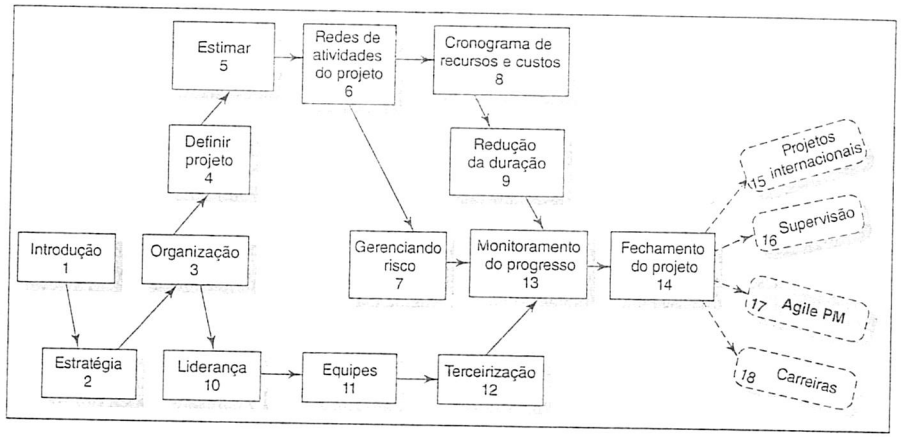
CAPÍTULO DEZESSETE

Introdução ao gerenciamento ágil de projetos



Sabemos menos sobre o projeto hoje do que em qualquer momento do futuro.

— Chet Hendrickson

Quando o gerenciamento de projetos ingressou no novo milênio, muitos profissionais perceberam que os métodos de gerenciamento de projetos "tamanho único" não satisfaziam as suas necessidades. Isso se aplicava especialmente àqueles que trabalhavam em projetos de desenvolvimento de software e produtos, em que o produto final não é bem-definido e evolui com o tempo. Esse ambiente de projeto exige flexibilidade e a capacidade de administrar mudanças à medida que mais informações e aprendizados surgem. Aí entra o **gerenciamento ágil de projetos** (Agile PM, do inglês *Agile Project Management*). Em vez de tentar planejar todo o projeto desde o início, o gerenciamento ágil de projetos se utiliza de ciclos de desenvolvimento incrementais e iterativos para concluir projetos.

Ken Schwaber usa a analogia da construção de uma casa para explicar a diferença entre desenvolvimento incremental e iterativo e o gerenciamento de projetos tradicional.[1] A abordagem tradicional seria quando os compradores não podem se mudar para a casa até ela estar totalmente pronta. A abordagem iterativa seria construir a casa aposento por aposento. A hidráulica, a elétrica e a infraestrutura seriam feitas primeiro para o aposento mais importante (por exemplo, a cozinha), sendo, então estendidas para cada cômodo à medida fossem construídos. Sempre que finalizado um cômodo, os construtores e os clientes avaliariam o progresso e fariam ajustes. Em alguns casos, os clientes podem decidir que não precisam daquele aposento extra que planejaram. Em outros casos, eles podem fazer acréscimos que não sabiam que precisariam ter. Em última instância, a casa é construída para se adequar aos desejos dos clientes.

O gerenciamento ágil é ideal para projetos exploratórios, em que os requisitos precisam ser descobertos e novas tecnologias têm de ser testadas. Ele enfoca a colaboração ativa entre a equipe do projeto e os representantes do cliente, fragmentando os projetos em pequenos pedaços funcionais e adaptando-os aos requisitos em mudança. Embora princípios de desenvolvimento iterativo já circulem há algum tempo, foi apenas recentemente que as metodologias de gerenciamento ágil fincaram raízes no gerenciamento de projetos.

Neste capítulo, os princípios centrais do gerenciamento ágil são discutidos e comparados aos métodos tradicionais de gerenciamento de projetos. Uma metodologia de gerenciamento ágil específica, chamada Scrum, é usada para descrever o funcionamento desses princípios. O capítulo termina com um debate sobre limitações e problemas. O objetivo não é fazer um relatório completo sobre todos os métodos associados ao gerenciamento ágil, mas mostrar como ele funciona.

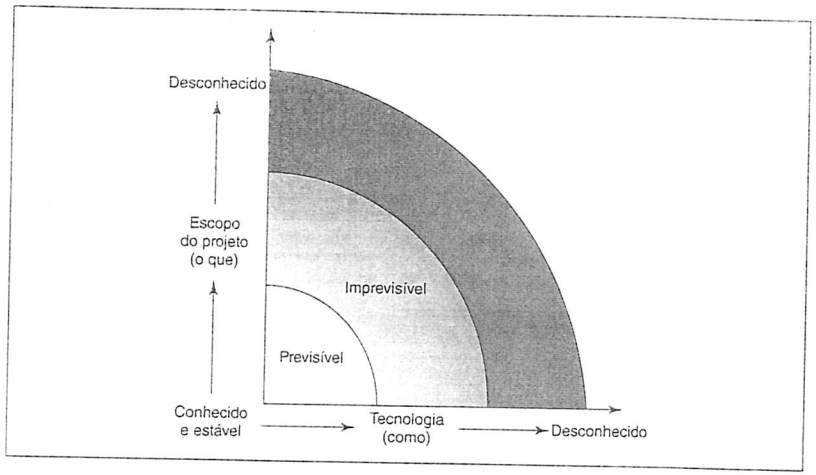
Métodos tradicionais *versus* gerenciamento ágil

As abordagens tradicionais do gerenciamento de projetos se concentram firmemente no planejamento completo desde o início. A lógica é que se você planejar, executar seu plano e corrigir eventuais desvios, você tem uma alta probabilidade de sucesso. Após o escopo do projeto estar firmemente estabelecido, todos os detalhes do projeto são definidos por meio da EAP. A maioria dos problemas e riscos é identificada e avaliada antes de o projeto começar. Estimativas são feitas, recursos são atribuídos, ajustes são efetuados e, por fim, criam-se um cronograma e um orçamento de linha de base. O controle do projetos é uma comparação entre planejado e efetivo e medidas corretivas para voltar ao planejado.

O gerenciamento de projetos tradicional exige um grau bem alto de previsibilidade para ser eficaz. Para que os planos sejam úteis, os gerentes precisam ter uma ideia firme sobre o que deve

[1] Schwaber, K., *Project Management with Scrum*, (Seattle: Microsoft, 2004) p. xviii.

FIGURA 17.1
Incerteza do projeto



ser realizado e como fazê-lo. Por exemplo, quando se trata de construir uma ponte sobre um rio, os engenheiros podem se basear em tecnologia comprovada e princípios de desenho para planejar e executar o projeto. Nem todos os projetos desfrutam dessa certeza. A Figura 17.1 diz respeito a essa questão e é, muitas vezes, usada para auxiliar o uso do Gerenciamento ágil.

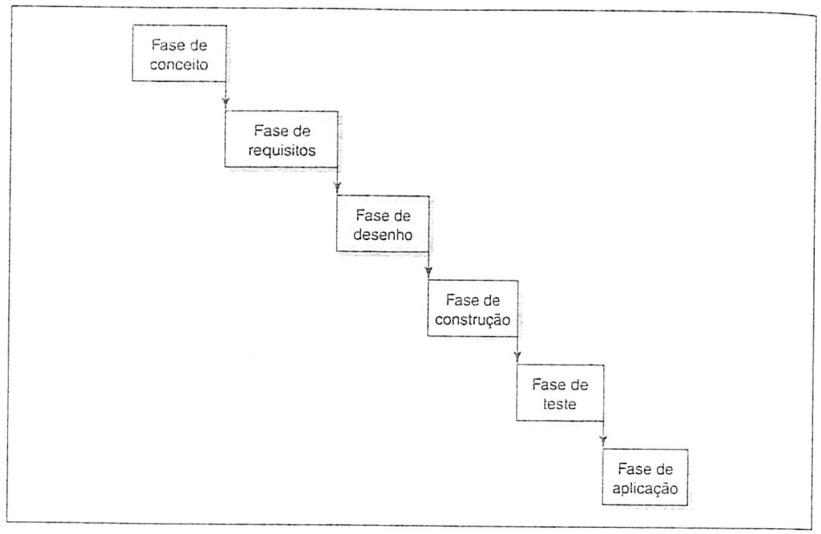
A incerteza do projeto varia em relação à medida que o escopo do projeto é conhecido e estável e em que a tecnologia a ser usada é conhecida e comprovada. Muitos projetos, como o exemplo da ponte, assim como outros de construção, eventos, extensões de produto, campanhas de marketing etc., têm escopos bem-estabelecidos e utilizam tecnologia comprovada, o que dá um grau de previsibilidade para um planejamento eficaz. No entanto, quando o escopo e/ou a tecnologia do projeto não são completamente conhecidos, as coisas tornam-se muitos menos previsíveis e os métodos guiados por planos fracassam.

Por exemplo, antes do gerenciamento ágil, os projetos de desenvolvimento de software utilizavam uma abordagem em "cascata", que contém uma série de fases lógicas, em que o progresso flui de uma fase para a próxima (Figura 17.2). O principal pressuposto é que os requisitos essenciais podem ser definidos desde o início para que o software possa ser desenhado, criado e testado. Projetos de software normalmente envolvem muitos clientes diferentes, com diferentes necessidades. Essas necessidades mudam frequentemente, dificultando, muitas vezes, a articulação entre elas. Em muitos casos, os clientes só começam a entender o que realmente desejam quando alguém lhes diz o que querem. Nessas condições, seria difícil (e talvez inútil) desenvolver uma lista detalhada de requisitos de escopo no lançamento do projeto. Essa é uma das principais razões por que os projetos de software que utilizam a abordagem em cascata têm um histórico de atrasos e/ou cancelamentos.

A tecnologia pode ser outra fonte de imprevisibilidade. Por exemplo, uma equipe de desenvolvimento encarregada de desenhar a próxima geração de carros elétricos sabe que deve criar um veículo que acomode quatro adultos com conforto e faça 360 km por carga, mas pode não saber se existe tecnologia de bateria para abastecê-lo. Novamente, seria muito difícil desenvolver um cronograma confiável quando dúvidas assim existem.

O ponto principal é que as técnicas tradicionais de gerenciamento de projeto foram desenvolvidas para funcionar em uma zona previsível, em que o escopo do projeto é razoavelmente bem-definido e a tecnologia a ser usada é estabelecida. O gerenciamento ágil vive na zona da imprevisão. Ele representa um afastamento fundamental em relação à abordagem tradicional do gerenciamento de projetos impulsionado por planos, adotando uma abordagem mais experimental e adaptativa ao gerenciamento de projetos. Os projetos não são executados: eles evoluem. A Tabela 17.1 exibe algumas das diferentes entre o gerenciamento ágil e o gerenciamento de projetos tradicional.

FIGURA 17.2
A abordagem em cascata do desenvolvimento de software



Gerenciamento ágil

Fundamentalmente, o **gerenciamento ágil** está relacionado com a metodologia de planejamento e programação de projetos em ondas sucessivas (Capítulo 5). Isso significa que o desenho final do projeto não é conhecido em grandes detalhes, sendo continuamente desenvolvido por meio de uma série de iterações incrementais ao longo do tempo. As iterações são pequenas janelas de tempo ("caixas de tempo"), normalmente de 1 a 4 semanas. O objetivo é que cada iteração desenvolva um produto maleável que satisfaça um ou mais atributos do produto, a fim de demonstrá-los ao cliente e outras partes interessadas. No fim de cada iteração, as partes interessadas e os clientes revisam o progresso e reavaliam as prioridades para assegurar o alinhamento com as necessidades do cliente e as metas da empresa. São feitos ajustes e um novo ciclo iterativo começa. Cada nova iteração incorpora o trabalho das anteriores e acrescenta novas capacidades ao produto em evolução (Figura 17.3) para gerar a próxima versão expandida do produto. Consulte o "Caso Prático: IDEO – Mestres do design" para ver um exemplo de desenvolvimento iterativo em ação.

Processos de desenvolvimento iterativo trazem as seguintes vantagens:

- Integração, verificação e validação contínuas do produto em evolução.
- Demonstração frequente do progresso para aumentar a probabilidade de que o produto final satisfaça as necessidades do cliente.
- Detecção precoce de defeitos e problemas.

TABELA 17.1
Gerenciamento de projetos tradicional *versus* gerenciamento ágil

Tradicional	Ágil
Desenho no início	Desenho contínuo
Escopo fixo	Escopo flexível
Entregas	Atributos/requisitos
Congelar o desenho assim que possível	Congelar o desenho o mais tarde possível
Baixa incerteza	Alta incerteza
Evita a mudança	Incorpora a mudança
Baixa interação com o cliente	Alta interação com o cliente
Equipes de projeto convencionais	Equipes de projeto auto-organizadas

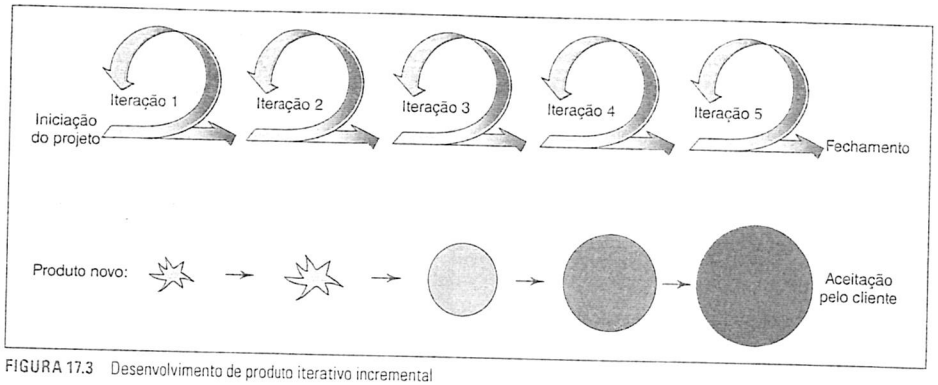


FIGURA 17.3 Desenvolvimento de produto iterativo incremental

Existem evidências crescentes de que o desenvolvimento iterativo e evolutivo é superior ao gerenciamento de projetos guiado por plano quando se trata de criar produtos (leia o "Destaque de Pesquisa: Práticas de Desenvolvimento de Produto que Funcionam").

Deve-se observar que o gerenciamento ágil não é um método fixo, mas uma família de métodos concebidos para responder aos desafios de projetos imprevisíveis. Listamos aqui alguns dos mais populares:

- Scrum
- Extreme Programming (XP)
- Gerenciamento ágil
- Lean Development
- RUP (Rational Unified Process)
- Crystal Clear
- Dynamic Systems Development Method (DSDM)
- Rapid Product Development (RPD)

Apesar de cada um desses métodos ter elementos e aplicações únicos, a maioria se baseia nos seguintes princípios do gerenciamento ágil:

- Foco no valor para o cliente – Priorização de requisitos e atributos guiados por negócios.
- Entrega iterativa e incremental – Criação de um fluxo de valor para os clientes ao "quebrar" a entrega do projeto em pequenos incrementos funcionais.
- Experimentação e adaptação – Teste de pressupostos no início e criação de protótipos funcionais para solicitar *feedback* do cliente e refinar os requisitos do produto.
- Auto-organização – Decisão, entre os membros da equipe, de quem fará o quê.
- Melhoria contínua – As equipes refletem, aprendem e se adaptam às mudanças; o trabalho dá forma ao plano.

A metodologia gerenciamento ágil conhecida como "Scrum" será usada para ilustrar como esses princípios centrais são postos em ação.

Gerenciamento ágil de projetos em ação: Scrum

O Scrum pode ser rastreado até o trabalho de Hirotaka Takeuchi e Ikujiro Nonaka que, em 1986, descreveram uma nova abordagem holística para o desenvolvimento de produtos comerciais. Eles comparam essa abordagem, em que uma equipe transfuncional colabora para desenvolver um produto, com o rúgbi, em que todo o time "tenta percorrer a distância como uma unidade, passando a bola para a frente e para trás". A metáfora do *scrum* (uma jogada de rúgbi) foi expandida e refinada até um modelo bastante prescritivo, que teve sucesso em projetos de alta tecnologia e desenvolvimento de software (ver "Caso Prático: Busca por Almas após 11/09").

CO Scrum, como outros métodos de gerenciamento ágil, começa com uma alta definição de escopo e estimativas de referência de tempo e custo para o projeto. As estimativas de escopo e

CASO PRÁTICO IDEO – Mestres do design*

A IDEO, com sede em Palo Alto, Califórnia, é uma das melhores empresas de design do mundo, responsável por um amplo espectro de inovações em produtos, incluindo o primeiro mouse da Apple, a raquete de tênis Airflow da Head, o lavador de salada da Zyliss e os *smartphones* N-Gage da Nokia. Os muitos clientes da IDEO incluem Pepsi-Cola, 3M, Logitech, Nike e HBO. A IDEO ganhou mais prêmios de *Excelência em Design Industrial BusinessWeek/IDSA* do que qualquer outra empresa.

A abordagem da IDEO ao design de produto depende muito de um processo de desenvolvimento iterativo, em que se usam protótipos iterativos para explorar e refinar as ideias de produtos. O CEO, Tim Brown, diz que o objetivo da prototipagem "é aprender sobre os pontos fortes e fracos da ideia e identificar as novas direções que o protótipo pode tomar".

Por exemplo, a IDEO trabalhou com a Procter and Gamble para desenvolver um novo tubo para o creme dental Crest. O desafio era aperfeiçoar a tradicional tampa de rosquear, na qual o creme dental sempre fica incrustado. A primeira solução da IDEO foi uma tampa com dobradiça. No entanto, quando os designers criaram protótipos iniciais e observaram as pessoas a utilizá-los, logo notaram que elas continuavam tentando desenroscar a tampa, apesar de instruídas sobre o novo funcionamento. Os designers concluíram que a ação era um hábito entranhado, provavelmente impossível de eliminar. Assim, conceberam um modelo híbrido: uma tampa de rosquear com rosca curta, fácil de limpar.

A prototipagem focada resolve os problemas críticos um por um. Brown recomenda que os protótipos só devem tomar o tempo e o esforço necessários para gerar *feedback* útil e desenvolver uma ideia.

Por exemplo, a IDEO estava trabalhando em uma cadeira para a Vecta, um fabricante de móveis de escritório de alto padrão. O projeto evoluiu a tal ponto que a alavanca de ajuste de altura que inclinava com a cadeira se tornara decisiva. A equipe não construiu a cadeira inteira, nem mesmo o mecanismo de inclinação; apenas uma pequena alavanca e sua interface com o mecanismo de inclinação. Só levou algumas horas. Quando finalizado, o protótipo demonstrou rapidamente que o princípio funcionaria.

"Não importa quão inteligente você é, a sua primeira ideia sobre uma coisa nunca está certa", diz Brown, "então o grande valor da prototipagem – prototipagem rápida e barata – é que você aprende sobre a ideia e a aprimora".

* J. M. Pethokoukis, "The Deans of Design: From the Computer Mouse to the Newest Swiffer, IDEO is the Firm behind the Scenes," *U.S. News & World Report*, Posted 9-24-2008; T. Brown, "Design Thinking," *Harvard Business Review*, June 2008, pp. 84-95.

custo devem ser completas o suficiente para deixar a gerência confortável. A teoria é que, uma vez que os requisitos evoluem com o tempo, planejamento detalhado desde o início é desperdício. No lugar de uma EAP de produto, o Scrum usa *atributos* do produto como entregas. Um **atributo** *é definido como uma parte de um produto que entrega alguma funcionalidade útil a um cliente.* No caso de um projeto de software, um atributo pode ser um cliente bancário conseguir mudar sua senha. No caso de um produto de alta tecnologia, pode ser acesso wireless 3G. Os atributos são priorizados segundo seu mais alto valor percebido. A equipe do projeto trabalha, primeiro, nos atributos viáveis de prioridade mais alta. As prioridades são reavaliadas após cada iteração. As iterações são chamadas de *sprints*, e não devem durar mais de quatro semanas. O objetivo de cada *sprint* é produzir atributos completamente funcionais. Isso força a equipe a resolver decisões difíceis no início para criar um demo funcional.

Os atributos específicos são criados de acordo com quatro fases distintas: análise, desenho, construção e teste (Figura 17.4). Cada atributo pode ser visto como um miniprojeto. A primeira fase compreende a análise e revisão dos requisitos funcionais necessários para concluir o atributo. A equipe se compromete com esses requisitos. A segunda abrange o desenvolvimento de um desenho que satisfaça os requisitos do atributo. A terceira se ocupa da construção do atributo de modo que ele seja funcional. Finalmente, o atributo é testado e documentado. No fim de cada *sprint*, os atributos são demonstrados. Nesse modelo de *sprint*, o Scrum utiliza papéis específicos, reuniões e documentos/registros para administrar o projeto.

DESTAQUE DE PESQUISA Práticas de desenvolvimento de produto que funcionam*

Alan MacCormack e seus colegas da Harvard Business School realizaram um aprofundado estudo de dois anos com 29 projetos de software que responderam à seguinte pergunta: "O desenvolvimento evolutivo, em oposição ao modelo em cascata, resulta em maior sucesso?". Modelo em cascata é o nome usado na indústria do software para a abordagem tradicional de gerenciamento de projetos, em que uma estrutura analítica de processo (PBS) é utilizada para, primeiro, definir todos os requisitos de início e, então, iniciar uma sequência de desenho, criação, teste e implantação. Ao passo que desenvolvimento evolutivo é o termo para descrever uma abordagem gerenciamento ágil em que os clientes testam versões iniciais do software e os requisitos vão surgindo e sendo refinados a cada demonstração.

Os resultados do estudo favoreceram claramente a abordagem ágil iterativa ao desenvolvimento de software. Diversas práticas essenciais, hoje associadas ao gerenciamento ágil, foram consideradas estatisticamente correlacionadas aos projetos mais exitosos:

1. Um ciclo de vida iterativo, como lançamento precoce do produto em evolução para que as partes interessadas o revisem e deem *feedback*.
2. Incorporação diária de software novo e *feedback* rápido sobre alterações de desenho.
3. Uma arquitetura de produto flexível, modular e escalável.

MacCormack assevera que a incerteza em projetos de software torna necessários "microprojetos" curtos – até o nível dos atributos. Isso não se limita a projetos de software, incluindo qualquer empreitada de produto novo em que a incerteza seja alta e a necessidade de *feedback* do cliente e refinamento, decisiva para o sucesso.

* A. MacCormack, "Product-Development Practices that Work: How Internet Companies Build Software," *MIT Sloan Management Review*, 42(2), 2001, pp. 75-84.

Papéis e responsabilidades

Existem três grandes papéis no processo de Scrum: proprietário do produto, equipe de desenvolvimento e mestre Scrum.

Dono do produto Esta pessoa atua em nome dos clientes/usuários finais, representando seus interesses. Em projetos de desenvolvimento comercial, o proprietário do produto pode ser o gerente de produtos. Em projetos internos, poderia ser o gerente do grupo de negócios que se beneficiará com o produto. Em outros casos, o proprietário poderia ser um representante da empresa cliente. Ele tem a responsabilidade de garantir que a equipe de desenvolvimento concentre esforços em desenvolver um produto que cumpra o objetivo comercial do projeto.

O proprietário, consultando os demais, estabelece uma lista inicial de requisitos do produto e os prioriza no *backlog* de produtos. Frequentemente, os proprietários trabalham com a equipe de desenvolvimento para refinar os atributos por meio de históricos e casos dos usuários finais (por exemplo, quando o usuário aperta a tecla F2, aparece uma janela de opções). Os donos do produto

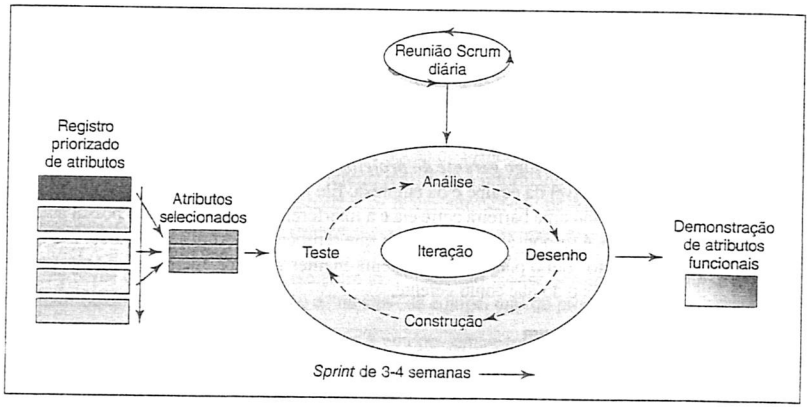


FIGURA 17.4 Processo de desenvolvimento do Scrum

CASO PRÁTICO Busca por Almas após 11/09*

Mais de 2.792 vidas foram perdidas com a queda do World Trade Center (WTC) em 11 de setembro de 2011. Enquanto as equipes de resgate trabalhavam dia e noite para resgatar os corpos, uma pequena equipe de engenheiros de software de Michigan se lançou à sua identificação.

O município de Nova York contratou Gene Codes, uma empresa de bioinformática de Ann Arbor, Michigan, para reinventar a ciência da identificação de DNA em massa, criando um software que inventariasse e comparasse os restos mortais da vítimas e as unisse às suas famílias. Isso deveria ser feito o mais rápido possível, e sem erros. Os *experts* previam que, com a violência do desabamento e o calor intenso das chamas, no máximo 25% das vítimas seriam identificadas.

A Gene Codes contratou William Wake, um técnico de *software* independente, para trabalhar no projeto com a equipe de oito engenheiros de software da empresa. Wake apresentou o gerenciamento ágil à equipe. Sob a orientação de Wake, criou-se na equipe de programação um ambiente de intensa interação e comunicação, por meio da programação de lançamentos frequentes, moderados por testes constantes e *feedback* dos usuários. Foram feitos testes antes, durante e depois do código ser escrito, para que os mesmos *bugs* (erros) não surgissem duas vezes.

No fim da iteração de cada semana, o estafe organizava uma retrospectiva e assinalava as coisas que funcionaram bem e o que precisava melhorar com *post-its* rosa, verde e amarelo fluorescentes, transformando a parede inteira em um exemplo de arte imitando a vida. Em "Funcionou bem", um adesivo dizia: "Descobri como usar o formulário de *debug* em uma aula de texto com quebra de linha". Um quadradinho na categoria "Precisa melhorar" dizia simplesmente: "Tô cansado".

Seja por patriotismo ou profissionalismo, a equipe chegava todo dia às 7h e trabalhava até a meia-noite. Engenheiros como Dave Relyea só queriam ajudar. "Pensávamos nas vítimas, nas famílias e no pessoal do Instituto Médico Legal trabalhando 24 horas. O que eles estavam passando nos fazia sentir que nenhum trabalho era suficiente."

O produto do seu trabalho foi o Sistema de Identificação de Fatalidades em Massa (M-FISys, do inglês *Mass Fatality Identification System*), contendo mais de 164 mil linhas de código. O M-FISys ligava todas as informações do projeto de identificação: 11.641 amostras de esfregaço de 7.166 familiares; 7.681 pertences pessoais (como escovas de dente e de cabelo) e os resultados de três tipos de teste de DNA; e quase 20 mil restos mortais humanos. As chances de um resultado errado era inferior a 1 em 3,58 milhões.

No fim, com a ajuda do M-FISys, o Instituto Médico Legal de Nova York conseguiu identificar 1.521 das 2.791 pessoas que pereceram no desastre do WTC.

* Melissa Krause, "Soul Searching" *Bio-ITworld*. Acessado em 10/03/2008 em http://www.bio-itworld.com/archive/091103/soul.html.

negociam metas de *sprint* e itens de *backlog* com a equipe de desenvolvimento. Eles têm a opção de modificar atributos e prioridades no fim de cada *sprint*, se desejarem. *No entanto, nenhuma mudança deve ser feita depois do início do sprint.* Os proprietários do produto são o árbitro final quanto a questões de requisitos, tendo o poder de aceitar ou rejeitar cada incremento do produto. No fim, eles acabam decidindo quando o projeto é concluído e são os guardiões da concepção do produto e as sentinelas do custo do projeto.

Equipe de desenvolvimento É responsável por entregar o produto. Uma equipe, normalmente, é formada por cinco a nove pessoas com conjuntos de habilidades transfuncionais. Não existem cargos ou títulos designados: as pessoas assumem diferentes responsabilidades de acordo com a natureza do trabalho. A equipe é **auto-organizada** no sentido de que decide quem realizará o trabalho e como. Para favorecer uma colaboração mais intensa, os membros da equipe devem ser coalocados, trabalhando no mesmo ambiente. Eles têm a responsabilidade de cumprir os compromissos que assumem nas reuniões de planejamento e revisão de *sprint*.

Mestre Scrum (vulgo gerente do projetos) O mestre Scrum facilita o processo e resolve impedimentos no nível da equipe e da empresa. Ele não é o líder da equipe (a equipe lidera a si mesma!), agindo como uma barreira entre ela e a interferência externa. Ele não possui autoridade formal. Em vez disso, é responsável por garantir que o processo Scrum seja seguido. Ele ajuda o proprietário do produto com o planejamento e tenta manter a equipe energizada. O mestre Scrum serve mais como técnico do que como gerente.

Reuniões Scrum

O Scrum usa uma série de reuniões coordenadas para gerenciar o processo de desenvolvimento (Figura 17.5).

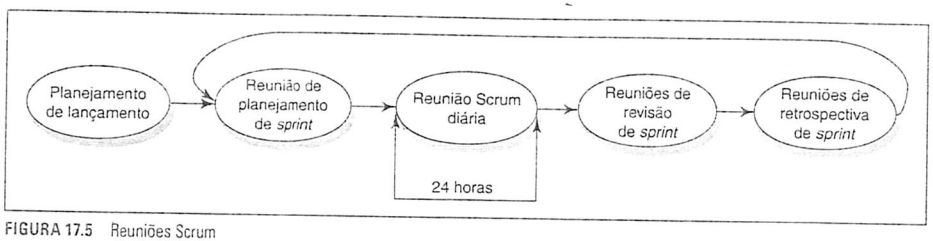


FIGURA 17.5 Reuniões Scrum

Planejamento de lançamento A finalidade do planejamento de lançamento é estabelecer as metas e o plano geral do projeto. O proprietário do produto trabalha com a equipe, o mestre Scrum e outros para resolver a questão de como o projeto pode satisfazer ou exceder as expectativas do cliente e do retorno sobre investimento. Os resultados dessa reunião incluem estabelecer o *backlog* de produtos de prioridade mais alta, os principais riscos, os atributos e as funcionalidades gerais que o produto lançado conterá. A reunião também gera uma data provável de entrega e estimativas iniciais de custo caso nada mude. Então, a gerência pode monitorar o progresso e efetuar mudanças no plano de lançamento *sprint* por *sprint*.

Planejamento de **sprint** No início de cada *sprint*, o proprietário do produto e a equipe de desenvolvimento negociam quais itens do *backlog* do produto a equipe tentará naquele *sprint*. A responsabilidade do proprietário do produto é identificar quais atributos são mais importantes, e a da equipe é determinar o que é possível dentro do *sprint*. Se for impossível concluir um determinado item-chave em 4 semanas, a equipe trabalha com o proprietário do produto para fragmentar o atributo em pedaços factíveis. Todos os itens assumidos são registrados em um *backlog* de produto. A equipe usa esse produto para priorizar o trabalho específico a ser feito e atribuir responsabilidades iniciais, tarefas essas registradas no *backlog* de *sprint*. Após o encerramento da reunião, as metas do *sprint* não podem ser alteradas.

Scrum diário O pulso de um projeto gerenciado de forma ágil são as reuniões diárias, normalmente chamadas de "Scrum". Todos os dias de trabalho, nos mesmos horário e local, os membros da equipe formam um círculo, em pé, e cada um responde às seguintes perguntas:

1. O que você fez desde o último Scrum?
2. O que você fará até o próximo Scrum?
3. O que está impedindo (blocos) que você realize o seu trabalho com o máximo de eficiência?

O Scrum, que geralmente dura 15 minutos, é realizado com um quadro branco, no qual todas as tarefas e blocos são registrados. O mestre Scrum apaga os blocos quando eles são eliminados.

As reuniões precisam começar no horário. Uma multa por atraso (por exemplo, US$ 1), recolhida pelo mestre Scrum e doada à caridade, é uma regra popular. A reunião é limitada apenas a essas três perguntas centrais. Os membros ficam de pé para criar um sentimento de urgência. Imediatamente depois, membros específicos podem se reunir para resolver questões que tenham surgido.

O valor do Scrum é a criação de um mecanismo diário para informar a equipe rapidamente sobre o estado do projeto, o que sustenta uma noção de identidade de equipe que incentiva a franqueza e a resolução dos problemas em tempo real. Todos informarem sobre o que planejam fazer no dia gera um clima de compromisso entre os integrantes da equipe, criando, assim, uma espécie de prestação de contas.

Observe novamente que a equipe é autogerenciada. O mestre Scrum não atribui tarefas diárias aos membros da equipe: eles o fazem entre si. O papel do mestre Scrum é fiscalizar se o Scrum está ocorrendo corretamente. Ele não é "mestre" da equipe, mas do processo.

Revisão de **sprint** No fim de cada *sprint*, a equipe demonstra os incrementos reais do trabalho no produto efetuados para o proprietário do produto e outras partes interessadas relevantes, dos quais é solicitado. O proprietário do produto declara quais itens estão "feitos" e quais precisam de

FIGURA 17.6
Backlog parcial de produto

Projeto de Software de Pedidos por Telefone
Backlog de Produto

ID	Produto	Prioridade	Situação	Horas estimadas	Horas praticadas
1	Informação do consumidor	2	Completo	100	90
2	Informação do plano de saúde	1	Completo	160	180
3	Informação do medicamento	3	Iniciado	80	
4	Informação do médico	5	Não iniciado	40	
5	Situação no inventário	4	Iniciado	120	

mais trabalho, voltando ao *backlog* de produto. A equipe pode aproveitar essa oportunidade para sugerir melhorias e novos atributos e o proprietário os aceita ou não. A reunião de revisão de *sprint* é uma oportunidade para examinar e adaptar o produto à medida que ele emerge e refinar iterativamente os requisitos-chave. Esses refinamentos serão o tema da reunião seguinte de planejamento de *sprint*.

***Retrospectiva de* sprint** A finalidade da reunião de retrospectiva é refletir como foi o *sprint* anterior e identificar ações específicas que possam aprimorar os próximos. Normalmente, é o mestre Scrum quem facilita essa reunião, enquanto a equipe decide quais mudanças serão feitas no modo como os respectivos integrantes trabalham para o próximo *sprint*. A retrospectiva reflete o comprometimento que o Scrum tem com a melhoria contínua e o valor que ele dá a melhorar não apenas produtos, mas interações da equipe.

Backlogs de produto e *sprint*

Cada projeto tem um *backlog* de produto e outro de *sprint*. O proprietário do produto controla o *primeiro* e a equipe, o segundo. O ***backlog* de produto** é a lista priorizada do cliente de atributos-chave desejados quando o projeto estiver concluído. O *backlog* de produto normalmente define cada atributo e as estimativas de tempo, custo e trabalho remanescentes. Veja na Figura 17.6 um *backlog* de produto parcial para um projeto de *software*.

O ***backlog* de *sprint*** é desenvolvido e controlado pela equipe; representa a quantidade de trabalho que ela se compromete a realizar durante o próximo *sprint* e lista as tarefas (atividades) que precisam ser concluídas para entregar um atributo ou segmento de atributo funcional. Ele também serve como um documento de *status* que lista a pessoa responsável pelas tarefas, as horas restantes de trabalho, a tarefa *finalizada*, *em curso* ou *ainda não iniciada*. A Figura 17.7 dá um exemplo parcial de um *backlog* de *sprint*.

O Scrum não usa nenhuma das ferramentas convencionais de gerenciamento de projetos como gráficos Gantt ou diagramas de rede, mas Scrums diários e o envolvimento ativo do proprietário do produto para administrar o fluxo de trabalho. O risco é atenuado por ciclos de desenvolvimento curtos e testes rigorosos.

Gráficos de *burndown* de *sprint* e lançamento

O Scrum usa gráficos de "burndown"* focados no trabalho restante para monitorar o progresso, acompanhando o progresso diariamente (Figura 17.8). O eixo à esquerda apresenta o esforços restantes necessários para concluir o *backlog* de *sprint*; o eixo inferior contém o número de dias até o *sprint* ser concluído. O esforço restante é calculado somando-se as estimativas de tempo das tarefas incompletas registradas no *backlog* de *sprint* e atualizadas diariamente.

* N. de R.T.: Gráficos que medem o desempenho do projeto realizado *versus* o que foi planejado por iteração.

Projeto de Software de Pedidos por Telefone
Backlog de *Sprint*

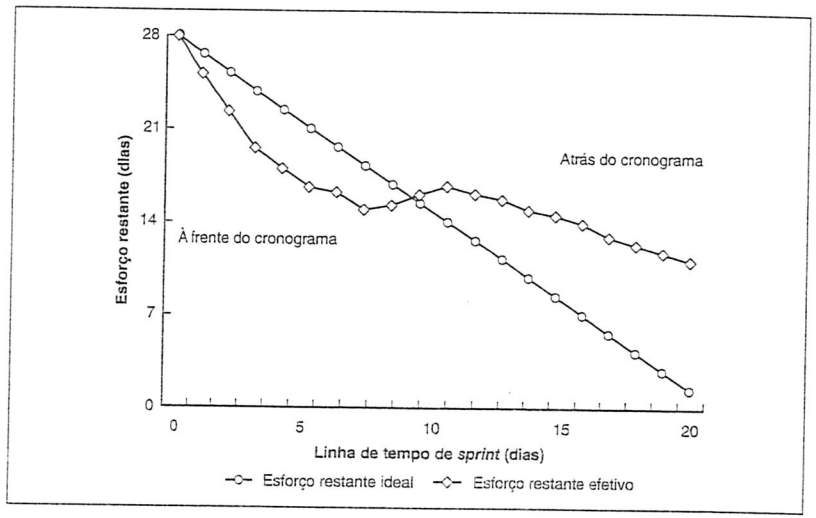
Descrição de *Sprint*	Responsável	Horas praticadas	Horas restantes	Definido	Em curso	Testado	Aceito
Categorias de medicamentos	RT	16	0	X	X	X	✓
Genéricos	CG	32	0	X	X	X	✓
De marca	AL	24	8	X	X	X	
Projeto do sistema de inventário de medicamentos	EL	40	0	X	X	X	✓
Código de avaliabilidade no inventário	CE		32				
Código de pedido no fabricante	MC		32				
Integração de todos os sistemas de inventário	LE	4	16	X			

FIGURA 17.7 *Backlog* parcial de *sprint*

Na Figura 17.8, a linha cheia mostra o cenário ideal se o *sprint* progredir como previsto pelas estimativas iniciais de tarefa, e a linha pontilhada mostra o desempenho efetivo. Idealmente, a linha pontilhada deve ficar bem perto da linha cheia. Se ficar acima, a equipe está atrás no cronograma; quando fica abaixo, a equipe está à frente. A linha do restante efetivo ficar acima da linha ideal por um período extenso sinaliza que devem ser feitos ajustes no projeto. Isso pode significar abandonar uma tarefa, atribuir recursos extras ou trabalhar até mais tarde. Nada disso é agradável, mas, por causa do gráfico de burndown, a equipe consegue responder antes do prazo do *sprint*.

O **gráfico de *burndown* de lançamento** é usado para monitorar o progresso rumo à conclusão do projeto (Figura 17.8). Ele mostra a quantidade de trabalho restante ao longo do tempo. Antes do início de cada *sprint*, as estimativas de conclusão são revisadas no *backlog* de produto e somadas no gráfico de *burndown* de lançamento. Ao longo do tempo, o gráfico torna-se um excelente método para visualizar a relação entre a quantidade de trabalho restante e a velocidade à qual a equipe

FIGURA 17.8
Gráfico de *burndown* de *sprint*



está tocando esse trabalho. Também, a intersecção da linha de tendência de trabalho restante com a linha do tempo horizontal pode ser usada para estimar a data provável de conclusão. O proprietário do produto e a equipe usam essas informações para testar hipóteses para o projeto, removendo ou acrescentando funcionalidades do lançamento a fim de atingir uma data específica de conclusão ou estender a data para incluir mais funcionalidades.

Na Figura 17.9, o trabalho restante é registrado para os seus primeiros *sprints*, sendo usada uma linha de tendência para revisar a data esperada de conclusão. Observe que, nesse exemplo, espera-se que o projeto seja concluído depois do planejado. Isso significa que o desempenho da equipe está baixo? Não necessariamente! Lembre-se de que o *backlog* de produto é dinâmico e revisado ao final de cada *sprint*. Atributos são acrescentados, modificados ou eliminados. As estimativas de trabalho são ampliadas ou reduzidas. Não existe uma linha de base estabelecida no gerenciamento ágil.

Aplicação de gerenciamento ágil em projetos grandes

O Scrum e a maioria dos outros métodos gerenciamento ágil são perfeitamente adequados para diferentes projetos que possam ser concluídos por pequenas equipes de cinco a nove integrantes. Os métodos de gerenciamento ágil podem ser usados em projetos de maior escala, em que diversas equipes trabalham em diferente atributos ao mesmo tempo. Na prática, essa situação é chamada de **escalar**. O principal desafio ao escalar é a integração – fazer os diferentes atributos em desenvolvimento funcionarem em harmonia uns com os outros.

Não existem soluções fáceis para o desafio da integração. É necessário um planejamento inicial considerável para administrar as interdependências dos diferentes atributos que serão desenvolvidos. Isso é chamado de *staging*, muitas vezes sendo o tema da primeira iteração de desenvolvimento. Nela, são definidos protocolos e papéis para coordenar esforços e assegurar compatibilidade, apoiados por uma visão clara de produto, para que as decisões de *trade-off* sejam consistentes no nível local da equipe.

Os defensores do gerenciamento ágil recomendam que se crie uma estrutura de *hub* (Figura 17.10), com papéis e responsabilidades sobrepostos, para administrar projetos grandes. Existem diversas equipes de desenvolvimento de atributos. Forma-se uma equipe separada de integração e construção com membros de cada equipe de atributo trabalhando meio período. Essa equipe ataca a delicada questão da integração por meio de testes e do estabelecimento de requisitos para as equipes de atributos. Para coordenar a estrutura de múltiplas equipes, é criada uma equipe de projeto central, consistindo de um gerente de projetos de alto nível, um gerente de produto (que representa os interesses do cliente) e os líderes ("gerentes de projetos") das equipes de desenvolvimento

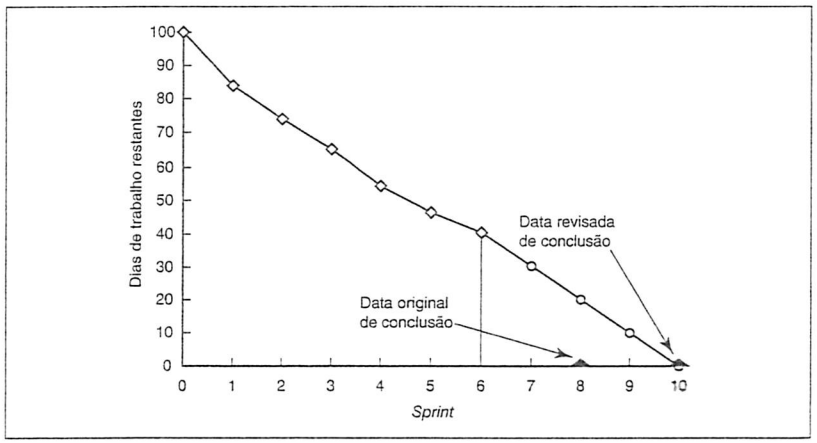


FIGURA 17.9
Gráfico de *burndown* de lançamento após seis *sprints*

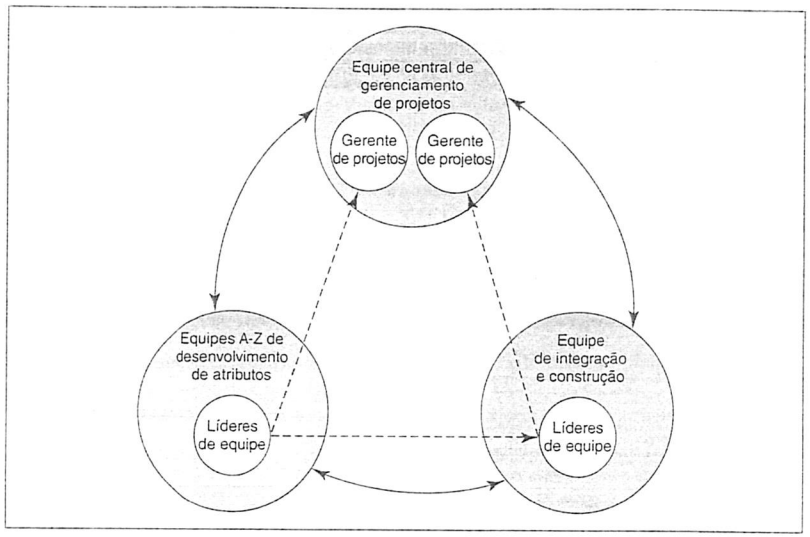


FIGURA 17.10
Estrutura de gerenciamento de projetos em *hub*

de atributos. A equipe de gerenciamento de projetos dá as coordenadas e facilita a tomada de decisão do projeto. Seu papel é guiar as outras equipes, em vez de comandá-las. As equipes podem ser reais, virtuais ou uma combinação. Todo o sistema exige espírito de colaboração para funcionar.

Limitações e problemas

Os métodos de gerenciamento ágil cresceram na base da indústria de *software*. Muitos engenheiros achavam que o gerenciamento de projetos tradicional, orientado por planos, sufocava o desenvolvimento eficaz, dando muita ênfase a processos e documentação e pouca à criatividade e à experimentação. No início, o movimento ágil tinha um tom rebelde, tanto que diversos dos principais fundadores publicaram um *Manifesto* ágil (ver "Caso Prático: Aliança ágil), em que afirmavam um conjunto de valores diferentes dos que eram aplicados pela gerência dos projetos em que eles trabalhavam.

A natureza revolucionária do gerenciamento ágil se reflete na história que uma gerente de TI contou aos autores sobre seus esforços para usá-lo, muitos anos atrás. Ela trabalhava em um grande empresa multinacional de alta tecnologia que passara cinco anos institucionalizando rigidamente um conjunto de políticas e procedimentos de gerenciamento de projetos tradicional. Apesar de todo o empenho, o departamento dela sempre concluía os projetos com atraso e diversos cancelamentos. Em desespero, ela começou a usar métodos ágeis em segredo para concluir projetos de software. Com ele, suas equipes de projeto conseguiram não apenas cumprir, mas superar os prazos dos projetos – uma raridade na empresa. Quando a alta gerência a recriminou por não se adequar ao procedimento, ela apresentou sua taxa de sucesso recente para que a deixassem em paz. No fim das contas, a gerência não tinha como discutir com o sucesso, e ela teve autorização para expandir seus esforços.

O sucesso repetido e a melhoria dos projetos de desenvolvimento de software levaram os principais elementos do gerenciamento ágil a integrar Corpo de Conhecimento em Gerenciamento de Projetos (Pmbok). Em 2011, o Project Management Institute começou a oferecer certificação em métodos ágeis, juntamente com o tradicional Project Management Professional (PMP).

O gerenciamento ágil não satisfaz a necessidade que a alta gerência tem de controle de orçamento, escopo e cronograma. Tenha em mente a analogia da casa nova. Os clientes receberam exatamente a que queriam, mas não sabiam quanto custaria. Também não sabiam quanto

CASO PRÁTICO Aliança ágil

Entre 11 e 13 de fevereiro de 2011, no resort de esqui The Lodge at Snowbird, nas Montanhas Wasatch de Utah, 17 representantes de novas metodologias de desenvolvimento de software (como Extreme Programming, Scrum, Adaptive Software Development e Crystal Clear) se reuniram para debater a necessidade de alternativas mais leves à metodologia tradicional de gerenciamento de projetos orientada por documentação. Eles estavam unidos pelo desejo de se livrar de manifestações a la Dilbert de sinecuras e políticas obscuras e desencadear uma revolução na indústria do software. Ao fim de dois dias, eles formaram a Aliança ágil, defendendo a mudança e publicando um manifesto que declarava quatro valores centrais:

> Estamos revelando novas maneiras de desenvolver software, utilizando-as e ajudando os outros a fazê-lo, o que nos levou a valorizar:
>
> Indivíduos e interações em vez de processos e ferramentas
>
> Software funcional em vez de documentação abrangente
>
> Colaboração com o cliente em vez de negociação de contratos
>
> Responder à mudança em vez de seguir um plano

Esses quatro valores foram expandidos, tornando-se um conjunto de 12 princípios-guia.

1. Ter como prioridade é satisfazer o cliente por meio de entrega antecipada e contínua de software eficaz.
2. Abertura a mudanças de requisitos, mesmo em fases adiantadas do desenvolvimento.
3. Entregar *softwares* com frequência, a cada poucas semanas ou meses, dando preferência à escala de tempo menor.
4. Os executivos e os desenvolvedores devem trabalhar juntos diariamente durante todo o projeto.
5. Fazer projetos com pessoas motivadas, proporcionando o ambiente e o suporte de que precisam e confiando que farão o trabalho.
6. O método mais eficiente e eficaz de transmitir informação para e dentro de uma equipe de desenvolvimento é conversar pessoalmente.
7. Software em funcionamento é a principal medida do progresso.
8. O gerenciamento ágil promove o desenvolvimento sustentável.
9. A atenção contínua à excelência técnica e ao bom design otimiza a agilidade.
10. Simplicidade, a arte de maximizar a quantidade de trabalho não feito, é essencial.
11. As melhores arquiteturas, requisitos e design surgem de equipes auto-organizadas.
12. A intervalos regulares, a equipe reflete formas de se tornar mais eficaz para, depois, ajustar seu comportamento a elas.

Acesse o site da Aliança Agile (http://www.agilealliance.org/) para mais informações sobre gerenciamento ágil.

demoraria, ou que cara teria quando estivesse pronta. Embora sejam fornecidas estimativas de referência, os métodos ágeis, por sua própria natureza, não dão as estimativas detalhadas de tempo e custos que agradam à gerência. Não importa o quão realista a expressão "depende de..." seja: a gerência e os clientes estão acostumados a trabalhar com um nível de certeza maior do que o proporcionado pelo gerenciamento ágil. Em resposta às questões financeiras, muitas empresas estabelecem "tetos", que são o orçamento máximo que não deve ser ultrapassado no desenvolvimento de um dado produto ou serviço.

Mesmo se a gerência se convencer inteiramente do valor do gerenciamento ágil, não se pode simplesmente instalá-lo na empresa da noite para o dia: ele precisa evoluir com o tempo. Muitos dos princípios gerenciamento ágil, incluindo equipes auto-organizadas e colaboração intensa, são incompatíveis com culturas corporativas. Por exemplo, o princípio das equipes auto-organizadas, em que os membros decidem quem deve fazer o quê, sem atentar para grau ou cargo, contradiz as estruturas de comando e controle. Da mesma forma, colaboração intensa não é para todo mundo. Uma "gerente ágil" confessou que teve de demitir diversos dos seus melhores engenheiros porque as suas personalidades "bicho do mato" não eram compatíveis com colaboração. A maioria das empresas prefere a introdução gradual do gerenciamento ágil. Por exemplo, a Siemens Medical Systems começou com uma equipe Scrum em 2004, depois 10 equipes em 2005, 70 em 2006 e 97 em 2007.

Como afirmado antes, os métodos ágeis parecem funcionar melhor em projetos pequenos que exigem apenas de cinco a nove membros de equipe dedicada para concluir o trabalho. Assim, a comunicação presencial substitui documentação demorada, e coordenação informal suplanta controle de cima para baixo.

Embora algumas empresas tenham tido sucesso na aplicação do gerenciamento ágil em projetos grandes, outras debateram-se com a tarefa desafiadora. É muito frequente que os requisitos de coordenação impossibilitem a adaptação das equipes pequenas, que é um dos principais

pontos fortes do gerenciamento ágil. Isso levou muitas empresas a utilizar modelos híbridos que combinam métodos ágeis e em cascata. Por exemplo, a IPC Media, uma subsidiária da Time Inc., usa uma abordagem iterativa em cascata, em que projetos grandes são fragmentados em subprojetos realizados por equipes diferentes em paralelo. *Sprints* curtos e pontos de verificação adicionais são usados em um processo planejado estruturado. Uma pesquisa recente com mais de 800 engenheiros de software concluiu que 40% estavam usando uma abordagem híbrida em seus projetos.[2]

Empresas com sucesso em projetos grandes tendem a ter uma história sólida de uso de gerenciamento ágil em projetos menores, cujos princípios se tornam uma parte da cultura organizacional de desenvolvimento de produtos.

O gerenciamento ágil requer o envolvimento ativo do cliente sob diferentes formas. É relativamente fácil designar uma pessoa interna como dona do produto a fim de representar os interesses dos clientes. Solicitar a participação ativa dos clientes externos pode ser mais problemático. Embora haja evidências consistentes de que a participação deles otimiza o sucesso do projeto, nem todos os clientes querem se envolver tão ativamente assim. Muitos simplesmente são ocupados demais. Outros acreditam que contrataram a equipe de projeto para não ter de se envolver. Obter a cooperação dos clientes necessária ao suporte do gerenciamento ágil é uma fonte comum de frustração.

Métodos ágeis, como o Scrum, são usados exclusivamente para concluir projetos de desenvolvimento de software do início ao fim. Outras empresas estão usando gerenciamento ágil apenas durante a fase exploratória inicial dos projetos. O gerenciamento ágil é usado para desenvolver tecnologia revolucionária crítica ou definir atributos essenciais. Após conhecidos os atributos e a tecnologia, o gerenciamento de projetos tradicional é aplicado para concluir o projeto.

Resumo

O gerenciamento ágil surgiu como uma resposta aos desafios de gerenciar projetos com escopos frouxamente definidos e altos níveis de incerteza. Ele se utiliza de um processo de desenvolvimento iterativo em que o escopo do projeto evolui com o tempo. Equipes de desenvolvimento criam produtos funcionais orientados por atributos no fim de cada ciclo de desenvolvimento. Usa-se o envolvimento ativo do cliente para guiar o processo. Eis algumas das principais vantagens dos métodos ágeis:

- O trabalho é dividido em pedaços cada vez menores, mais fáceis de programar e controlar.
- A colaboração entre o cliente e quem desenha é maior, levando a um sólido controle das mudanças.
- Os métodos exigem que os atributos sejam testados e estejam funcionais quando concluídos.

O gerenciamento ágil ainda está evoluindo. Embora muito da atenção deste capítulo tenha sido dedicado ao desenvolvimento de software, o gerenciamento ágil está sendo aplicado com sucesso em uma ampla variedade de projetos imprevisíveis. Novos métodos e abordagens continuarão sendo desenvolvidos e adaptados para satisfazer as necessidades específicas dos projetos. Fique atento.

Termos-chave

Atributos, *516*
Backlog de produto, *520*
Backlog de sprint, *520*
Desenvolvimento incremental iterativo (IID), *515*
Dono do produto, *517*
Equipe auto-organizada, *518*
Escalar, *522*
Gerenciamento ágil de projetos, *512*
Gráfico de *burndown* de lançamento, *521*
Gráfico de *burndown de sprint*, *520*
Mestre Scrum, *518*
Reunião Scrum, *518*

[2] http://www.jamasoftware.com/media/documents/State_of_Requirements_Management_2011.pdf.

Questões de revisão

1. Por que a abordagem tradicional de gerenciamento de projetos é menos eficaz quando o escopo e a tecnologia do projeto não são bem-conhecidos?
2. O que é desenvolvimento incremental iterativo? Por que ele é útil para desenvolver produtos?
3. Quais são as vantagens do gerenciamento ágil de projetos? Quais são as desvantagens do gerenciamento ágil?
4. Quais semelhanças e diferenças existem entre um gerentes de projetos tradicional e um mestre Scrum?
5. Quais são as diferenças entre uma equipe auto-organizada e uma de projeto convencional?
6. Por que é difícil aplicar o gerenciamento ágil a projetos de grande porte?

Exercícios

1. Formem pequenos grupos e identifiquem ao menos dois exemplos da vida real de projetos em que:
   a. O escopo e a tecnologia são bem-conhecidos.
   b. O escopo é bem-conhecido, mas a tecnologia não.
   c. O escopo não é bem-conhecido, mas a tecnologia é.
   d. Nem o escopo nem a tecnologia são bem-conhecidos.
2. Formem pequenos grupos e discutam a seguinte questão:
   Quais fatores organizacionais, grupais, individuais e do projeto você acha que promovem a adoção exitosa de metodologias ágeis como Scrum? Por quê?
3. Usem um projeto em que estejam trabalhando agora para realizar uma reunião Scrum, conforme as diretrizes da página 519. Designem um membro para atuar como mestre Scrum e façam uma reunião em pé, com a duração máxima de 15 minutos. Avalie o valor dessas reuniões.
4. A seguir, há quatro minicasos retirados da prática. Formem pequenos grupos e (1) analisem o caso e (2) deem cinco recomendações para o departamento de TI.

Projeto A

Você assumiu recentemente um projeto de um outro gerente de projetos e está voltando de uma reunião muito desconfortável com o seu patrocinador em que ele lhe disse como está insatisfeito com o desempenho do projeto até o momento e que está pronto para tirar o plugue da tomada. Os prazos estão sempre estourados, a aplicação não está concluída e o patrocinador tem a impressão de que não consegue entrar em contato com ninguém que possa atualizá-lo sobre o *status* e o progresso do projeto.

Conversando com a sua equipe de projeto, você descobre que os requisitos ainda não foram finalizados e que ela está esperando informações para poder prosseguir com diversas partes cruciais da aplicação. Apesar disso, ela avançou em outras áreas, estando bastante orgulhosa do que alcançou. No entanto, ela ainda não teve a oportunidade de mostrar isso ao patrocinador.

Para complicar ainda mais as coisas, o seu chefe deixou claro que esse projeto precisa ser concluído dentro do cronograma, pois ele precisa dos recursos para outro projeto.

O que você faz? Que impacto as suas decisões têm sobre o custo, o cronograma e o desempenho do projeto?

Projeto B

A sua equipe de projeto terminou de coletar os requisitos e desenvolver o design da solução. Ela se divide em dois grupos principais: o primeiro consiste em gerente do projeto, analistas comerciais e gerência, localizado nos Estados Unidos. O segundo grupo é composto de equipes de desenvolvimento e QA (garantia de qualidade), localizado na Índia.

A EAP foi desenvolvida com base em estimativas das equipes da Índia. A equipe de desenvolvimento concordou em fornecer atualizações diárias sobre o progresso em relação à EAP para garantir que os marcos do projeto sejam cumpridos.

Porém, quando a equipe de desenvolvimento chegou perto do primeiro marco, ficou claro que estava atrasada, muito embora suas atualizações diárias indicassem que estava no trilho certo. Além disso, a equipe adotou uma abordagem de design diferente da combinada no início do projeto.

A falta de atualizações coerentes por parte da equipe de desenvolvimento, aliada a um rumo de design diferente, ameaçou todo o projeto pela obsolescência do plano. Agora, a sua equipe corre o risco de não entregar o projeto.

O que você faz? Qual o impacto de custo, cronograma e desempenho?

Projeto C

Você assumiu recentemente a gerência de projeto de um grande programa com várias vertentes e lançamento programado para três meses. Na primeira reunião com os patrocinadores e principais partes interessadas do projeto, você descobre que os requisitos comerciais não estão concluídos e, em alguns casos, não foram iniciados, que o escopo do projeto não é realista para cumprir a data de lançamento e que, em geral, as equipes do projeto estão confusas devido à falta de comunicação e de compreensão das prioridades.

O que você faz? Qual impacto as suas decisões têm sobre o custo, o cronograma e o desempenho do projeto?

Projeto D

Você foi designado recentemente para assumir um novo projeto de um gerente de projetos que está indo embora. O projeto é de alta visibilidade e utiliza uma metodologia de desenvolvimento nova para você e a empresa. Nas reuniões de transição, o atual gerente de projetos lhe garante que o desenvolvimento está concluído e tudo o que você tem de fazer é guiar o projeto até o teste de aceitação e o lançamento. Como resultado, diversos membros da equipe do projeto foram liberados, conforme o programado.

O teste de aceitação não transcorre tão tranquilamente quanto se planejou. A aplicação tem mais defeitos do que o previsto e algumas funcionalidades centrais não podem ser testadas. A equipe do projeto acha que não está obtendo a direção de que precisa para prosseguir, e o patrocinador comercial lhe perguntou para quando ele pode esperar o teste de funcionalidades da aplicação que você não sabia que estavam no escopo. Além disso, o prazo do seu projeto está rapidamente se aproximando e dependências entre projetos tornam improvável que você consiga postergar sua data de lançamento.

O que você faz? Qual impacto as suas decisões têm sobre o custo, o cronograma e o desempenho?

Referências

Boulter, M., *Smart Client Architecture and Design Guide* (Seattle: Microsoft Press, 2004).

Decarlo, D., *eXtreme Project Management* (Jossey-Bass, 2004).

Faris, R., and I. Abdelshafi, "Project Management and Agile: Perfect Fit," 2006 PMI Global Congress Proceedings, Seattle, Washington.

Griffiths, M., *Using Agile Principles Alongside: A Guide to the Project Management Body of Knowledge*, PMI Global Proceedings, Anaheim, California, 2004. J., *Agile Project Management* (Boston: Addison Wesley, 2004).

Hildebrand, C., *Full Speed Ahead, PM Network*, Vol. 21, No. 10, October 2007. pp. 36-41.

Jackson, M. B., "Step by Step," *PM Network*, 26 (6), June 2012, pp. 57-61.

James, M., "Scrum" (Download PDF @ *http://refcardz.dzone.com/*on 5/18/2009), 2009.

Jonasson, Hans, *Determining Project Requirements* (Boca Raton, FL: Auerback Publications, 2008).

Kruchten, P., *The Rational Unified Process: An Introduction*, Third Edition. (Upper Saddle River, NJ: Pearson Education, 2004).

Larman. C., *Agile & Iterative Development: A Manager's Guide* (Boston: Addison-Wesley, 2004).

McConnel, S., *Rapid Development: Taming Wild Software Schedules* (Redmond, WA: Microsoft Press, 1996).

Schwaber, K., *Agile Project Management with Scrum* (Redmond, WA: Microsoft Press, 2004).
Takeuchi, H., and I. Nonaka, "The New Product Development Game," *Harvard Business Review,* January-February 1986.
Vanderjack, B., "21 Methods to Engage and Retain Your Product Owner in an Agile Project," www.pmworldjournal.net, Vol. 1, No. 4, November 2012, pp. 1-6.
Worthen, B., "Try Software on Workers First, Fix It Later," *The Wall Street Journal,* September 25, 2007, pp. B1 and B4.

Caso Introduzindo o Scrum na P2P

PARTE A

Kendra Hua trabalhava há seis anos como engenheira de software no departamento de TI da Point 2 Point (P2P), uma grande empresa de mudanças. Ela gostava do emprego e dos colegas. Embora fizesse um pouco de manutenção, ela trabalhava essencialmente com projetos, quase sempre em tempo integral. O seu trabalho cobria um grande espectro de projetos, incluindo atualizações de sistema, controle de estoque, rastreamento por GPS, faturamento e bases de dados de clientes. Esses projetos normalmente conseguiam satisfazer os requisitos, mas sempre atrasavam. No departamento de TI, era prática comum surgirem apostas quanto às datas de conclusão. A regra geral era pegar o cronograma original e multiplicá-lo por 1,5, começando os palpites a partir daí.

A gerência decidiu transformar as coisas mudando o modo com a P2P concluía projetos de TI. Em vez da abordagem tradicional em cascata, em que todos os requisitos são definidos já no início, o departamento de TI deveria começar a usar métodos ágeis (Scrum, mais especificamente) para concluí-los.

Kendra fora designada recentemente ao projeto Big Foot, que envolvia o desenvolvimento de um sistema para monitorar a pegada de carbono da P2P. Para se preparar para o projeto, Kendra e toda a sua equipe de engenheiros de software teriam de frequentar um *workshop* de Scrum de dois dias, para o qual todo mundo recebeu um livro sobre o método.

No início, Kendra ficou aturdida com a terminologia: mestre Scrum, *sprints*, gerente de produto, logs de *sprint* etc. Ela questionou a metáfora do rúgbi, já que a única coisa que ela sabe sobre o esporte é que um ex-namorado da faculdade voltava à residência universitária embriagado e ensanguentado dos jogos. E por que o gerente do projeto era chamado de mestre? Isso lhe parecia um absurdo. Ainda assim, tinha ouvido boas coisa sobre o Scrum de um amigo que o estava usando em outra empresa e que dizia que ele dava aos programadores mais liberdade para trabalhar, acelerando o ritmo. Assim, ela chegou ao *workshop* de dois dias com a mente aberta.

O *workshop* foi facilitado por um treinador bem-versado no mundo do desenvolvimento de software. Os participantes incluíam seus outros cinco membros de equipe, assim como Prem Gupta, um gerente de projetos veterano que agora assumiria o papel de mestre Scrum, e Isaac Smith, que atuaria como gerente de produto representando os interesses dos clientes. No início, não largaram do pé de Prem, curvando-se diante dele e implorando: "Mestre, mestre, mestre...". O facilitador logo os corrigiu, dizendo que ele não mestre deles, mas do processo Scrum. Prosseguiu enfatizando que eles trabalhariam como uma equipe auto-organizada. Kendra não tinha bem certeza do que ele queria dizer, mas achava que tinha algo a ver com a equipe ser gerenciada por si mesma, e não por Prem.

O *workshop* cobriu todas as ferramentas, conceitos e papéis básicos do Scrum. Todo mundo pôde praticar o processo concluindo um projeto simulado que envolvia a criação de um jogo de tabuleiro. Kendra gostou da ideia da reunião Scrum de pé, já que a maioria das suas reuniões na P2P demorava demais. Também a agradou ter um gerente de produto, que era quem tomava a decisão final sobre os atributos e quando o trabalho estivesse concluído. Todos riram com a analogia do "só um pescoço para torcer" que o facilitador usou para descrever seu papel. Em geral, ela achava que o processo era promissor e estava entusiasmada para experimentá-lo no projeto Big Foo, estimado para ser concluído em cinco *sprints*, com cada um deles levando quatro semanas.

O PRIMEIRO *SPRINT*

A primeira reunião de planejamento de *sprint* seguiu o regulamento, basicamente. Isaac tinha feito seu dever de casa e veio à reunião com uma abrangente lista de atributos que o software deveria contemplar. Houve uma saudável discussão e Isaac emendou a lista, incluindo alguns atributos que a equipe julgava necessários. Na sessão da tarde, Isaac, o proprietário do produto, priorizou os atributos no *backlog* de produto com o *feedback* da equipe. O segmento final foi dedicado à decisão, por parte da equipe, sobre quais atributos de alta prioridade eles se comprometeriam a criar durante o *sprint* de quatro semanas. Prem fez um bom trabalho ao lembrar a equipe que deveriam criar um atributo completamente funcional. Isso moderou o entusiasmo da equipe e, no fim, um conjunto desafiador, mas exequível, de atributos foi designado ao *backlog* de *sprint* para o primeiro *sprint*.

As duas primeiras reuniões Scrum diárias foram um pouco estranhas, com os membros ainda desconfortáveis uns com os outros. Um dos primeiros impedimentos identificados foi a inexistência de uma compreensão compartilhada de como funcionava uma equipe auto-organizada. Prem continuava salientando que era tarefa da equipe decidir quem fazia o quê e quando. Aí, em uma manhã, a ficha subitamente caiu e os membros se manifestaram a respeito de trabalhos que eles achavam que precisavam ser feitos. Depois disso, os scrums diários tomaram vida própria, sendo interrompidos apenas quando um membro tinha de fazer cinco flexões para cada minuto de atraso. O ritmo de trabalho tomou embalo e havia um entusiasmo compartilhado à medida que as tarefas e os atributos funcionais eram concluídos rapidamente. Kendra trabalhava lado a lado com os demais engenheiros de software, resolvendo problemas e dividindo o que haviam aprendido. Ocasionalmente, Isaac era chamado à sala do projeto para responder a perguntas sobre atributos específicos e ver o trabalho em curso.

Quando chegou a primeira reunião de revisão de *sprint*, a equipe pôde demonstrar a Isaac todos os atributos designados, menos um, e mais três que não estavam na lista inicial. A equipe obteve *feedback* útil não apenas de Isaac, mas também de alguns usuários finais que ele trouxe. Isaac declarou 80% dos atributos finalizados, enquanto os outros só precisavam de ligeiras modificações. Todos concordaram que a próxima revisão de *sprint* seria ainda mais bem-sucedida.

A reunião de retrospectiva de *sprint* foi um alívio, com os membros falando abertamente sobre as coisas boas e más. Todos concordavam que a equipe precisava fazer um trabalho melhor de documentação. Foram trazidos à tona temas como igualdade em distribuir tanto o trabalho divertido como o árduo por toda a equipe. Kendra ficou impressionada com a forma como todo mundo estava focado no que era melhor para o projeto, e não apenas para si.

O SEGUNDO *SPRINT*

A segunda reunião de *sprint* foi boa. Os atributos que precisavam de retrabalho após a reunião de revisão do primeiro *sprint* estavam no topo do *backlog* e Isaac fez os ajustes adequados nas prioridades, acrescentando alguns novos atributos descobertos durante a reunião de revisão de *sprint*. A reunião foi encerrada com a equipe confiante em que conseguiria concluir o trabalho a que se comprometera.

O trabalho do projeto progrediu velozmente na semana seguinte. Kendra se sentia pressionada para realizar o que dizia que faria no Scrum diário. Ao mesmo tempo, sentia uma tremenda de satisfação ao relatar o trabalho feito. Toda a equipe parecia estimulada. Então, certo dia tudo empacou por causa de um espinhoso problema de integração. A equipe passou os 3 dias seguintes lutando para resolver o problema, até que, no Scrum seguinte, Prem se prontificou, dizendo: "Acho que vocês devem fazer isso...". Ele, então, delineou um método específico para resolver o problema, chegando inclusive a atribuir tarefas específicas para cada membro da equipe. Durante os dois dias subsequentes, Prem ia e vinha entre os membros da equipe, coordenando o trabalho e resolvendo problemas. Apesar de haver alguns resmungos na equipe, a solução dele funcionou, e Kendra teve a satisfação de retomar o rumo.

Dali em diante, Prem assumiu um papel mais ativo nas reuniões Scrum diárias, muitas vezes tendo a decisão final sobre a pauta de trabalho do dia. As reuniões assumira um tom diferente

quando os membros passaram a esperar que Prem falasse primeiro. Nessa época, Isaac andava ausente da sala do projeto, pois estava visitando locais que usariam o novo software. Mesmo assim, os atributos estavam sendo concluídos, e Kendra estava contente com seu progresso. Então, um dia Isaac apareceu na reunião Scrum matinal. Ele tinha voltado recentemente com informações frescas que queria introduzir no projeto. Ele havia reescrito o projeto e acrescentado diversos atributos novos de alta prioridade, eliminando alguns dos atributos em que a equipe andara trabalhando. Ele queria que a equipe redirecionasse os esforços e concluísse os novos atributos até o fim do *sprint*.

A equipe ficou chocada, pois um dos princípios que eles tinham aprendido era que não se muda de curso no meio de um *sprint*. Prem fez o que pôde para explicar isso a Isaac, mas ele insistia. Ficava dizendo que as mudanças tinham de ser feitas, senão muito da saída do *sprint* seria perda de tempo. Ele não parava de repetir que a equipe precisava ser flexível: "Afinal de contas, gerenciamento ágil é isso". A reunião chegou a um impasse, até que Prem propôs um meio-termo. A equipe concordaria em fazer o trabalho novo, mas o *sprint* precisaria ser estendido por mais 2 semanas. Todo mundo concordou e Kendra voltou ao trabalho.

Até o fim do segundo *sprint*, Prem continuou dirigindo o trabalho do projeto. Quando chegou a reunião de revisão de *sprint*, quatro dos cinco atributos novos estavam concluídos, assim como a maioria dos atributos originais. Entretanto, a demonstração dos atributos não foi boa. Isaac e vários do usuários finais presentes criticaram aspectos de diversos atributos concluídos. Kendra e outros membros da equipe defendiam o trabalho dizendo: "Por que você não disse que queria que ele funcionasse desse jeito?". Prem fez o que pôde para manter a reunião sob controle, mas a equipe tinha pouco a dizer quando um atributo importante simplesmente não funcionava. No fim, apenas metade dos atributos foi aceita como finalizada.

Kendra saiu desanimada da revisão de *sprint*. A reunião de retrospectiva de *sprint* seria na manhã seguinte. Ela tinha muitas coisas na cabeça, mas não tinha certeza do que deveria dizer na reunião, nem como.

1. O Scrum está funcionando bem?
2. Quais as principais questões que o projeto Big Foot enfrenta?
3. Imagine que você é Kendra. O que você gostaria de dizer na retrospectiva? Como você diria isso?
4. Quais melhorias ou mudanças precisam ser feitas?

PARTE B

Prem abriu a retrospectiva dizendo que recebera uma ligação da sua chefe, que não estava contente com o progresso das coisas. Prem disse que ele e toda a equipe estavam com a corda no pescoço para voltar ao rumo certo. A lista do que foi bem no segundo *sprint* era curta e, quando chegou a hora de discutir melhorias, fez-se um estranho silêncio. Kendra se manifestou e começou relatando que repassara o livro sobre Scrum. Seguiu dizendo que achava que a ideia por trás do Scrum era que a equipe deveria trabalhar para resolver seus próprios problemas, não sendo o papel de Prem se fazer de mestre de tarefas. Alguns outros membros da equipe murmuraram em concordância. Prem ficou na defensiva e disse que se ele não tivesse intervindo, a equipe teria levado dias para resolver o problema.

Outro membro disse que achava que era um erro permitir que Isaac mudasse os compromissos do *sprint*. Prem concordou que, em princípio, isso era verdade, mas disse que, às vezes, devem-se flexibilizar as regras para fazer o que é certo. Ele repreendeu a equipe, dizendo que ela precisa ser mais ágil. A retrospectiva terminou com poucas recomendações práticas além de que, para voltar ao rumo certo, Prem achava que teria que se envolver ainda mais com a execução do projeto.

A reunião de planejamento do *sprint* 3 subsequente foi mais uma formalidade. Isaac atualizou o *backlog* de produto com prioridades revisadas e Prem deu a aprovação em nome da equipe a respeito de quais seriam seus compromissos. Houve pouca interação entre a equipe e Isaac, salvo para buscar esclarecimento sobre os requisitos de desempenho de atributos específicos.

Nos Scrums diários, a equipe se reunia sob a liderança de Prem. Às vezes, os Scrums passavam dos 15 minutos normais, pois Prem revisava o progresso e descrevia em detalhes o que precisava

ser feito no dia. Isaac aparecia volta e meia, mudando prioridades, revisando o trabalho e respondendo a perguntas. Kendra trabalhava com afinco nas suas incumbências e recebia elogios frequentes de Prem pelo trabalho bem-feito.

Uma noite, quando a equipe se reuniu para tomar cerveja e comer sushi, um dos membros puxou uma planilha e perguntou quem queria fazer a primeira aposta sobre quando o projeto seria finalizado.

Depois de diversos *sprints*, Isaac finalmente aprovou o último atributo e declarou o projeto concluído, a equipe soltou um "urra" coletivo. Depois da reunião, Kendra saiu recolhendo dinheiro de todos os colegas de equipe – ela havia previsto que o projeto tomaria 12 semanas a mais do que o planejado.

1. Como você avaliaria o esforço da P2P para introduzir o Scrum?
2. Quais desafios uma empresa encara ao adotar uma abordagem ágil como o Scrum?
3. O que a P2P poderia ter feito para otimizar o sucesso?