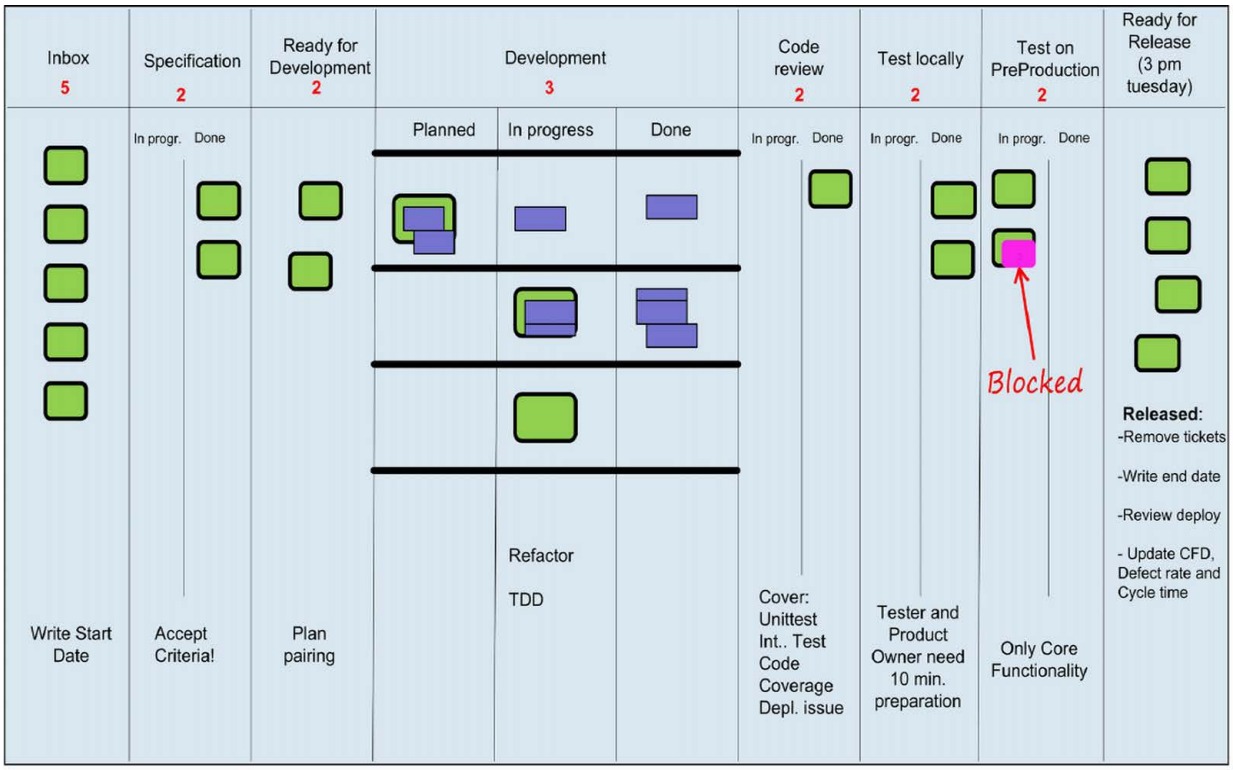

Figura 15. Visualização de item bloqueado com marcador

Evite separar um local específico do quadro para os itens bloqueados. Como esse local não faz parte do fluxo atual de trabalho, há a tendência de a área não ganhar a devida importância (até que alguém apareça berrando, querendo saber porque um item não foi concluído ainda!). Usar um espaço específico do quadro também parece fazer com que haja menos interesse em resolver os problemas, e mais interesse em declarar que outras partes são os responsáveis pela resolução.

Geralmente, quatro diagramas próximos ao quadro constituem o limite de quantidade de informação que a maioria das pessoas é capaz de absorver, antes que percam interesse ou atenção. Mas é importante que essas informações estejam visíveis; mantê-las escondidas em uma planilha em algum sistema eletrônico raramente chamará a atenção devida. A **Figura 16** mostra os quatro gráficos discutidos nesse capítulo, exibidos logo acima do quadro da equipe.

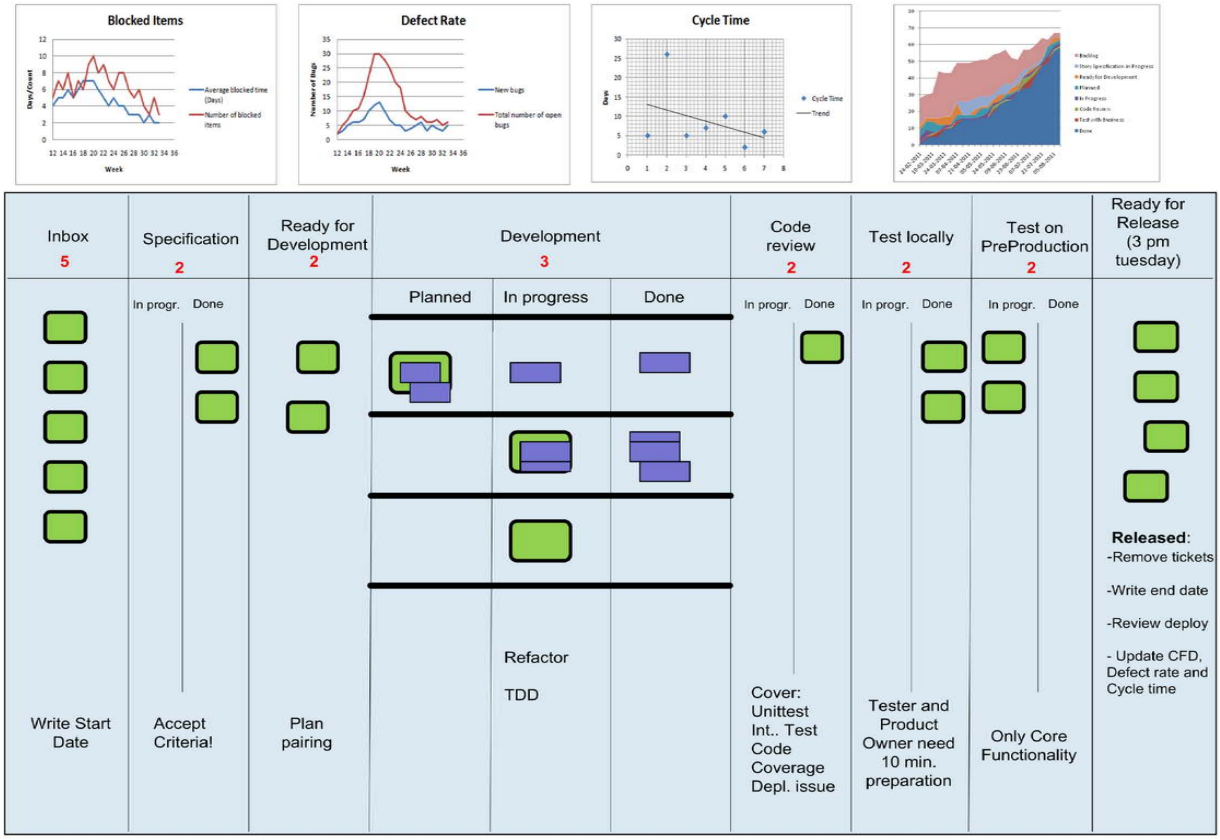

Figura 16. Métricas de fluxo visualizadas no topo do quadro

Passo 6: Priorização

Pode ser surpreendente estarmos no sexto passo e ainda não termos começado a priorizar. Mas, como diz David J. Anderson, caso não se tenha um sistema de entrega de software funcionando, com um fluxo contínuo e entregas seguras e de qualidade, a priorização terá pouca importância.

Quando chega o momento de priorizar, como fazer esse trabalho da melhor forma possível? Focaremos aqui na priorização de um tipo de trabalho específico, as histórias de usuário. No Passo 7, veremos como tipos de trabalho diferentes devem ser tratados.

Custo de atraso

Um conceito importante usado para priorização é o Custo de Atraso (COD – Cost of Delay). O Custo de Atraso representa a receita ou a economia perdidas com a escolha de *não* se trabalhar em um determinado item (uma história de usuário, por exemplo). A mais alta prioridade deve ser o item com o maior custo de atraso. O custo de atraso geralmente associado ao Custo de Implementação (COI), a prazos e a outros fatores.

Não está no escopo deste texto explicar em detalhes o Custo de Atraso, mas aqui só precisamos conhecer o conceito de custo de oportunidade e saber que toda vez que se escolhe trabalhar em algum item, escolhe-se também bloquear algum outro. Calcular exatamente o COD é quase sempre impossível no desenvolvimento de produtos; então geralmente será necessário seguir

bons palpites. Aprender a associar um valor econômico ao trabalho desempenhado é um exercício de amadurecimento para projetos e organizações.

Visualizando prioridades

Para garantir que o item certo será escolhido, a fila de entrada deve conter sempre os itens ordenados por prioridade. Essa regra é aplicável tanto ao planejar um lote de trabalho para o próximo Sprint no Scrum, quanto no trabalho com um fluxo contínuo. Neste último caso, os itens de maior prioridade são puxados continuamente – sempre que houver uma autorização de trabalho. A **Figura 17** mostra um exemplo.

Figura 17. Políticas de priorização visualizadas no quadro

Outros fatores importantes para a priorização, que também devem ser incluídos na classificação final, são:

- Riscos e incertezas: obtenha informações o mais cedo possível, para apoiar decisões de alto risco e de alto impacto;
- Necessidades básicas: infraestrutura do projeto etc.;
- Tamanho equilibrado: misture histórias de diferentes tamanhos para manter um fluxo constante;
- Tipos de história equilibrados: misture histórias funcionais e não-funcionais para garantir um fluxo constante de valor;
- Dependências: trate dependências proativamente, para que o trabalho não fique impedido.

Ao trabalhar com um backlog tradicional, ou uma especificação de requisitos como a do modelo Cascata, contendo 50 itens ou mais, pode-se questionar quantos dos itens irão ser visualizados no quadro. Não há regra geral. Algumas equipes acreditam que é melhor visualizar toda a fila de entrada; outras mantêm uma lista separadamente, em um local físico ou ferramenta, e gradualmente vão puxando alguns itens mais importantes para o quadro.

Gosto de usar o iceberg como metáfora para descrever essa política. Apenas o topo do iceberg fica visível acima d'água, mas quando removemos o topo (ou seja, puxamos os itens para o sistema), o gelo abaixo emerge para formar um novo topo. Ao mesmo tempo, manter um limite explícito de WIP na fila de entrada é uma boa ideia, para que o trabalho em progresso não saia do controle.

Geralmente comparo um backlog com o piso superior não utilizado de uma casa. Caso se coloque lá tudo o que se tem medo de jogar fora, há pouca chance de encontrar o que realmente se precisa quando necessitar. Mantenha o backlog enxuto e certifique-se que ele não esteja crescendo demais e saindo do controle.

Passo 7: Identificação de classes de serviço

Poucos questionariam a necessidade de tratamento especial de um problema que resulta em 10 mil usuários sem acesso ao sistema, causando perda de $10o mil por hora. Mas muitas vezes a importância e prioridade de um item não são tão claras. Como ter certeza que foi escolhida a melhor maneira de lidar com diferentes tipos de trabalho?

No Kanban, itens de trabalho são tratados de acordo com suas características. Esse tratamento diferenciado recebe o nome de "classe de serviço".

Identificando tipos de trabalho

Um bom começo é identificar os diferentes tipos de trabalho. Por exemplo, quase todos os projetos de software possuem algum tipo de requisito, como histórias de usuário e casos de uso; também possuem defeitos. Todos esses seriam tipos de trabalho. E se pode quebrá-los em subtipos. Veja alguns exemplos típicos de tipos de trabalho:

- Histórias de Usuário (Pequenas, Médias e Grandes);
- Defeitos (Cosméticos, Críticos e Impeditivos);
- Relatórios Manuais;
- Edições Textuais;
- Tarefas de Suporte;
- Instalação.

Definindo classes de serviço

O próximo passo depois de definir os diferentes tipos de trabalho é determinar como serão tratados em seu sistema. Cada forma distinta de lidar com tipos diferentes é uma Classe de Serviço. Abaixo definimos quatro classes de serviço:

Classe Padrão:

- Custo adicional: 0
- Tipos de trabalho: Defeitos cosméticos, histórias de usuário
- Tratamento especial: Nenhum

Classe Prioritária:

- Custo adicional: $500
- Tipos de trabalho: Defeitos críticos, histórias de usuário de alta prioridade
- Tratamento especial: Ganha prioridade em todos os estágios

Classe de Prazo Fixo:

- Custo adicional: $0-2000
- Tipos de trabalho: Histórias de Usuário
- Tratamento especial: Tem prioridade em todos os estágios, se o prazo estiver próximo; caso contrário é tratada como uma classe de serviço padrão. A implantação emergencial pode ser necessária.

Classe Urgente:

- Custo adicional: $3000-5000
- Tipos de trabalho: Defeito impeditivo
- Tratamento especial: Rompe os limites do WIP, para todo o WIP existente; implantação emergencial

Em nosso sistema de entrega de software, uma classe de serviço se diferencia de um tipo de trabalho comum, de acordo com o tratamento especial que recebe. Observe que sempre haverá custos associados a algum tratamento especial, e medir o fluxo irá permitir fazer uma estimativa mais embasada. Qual o efeito do tratamento especial? Qual seu impacto nos demais itens do fluxo (considerando que se possa estimar o custo de atraso)? Quanto tempo mais, ao todo, será gasto devido a mudanças de contexto e de implantações prioritárias?

Uma boa ideia é colher dados referentes a várias semanas. Assim será possível chegar a uma melhor estimativa dos custos de tratar certos itens de forma especial. Ficará mais claro o efeito dos itens urgentes sobre o diagrama de tempo de ciclo e o quanto implantações emergenciais atrapalham o fluxo e consomem recursos.

Em geral, é uma boa ideia determinar um limite para o número de itens em classes não-padrão em seu sistema. Leve em conta a regra: "se tudo é urgente, nada é urgente".

Visualizando classes de serviço

Há diversas formas de se visualizar classes de serviço. Duas das mais populares são um código de cores (**Figura 18**) e raias (**Figura 19**). Pode-se também usar uma combinação dessas duas formas.

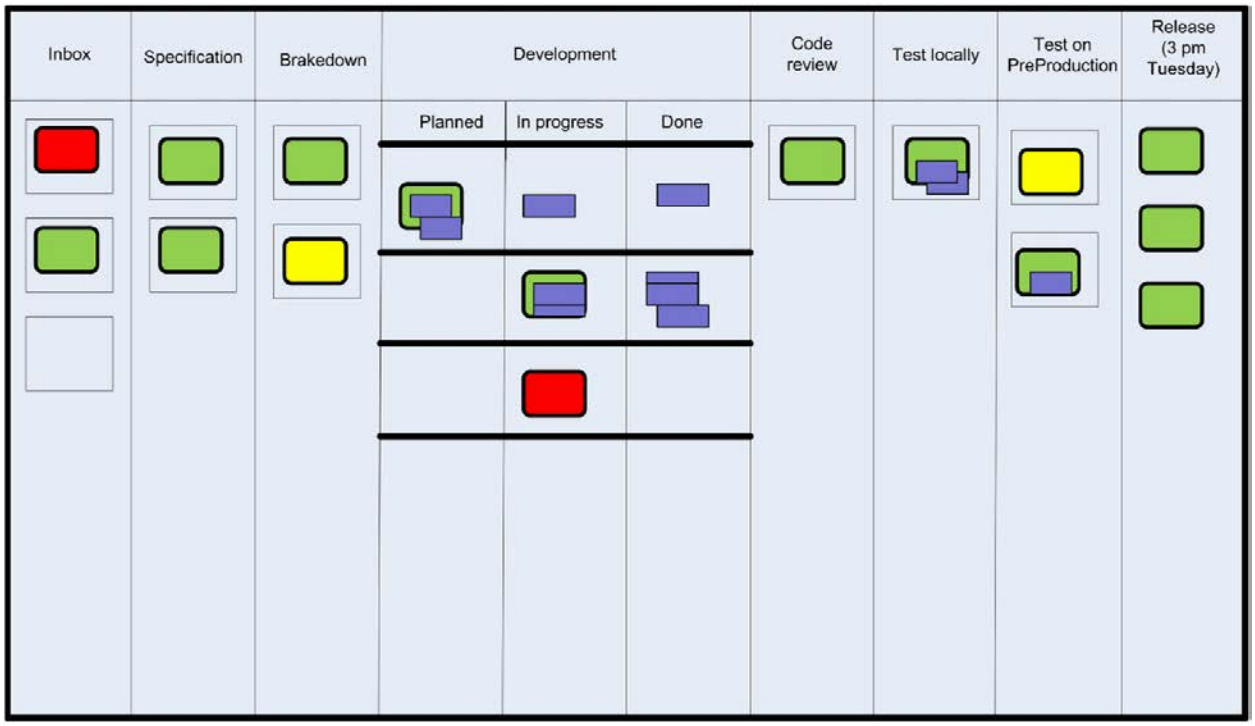

Figura 18. Classes de serviços visualizadas com um código de cores

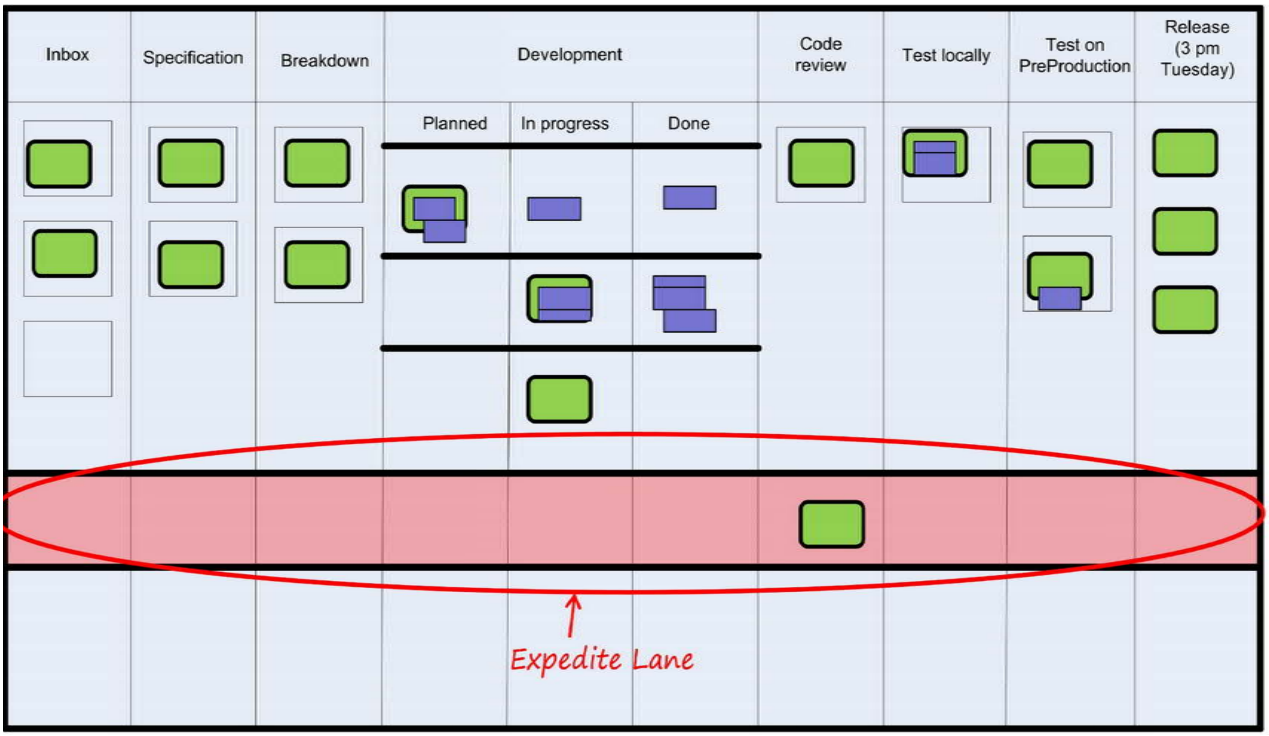

Figura 19. Classes de serviços visualizadas com raias

A utilização das classes de serviço permite lidar com cada tipo de item de forma racional, de acordo com seu impacto econômico, em vez de atacar o incidente reativamente. Permite também fazer promessas fundamentadas aos clientes, levando em conta a classe de serviço. Esse é um tópico que cobriremos em mais detalhes no Passo 9.

Passo 8: Gerenciamento do fluxo

Ao chegar neste ponto, já estaremos operando em um ambiente ágil e maduro. Visualizamos o fluxo de trabalho, o trabalho em progresso está limitado, políticas de qualidade estão estabelecidas, o fluxo está sendo medido. O próximo passo é aprender a interpretar o sistema e tomar ações apropriadas quando houver oportunidades de melhoria.

Filtros de decisões

Os filtros de decisão Agile e Lean, criados por David J. Anderson, serão úteis para guiar nossas ações. Veja a seguir as questões/aspectos considerados por cada um.

Filtro de decisões Agile

- Estamos obtendo progresso mesmo com informação imperfeita?
- Estamos encorajando uma cultura baseada em alta confiança?
- Estamos tratando o WIP como um risco e não como um ativo?

Os dois primeiros itens do filtro são mais abrangentes; são utilizados para tomar decisões mais bem embasadas, ao se enfrentar desafios e situações difíceis.

Filtro de decisões Lean

- Valor é mais importante que fluxo
- O fluxo é mais importante que eliminação de desperdícios
- Elimine desperdícios para melhorar a eficiência

O filtro de decisões Lean diz, essencialmente, que o valor é mais importante que o fluxo. Portanto, deve-se ter cuidado ao abrir mão do valor gerado para reduzir o tempo de ciclo. Isso é comum em projetos ágeis, nos quais o valor de negócio – e às vezes a qualidade – são sacrificados apenas para conseguir que mais trabalho seja feito.

Em um dos piores casos que já presenciei, três equipes trabalhando no mesmo produto precisaram fazer horas extras por três meses. Quando perguntei porque isso aconteceu, a única resposta que consegui foi que havia uma longa lista de funcionalidades a serem desenvolvidas, e de defeitos a corrigir. Quando perguntei sobre o objetivo de negócio, a maioria me olhou com olhar de interrogação. Consegui convencer o grupo de que havia necessidade de chegar a um acordo com o cliente sobre a visão para a entrega, em vez de trabalhar cegamente com base em uma longa lista – e finalmente senti que estávamos indo na direção certa.

Depois de uma semana de muito trabalho e negociações, os POs e o gerente de programa apresentaram uma nova visão. A maioria ficou satisfeita, mas uma pessoa perguntou: "Notaram que o quadro não tem nenhum item que nos aproxime dessa visão?" Apesar de terem tentado respondê-lo com naturalidade, a vergonha estava estampada na cara de todos. A equipe tinha no mínimo dez itens em vários estágios no quadro, e nenhum deles estava alinhado com a nova visão.

Terem trabalhado tantas horas extras, por tanto tempo, parecia ter sido um enorme desperdício de tempo e recursos.

Foi uma lição valiosa. Pode-se ter o melhor sistema Kanban do mundo e um ótimo fluxo pela cadeia de valor inteira, mas se não houver ciclos de feedback com base na sua hipótese de valor, você pode simplesmente estar gerando desperdício com eficiência. Porém, o fluxo é mais importante que eliminação de desperdício; então tenha cuidado ao trocar fluxo por uma maior utilização de capacidade, por exemplo.

O momento de procurar eliminar desperdícios chega quando o valor e o fluxo já estiverem otimizados; mas lembre-se de prestar mais atenção ao produto do que às pessoas.

Com os filtros de decisão Agile e Lean em mente, vamos examinar alguns conceitos fundamentais na gerência do fluxo em nosso sistema de entrega de software.

Otimizando o fluxo, não a utilização

Ao procurar oportunidades de melhorias, evite pensar em termos de utilização. Busque oportunidades para aumentar o fluxo de itens de trabalho passando pelo seu sistema, e faça perguntas como:

- Os limites de trabalho em progresso (WIP) são adequados?
- É possível diminuir o tamanho das histórias?
- Existe alguma forma de identificar funcionalidades grandes, antes que entrem em seu sistema e acabem bloqueando a capacidade por longos períodos?
- É possível equilibrar o tamanho das histórias de usuário para criar um fluxo mais contínuo?
- Pode-se treinar a equipe visando à flexibilidade, para evitar silos e aliviar gargalos?
- Existem buffers adequados em seu sistema para lidar com variações?
- O foco está na otimização do todo e não na otimização de estágios individuais?

A otimização do fluxo em vez da otimização da utilização está no coração do pensamento Lean.

Os fabricantes de veículos norte-americanos costumavam medir e premiar pessoas de acordo com a produção de portas, por exemplo – mesmo que fossem ficar empilhadas em algum armazém. Como efeito, máquinas individualmente trabalhavam de forma rapidíssima, mas a produção como um todo era mais lenta. Grandes armazéns tornavam difícil a procura pelas peças ou de movê-las de uma parte a outra. E se tinha uma quantidade enorme de recursos financeiros presa em estoques.

O sistema Toyota de produção mudou completamente esse panorama. Mostrou que colocar o foco no produto final, e adequar a velocidade das máquinas individuais à velocidade com que veículos saíam da linha de produção (tempo tático), trazia vantagens econômicas incontestáveis.

O principal é sempre pensar em termos de fluxo do produto final, e não em tornar mais rápidos os passos individuais. Infelizmente, muitos gerentes ainda dão ênfase muito maior em fazer as

pessoas trabalharem mais rapidamente, do que na qualidade e no fluxo do produto. Lembre-se sempre de observar o produto, e não as pessoas!

Em conversa recente com um cliente, em que todos estavam preocupados com a utilização, surgiu a pergunta: "O que fazer para garantir que todos estejam sempre ocupados?", à qual alguém respondeu: "Tenho uma tarefa em que posso trabalhar sempre que não houver nada mais importante a fazer". Perguntei há quanto tempo vinha trabalhando naquela tarefa e qual o valor que gerava. "Trabalho nela há um ano, mas ainda não foi lançada, e nenhum valor foi gerado ainda". Um de seus colegas perguntou quando acreditava que poderia ser lançada. A resposta: "Devido a mudanças recentes em nosso portfólio de produtos, é provável que a tarefa não faça mais sentido e seja descontinuada em uma semana ou duas". O resto do grupo riu e admitiu abertamente que esse não era um cenário incomum na empresa.

Aliviando gargalos

A Teoria das Restrições (TOC – Theory of Constraints) prega que sempre haverá um gargalo em um sistema, limitando o fluxo de produção. Apesar da TOC oferecer uma visão simplificada do fluxo e da forma de se lidar com os gargalos, ajuda a entender a importância de ver o sistema como um todo, e de aplicar esforços onde geram mais valor.

A utilização de um sistema puxado kanban torna aparentes os gargalos. Será percebido visualmente o trabalho se acumulando nos estágios anteriores ao gargalo e ficando escasso em estágios posteriores. Uma reação comum, nesse caso, é adicionar mais capacidade – mas essa nem sempre é a maneira mais efetiva de se lidar com gargalos. Não se pode escalar pessoas como máquinas, e o possível aumento de capacidade por meio de aumento de pessoal muitas vezes é consumido por custos adicionais de coordenação e de treinamento. Vale lembrar a lei de Brook: "Adicionar mais gente a um projeto de software atrasado gera mais atraso."

Em vez disso, procure ações para proteger o gargalo de trabalho desnecessário. Em um dos projetos de participei, identificamos que a equipe de POs representava o gargalo. Ficou claro que boa parte do tempo era gasta na investigação de defeitos e no apoio a usuários que não haviam recebido treinamento adequado para trabalhar com o sistema. Essas tarefas poderiam ser facilmente transferidas para outros integrantes da equipe de desenvolvimento. Decidimos então fazer um rodízio dos desenvolvedores para realizar as tarefas, com isso retirando a sobrecarga no grupo que estava gerando o gargalo.

A alternativa de mais longo prazo seria procurar entender o que levou à situação atual e melhorar os fluxos no sistema para torná-los mais intuitivos – ou treinar melhor os usuários. A remoção de trabalho que não gera valor é de longe a forma mais efetiva de aliviar gargalos.

Uma terceira alternativa seria investigar se havia impedimentos bloqueando itens de trabalho para a equipe de POs, o que poderia estar consumindo sua capacidade. Nesse caso, a equipe deveria fazer o que se chama de *swarm* (esforço concentrado) para remover esses impedimentos o quanto antes.

No livro "Kanban", de David J. Anderson, são apresentadas outras maneiras de se lidar com gargalos.

Introduzindo buffers

Se você sabe que existe um gargalo no sistema, é uma boa ideia protegê-lo com um *buffer* logo antes. Se, por exemplo, seu gargalo for o desenvolvimento, um estágio de buffer com itens "Prontos para desenvolvimento" poderia ser adicionado. Escolher o tamanho correto para o buffer exige experiência. Se o buffer esvazia frequentemente, duas vezes por semana, por exemplo, é recomendável aumentar o seu tamanho ou avaliar se o estágio que está sendo protegido ainda é um gargalo (pode ser que um estágio anterior seja o gargalo, porque o buffer está ficando vazio constantemente).

Planejando entregas

Apesar de ser chamado "Desenvolvimento de Produtos", a maioria das equipes que trabalham atualmente com desenvolvimento de software desenvolvem suas atividades sob as restrições da gerência de *projetos* – custo, tempo e escopo. Pensar apenas em termos de fluxo é muitas vezes uma abordagem ingênua, já que haverá gerentes ou diretores que esperam respostas a questões como: Estamos no prazo? Os custos estão dentro do esperado? Podemos entregar o escopo combinado?

Para chegar a essa situação, duas coisas precisam ser feitas. Primeiro, deve-se concordar que, como não se pode fixar as três restrições ao mesmo tempo, o escopo irá ser flexível. Atrasar uma data combinada é difícil e comumente significa muito tempo gasto em reorganização, coordenação e comunicação. Por outro lado, aumentar o orçamento com o aumento de pessoal também, conforme já vimos, raramente é uma boa ferramenta para atingir um prazo apertado. Adicionar pessoas é um movimento estratégico de longo prazo. Aumentar o orçamento não significa aumentar o número de pessoas, e sim aumentar a quantidade de horas que as pessoas trabalham. Isso pode ser uma ferramenta útil se houver apenas uma semana de prazo. Mas em todas as outras situações, vale lembrar a declaração de Kent Beck, de que caso se tenha um problema que não pode ser resolvido com uma semana de horas extras, o problema não deve ser resolvido com horas extras.

É comum que gerentes de projeto utilizem horas extras como ferramenta desesperada para mostrar que estão tomando alguma atitude. Os gerentes sabem que não resolverão os problemas dessa forma, mas fazem assim mesmo para mostrar algum tipo de ação. São raros os casos em que alterar o prazo é a solução correta. Em caso de dúvida, o melhor é manter o escopo flexível.

Em segundo lugar, é preciso entender o que é "escopo flexível" – um conceito que costuma ser interpretado como abordagem "deixe fazer" no desenvolvimento de software, com pouca ou nenhuma disciplina. Contudo, trabalhar com escopo flexível exige justamente o contrário. Deve haver tanto disciplina quanto habilidade de medir o progresso com precisão, para garantir que são bem fundamentadas as decisões tomadas em um ambiente de mudanças contínuas. Esse é um fator fundamental para trazer o melhor retorno sobre investimento (ROI) possível.

Sabemos que nossas estimativas originais, em termos de complexidade, custo e valor para o negócio, foram feitas no momento em que tínhamos poucas informações disponíveis. Ainda assim, nossos prazos e custos estão apoiados sobre essas suposições. Portanto, é fundamental medir nosso progresso para ter certeza que o projeto ainda é viável.

Já que estamos utilizando diagramas CFD, por que não usá-los para medir também o progresso? Os orçamentos, na maioria, são feitos com base em entregas; por isso vamos ver um exemplo desse tipo. Ao ter um orçamento, um prazo e um escopo inicial, simplesmente desenhamos nossa velocidade esperada no CFD e medimos nosso progresso com base nela.

Lembre-se que a maioria dos projetos gera uma curva em S, e que os desvios, como regra geral, indicam que há mais informações disponíveis do que havia antes. Somos, portanto, capazes de tomar decisões mais bem embasadas (inclusive a de encerrar o projeto).

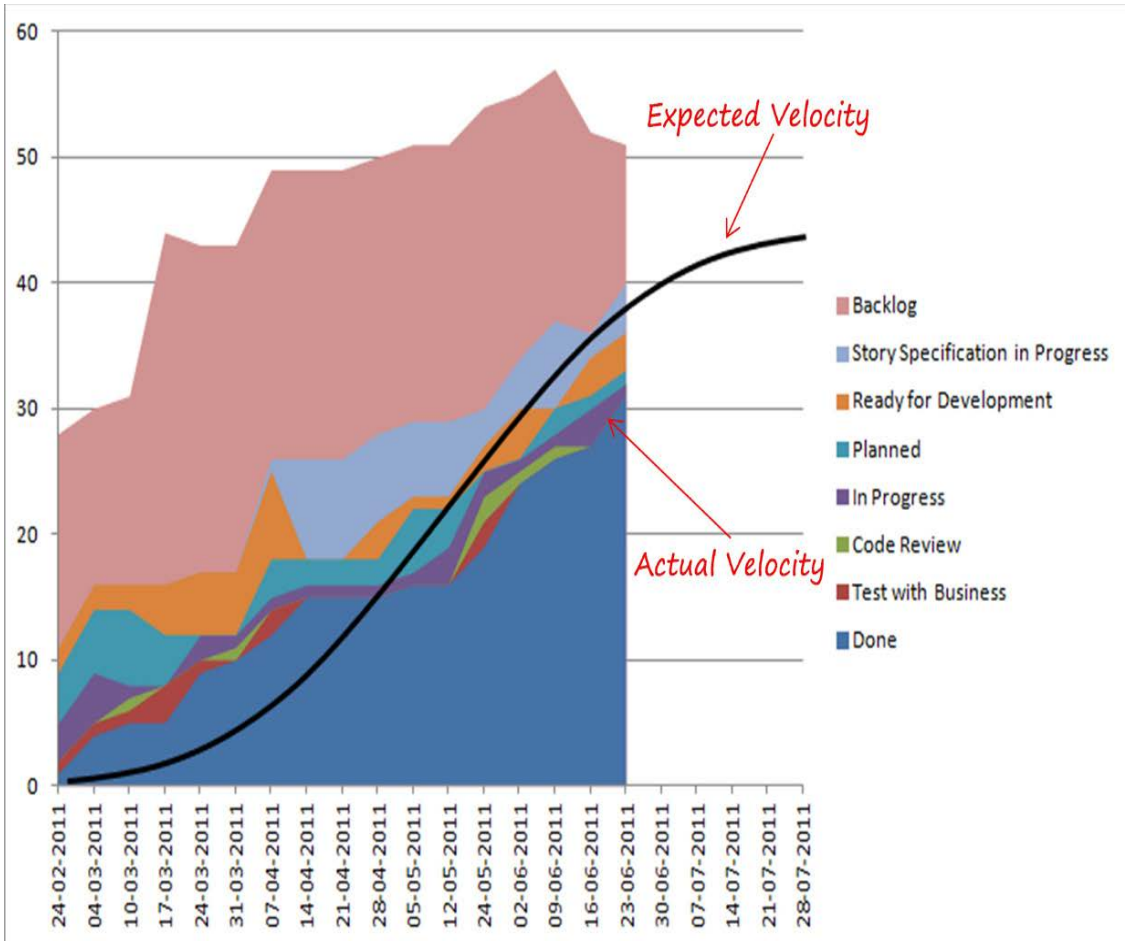

Figura 20. Planos de entregas visualizados no diagrama de fluxo cumulativo (CFD)

A **Figura 20** mostra um exemplo real de curva S sobre um gráfico CFD. Como se vê, era esperado que o backlog crescesse por volta de 40% (com base em experiência) – de 28 para 40 pontos. Mas em determinado momento, ele havia mais que duplicado de tamanho (57 pontos). Apesar de a imagem mostrar que se começou com uma velocidade acima do esperado, a velocidade caiu depois da metade da release, devido a problemas de qualidade.

Experimente!

Gerenciar o fluxo também significa tentar melhorar continuamente (visto em mais detalhes no Passo 10). Muitos projetos fazem isso às cegas, sem ter como saber se as alterações foram de fato bem sucedidas.

Infelizmente, muitos projetos ágeis estão nessa categoria. As retrospectivas são utilizadas para estabelecer experimentos, mas o acompanhamento (quando ocorre) apenas inclui verificar que o experimento foi conduzido. É claro que sempre haverá boa dose de incerteza, mas de forma mais frequente do que se pensa, as métricas citadas neste livro oferecem indicação visual clara de que algo funcionou ou não.

Por exemplo, considere o caso em que você decide incluir mais analistas de teste à equipe de desenvolvimento, para fazer mais testes com antecedência. É esperado que haja uma queda no índice de defeitos em um prazo razoável, e então se possa trabalhar com um limite de WIP menor para o estágio de testes.

Gerenciar o fluxo significa ser capaz de interpretar seu sistema de software, para tomar as melhores decisões possíveis com a informação disponível, em busca do maior retorno sobre o investimento.

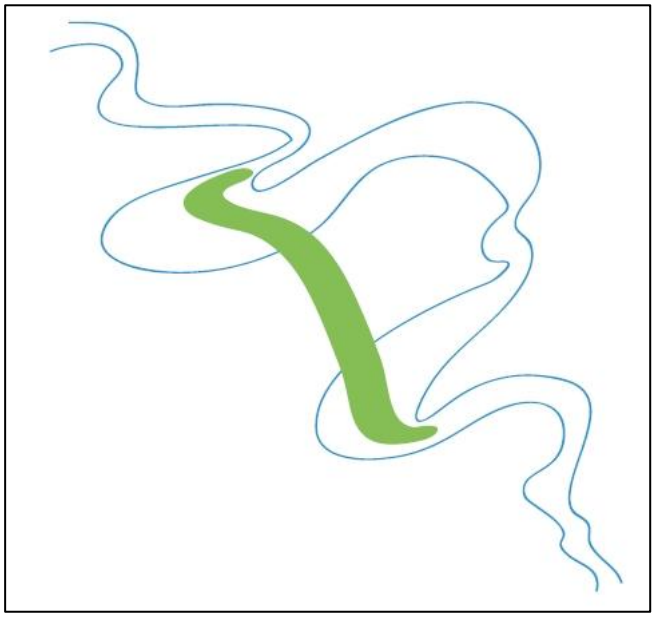

Figura 21. Alivie gargalos para melhorar o fluxo

Passo 9: Estabelecer SLAs

Nesse ponto, já estamos adiantados na direção de estabelecer um sistema de entrega de software efetivo e confiável, e é o momento de mostrar os resultados para o público externo. Possuir um sistema puxado estável, e utilizar métricas simples para medir o desempenho do sistema, torna possível estabelecer acordos de níveis de serviços (SLAs) que realmente são cumpridos. Isso ajuda a manter o sistema saudável e evita a tradicional recaída de se voltar ao caos e ao apagamento de incêndios, a partir do momento que a iniciativa Kanban deixar de ser novidade. Então como funciona?

Estabelecendo o SLA correto

Diferentemente das abordagens ágeis tradicionais como o Scrum, que valorizam muito a previsibilidade por meio do compromisso do Sprint, um sistema kanban pressupõe que a

previsibilidade será adquirida por meio de um sistema de entrega de software que funciona de forma previsível. Existe uma diferença sutil nessas duas abordagens em relação à previsibilidade, a qual não deve ser subestimada. Os métodos ágeis são orientados por um plano; os sistemas kanban, são orientados pelo fluxo.

Caso cada classe de serviço tenha tratamento específico e consistente ao longo do tempo, e as consequências dos esforços de melhoria sejam medidas, é grande a possibilidade de que o tempo de ciclo e a qualidade e os custos melhorem com o tempo. Tais informações podem ser compartilhadas com seus clientes. As classes de serviço mencionadas anteriormente poderiam ter SLAs semelhantes aos exemplos abaixo:

Classe Padrão
SLA:

- Média: 15 dias
- 90% dentro do prazo de 21 dias
- Todas no prazo máximo de 30 dias

Classe urgente
SLA:

- Média: 2 dias
- 90% dentro do prazo de 3 dias
- Todas no prazo máximo de 4 dias

Classe de prazo fixo
SLA:

- 98% dentro do prazo

Classe prioritária
SLA:

- Média: 8 dias
- 90% dentro do prazo de 13 dias
- Todas no prazo máximo de 18 dias

O fundamental aqui é que os números são baseados em informações colhidas por meio do acompanhamento do desempenho de nosso sistema de entregas. Caso surja a necessidade de informações mais detalhadas, podemos facilmente ajustar as métricas. Se, por exemplo, as classes padrão variarem muito em tamanho, pode-se adicionar detalhes para mostrar aos clientes o efeito direto dessa variação – como nos exemplos a seguir.

Classe padrão

SLA 200-300 pontos (grande):

- Média: 15 dias
- 90% dentro do prazo de 21 dias
- Todas no prazo máximo de 30 dias

SLA 100-200 pontos (média):

- Média: 13 dias
- 90% dentro do prazo de 18 dias
- Todas no prazo máximo de 25 dias

SLA 10-100 pontos (pequena):

- Média: 14 dias
- 90% dentro do prazo de 14 dias
- Todas no prazo máximo de 18 dias

Para muitos clientes, essas informações são valiosas para a priorização. E saber que esses números são reais gera muito mais confiança e colaboração do que em projetos anteriores. Isso também contribui para tornar visíveis os benefícios diretos de se dividir o trabalho em partes menores – tanto em termos de riscos, quanto de custo e tempo de ciclo. Para garantir que todos estejam cientes dos SLAs atuais e das Classes de Serviço, a maioria das equipes considera útil afixar essas informações próximas ao quadro, como mostra a **Figura 22**.

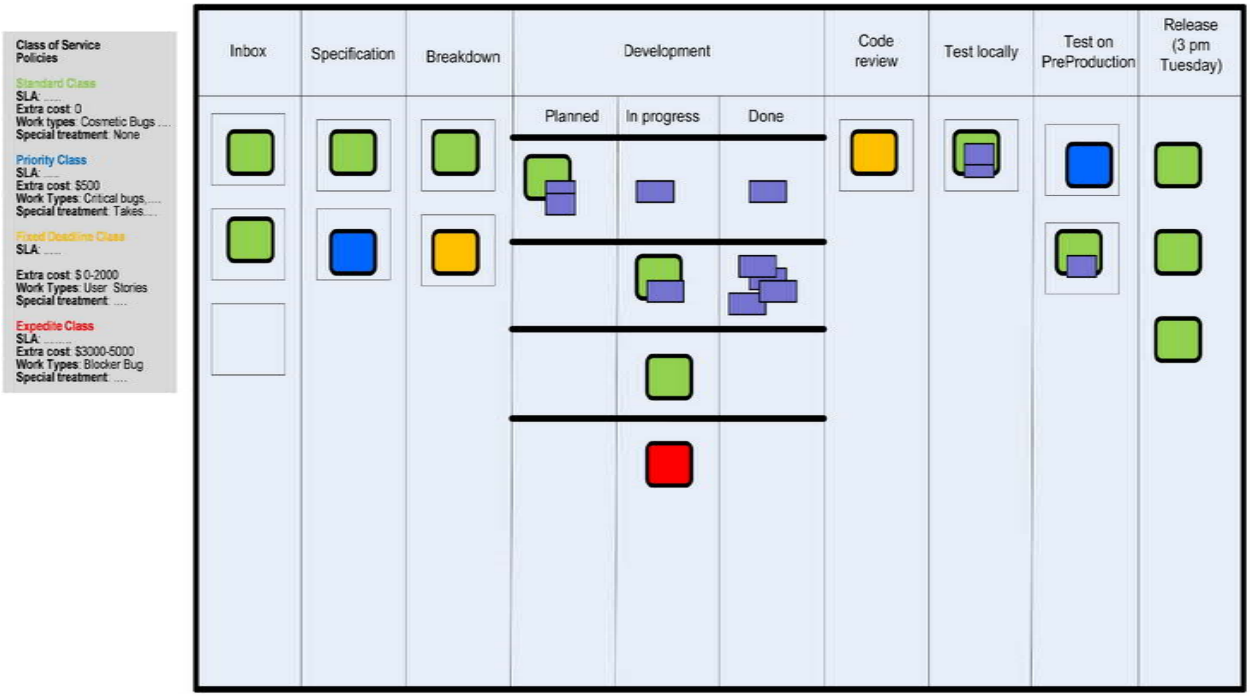

Figura 22. Políticas para classes de serviços afixadas ao lado do quadro

Passo 10: Melhoria contínua

Criar e manter um ciclo constante de melhorias é um dos elementos mais importantes ao implementar o Kanban. Como dito anteriormente, o Kanban é um método para criar mudanças evolucionárias – e a boa notícia é que ter seguido os passos anteriores torna muito mais fácil o processo de melhoria contínua.

A grande distribuição de informações com a visualização do fluxo de trabalho e de políticas explícitas, além dos SLAs para cada classe de serviço, é grande incentivo ao diálogo constante sobre oportunidades de melhoria – muito além do que se vê em projetos de software tradicionais. Todos os dias, somos obrigados a realizar decisões explícitas sobre como lidar melhor com o trabalho. O aumento na irradiação de informações também constitui boa base para a melhoria contínua. E parece haver um entendimento mais profundo dos conceitos do Agile e do Lean, de quem trabalha em um sistema Kanban; portanto é mais difícil acontecer retrocessos para um processo anterior, ou a adoção de algum antipadrão.

Como são coletados dados reais, o Kanban também oferece a oportunidade de executar e validar experimentos de forma mais científica, em comparação ao que acontece em projetos ágeis tradicionais. As iniciativas para melhorar o tempo de ciclo geralmente resultam em efeitos mensuráveis. Isso torna a melhoria contínua em um sistema kanban muito mais confiável e consistente. E como podemos ver e medir os efeitos das iniciativas de mudança, ficamos muito mais suscetíveis a aumentar a exigência quanto à qualidade do sistema de entregas.

Para algumas pessoas, trabalhar com Kanban faz com que vejam pela primeira vez o sistema de entregas como um todo, o que dá uma percepção profunda do trabalho das outras pessoas, e de como dependem de você e vice-versa. Também surgem oportunidades de otimizar não apenas silos individuais, mas todo o sistema de entregas.

Apesar de os círculos espontâneos de qualidade (o termo Lean para discussões entre membros da equipe) serem excelentes veículos para a melhoria contínua, muitas equipes usando Kanban podem também se beneficiar de uma cadência regular de retrospectivas (eventos Kaizen, na terminologia Lean). As retrospectivas dão oportunidade à equipe de ver o trabalho de outra forma; permite que surjam sugestões para mudanças estruturais maiores, conhecidas no Lean como Kaikaku (mudança dramática). A combinação de círculos de qualidade, reuniões diárias e uma cadência de retrospectivas, forma um coquetel poderoso em prol de melhorias.

Como mencionado anteriormente, um fator fundamental na obtenção desses efeitos é se ater à regra "Mude suas políticas; não as quebre". Se as pessoas não agem de acordo com as políticas da equipe, o sistema de entregas provavelmente irá se degradar com o tempo – e não será possível ver o verdadeiro valor de se visualizar o trabalho.

Frequentemente me perguntam se o Kanban não serviria como desculpa para voltar a práticas “disfuncionais”, dada à ausência de regras para garantir o uso de práticas Lean ou Agile. Embora eu compreenda a lógica por trás das perguntas, não tenho esse receio.

O Kanban é uma forma única de catalisar os princípios do Agile e do Lean, mesmo em situações em que é aparentemente impossível fazê-lo. As poucas vezes que presenciei o Kanban falhar foram devido à falta de apoio da gerência, que em alguns casos até mesmo trabalhou contra o processo. Ou onde não houve esforços para explicar a importância de visualizar o trabalho, gerenciar o fluxo e obter feedback.

Para ser bem sucedido com Kanban, é necessário o comprometimento da gestão. Além disso, as pessoas que estão adotando os princípios em seus trabalhos diários devem entender sempre porque isso faz sentido. Quando tais aspectos estão presentes, não há razão para esperar que a

iniciativa irá falhar, ou que o sistema será usado para desfazer iniciativas de mudanças anteriores, ou regredir para práticas antigas e disfuncionais.

Boa sorte na sua jornada!

Espero que estes capítulos possam ter inspirado você a avançar na sua jornada Agile e Lean – e a usar o Kanban nos projetos de sua empresa. Adoraria receber seu feedback sobre esse trabalho, assim como conhecer experiências suas sobre como o livro pode tê-lo ajudado a obter um retorno de investimento melhor para você e os seus clientes.

Para se aprofundar em Kanban, sugiro entrar nos grupos de Kanban "kanbandev" e "kanbanops" do Yahoo Groups, para participar de discussões sobre a aplicação do Kanban na prática. Uma forma de aprofundar ainda mais o conhecimento é através da leitura dos seguintes livros:

- Kanban, David J. Anderson, 2010
- The Principles of Product Development Flow: Second Generation Lean Product Development, Donald G. Reinertsen, 2009
- The Elegant Solution: Toyota's Formula for Mastering Innovation, Matthew may, 2008
- Lean Thinking: Banish Waste and Creaste Wealth in Your Corporation, James P. Womack and Daniel T. Jones, 2003
- The Toyota way: 14 Management Principles from the World's Greatest Manufacturer, Jeffrey Liker, 2004
- The Lean Startup: How Constant Innovation Creates Radically Successful Businesses

Boa sorte!

Jesper Boeg
jbo@trifork.com