Engenharia de Software

Desenvolvimento Ágil Limpo

Valores e Princípios Ágeis para uma Nova Geração

Quase 20 anos após a primeira apresentação do Manifesto Ágil, o lendário Robert C. Martin ("Uncle Bob") reapresenta os valores e os princípios ágeis para uma nova geração — para programadores e não programadores. Martin, autor da obra *Código Limpo, Arquitetura Limpa* e de outros guias de desenvolvimento de software bastante influentes, estava lá quando o ágil nasceu. Agora, em ***Desenvolvimento Ágil Limpo: De volta às origens***, ele procura esclarecer os mal-entendidos e os equívocos que, ao longo dos anos, dificultaram o uso do ágil e deturparam os objetivos originais dos princípios.

Martin descreve o que é o ágil em termos claros: uma pequena disciplina que ajuda equipes pequenas a gerenciar pequenos projetos... com enormes impactos. Porque todo projeto grande é constituído de muitos pequenos projetos. Partindo de seus 50 anos de experiência em projetos de todos os tipos, o autor mostra como os princípios ágeis podem contribuir para trazer o verdadeiro profissionalismo para o desenvolvimento de software.

- ***Volte às origens — o que é o ágil, o que era e o que deveria ser***
- ***Compreenda o essencial e a prática adequada do SCRUM***
- ***Domine práticas ágeis vitais para os negócios, desde pequenos lançamentos até testes de aceitação e comunicação em equipe***
- ***Explore o relacionamento dos membros da equipe ágil uns com os outros e com seus produtos***
- ***Redescubra práticas técnicas ágeis indispensáveis: TDD, refatoração, design simples e programação em dupla***
- ***Conheça habilidades ágeis primordiais para sua equipe***

"Na jornada para a agilidade, Uncle Bob fez e aconteceu e pode demonstrar tudo. Este livro encantador é parte história, parte trajetórias pessoais, e está repleto de sabedoria. Se quer entender o que é a agilidade e como surgiu, este livro é para você."

— Grady Booch

"O desapontamento de Bob está presente em todas as frases de Desenvolvimento Ágil Limpo, *mas é um desapontamento justificado. O que existe no mundo do desenvolvimento ágil não é nada comparado ao que poderia existir. Este livro é a perspectiva de Bob a respeito do que focar para chegar ao que 'poderia existir'. E ele participou de tudo, então vale a pena ler."*

— Kent Beck

"É bom ler a opinião de Uncle Bob sobre a agilidade. Quer você seja apenas um iniciante ou um agilista experiente, é importante ler este livro. Concordo com quase tudo. Só não concordo com as partes que me fazem perceber minhas falhas. Agora, verifico duplamente o código (85,09%)."

— Jon Kern

Robert C. Martin ("Uncle Bob") é programador desde 1970. Ele é cofundador da cleancoders.com, que oferece treinamento online para desenvolvedores de software, e fundador da Uncle Bob Consulting LLC. Trabalhou como prestador de serviço especializado na 8th Light Inc., uma empresa de consultoria de software sediada em Chicago, é editor-chefe da *C++ Report* e presidente da Agile Alliance. Uncle Bob também é autor das obras *Código Limpo, O Codificador Limpo* e *Arquitetura Limpa*.

Imagem de capa: Peresanz/Shutterstock

ALTA BOOKS EDITORA
www.altabooks.com.br
/altabooks
/altabooks

Pearson

DESENVOLVIMENTO ÁGIL LIMPO

De Volta às Origens

Robert C. Martin

ALTA BOOKS
EDITORA
Rio de Janeiro, 2020

A compra deste conteúdo não prevê o atendimento e fornecimento de suporte técnico operacional, instalação ou configuração do sistema de leitor de ebooks. Em alguns casos, e dependendo da plataforma, o suporte poderá ser obtido com o fabricante do equipamento e/ou loja de comércio de ebooks.

Desenvolvimento Ágil Limpo – De volta às origens

Suporte Técnico: A obra é comercializada na forma em que está, sem direito a suporte técnico ou orientação pessoal/exclusiva ao leitor.

A editora não se responsabiliza pela manutenção, atualização e idioma dos sites referidos pelos autores nesta obra.

Produção Editorial
Editora Alta Books

Gerência Editorial
Anderson Vieira

Gerência Comercial
Daniele Fonseca

Produtor Editorial
Illysabelle Trajano
Juliana de Oliveira
Thiê Alves

Assistente Editorial
Maria de Lourdes Borges

Marketing Editorial
Lívia Carvalho
marketing@altabooks.com.br

Coordenação de Eventos
Viviane Paiva
eventos@altabooks.com.br

Editor de Aquisição
José Rugeri
j.rugeri@altabooks.com.br

Ouvidoria
ouvidoria@altabooks.com.br

Equipe Editorial
Ian Verçosa
Raquel Porto
Rodrigo Dutra
Thales Silva

Equipe de Design
Larissa Lima
Paulo Gomes

Tradução
Cibelle Ravaglia

Copidesque
Jana Araujo

Revisão Gramatical
Thamiris Leiroza
Thais Pol

Adaptação para formato e-Book
Joyce Matos

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD

M379d Martin, Robert C.

Desenvolvimento Ágil Limpo: De Volta às Origens / Robert C. Martin; traduzido por Cibelle Ravaglia - Rio de Janeiro: Alta Books, 2020.

ISBN: 978-8-550-81689-0

1. Administração. 2. Desenvolvimento Ágil. I. Ravaglia, Cibelle II. Título.

2020-1736

Rua Viúva Cláudio, 291 — Bairro Industrial do Jacaré
CEP: 20.970-031 — Rio de Janeiro (RJ)
Tels.: (21) 3278-8069 / 3278-8419
www.altabooks.com.br — altabooks@altabooks.com.br
www.facebook.com/altabooks — www.instagram.com/altabooks

Para todos os programadores que já enfrentaram o desafio do método cascata.

Aviso

Acesse o site www.altabooks.com.br e procure pelo título ou ISBN do livro para ter acesso às imagens coloridas desta obra.

SUMÁRIO

Apresentação

O que é o desenvolvimento Ágil? De onde surgiu? Como evoluiu?

Neste livro, Uncle Bob apresenta respostas práticas para essas perguntas. Ele também identifica as diversas maneiras pelas quais o desenvolvimento ágil foi mal interpretado ou deturpado. Seu ponto de vista é relevante porque ele é uma autoridade no assunto, participou do nascimento do desenvolvimento ágil.

Bob e eu somos amigos há muitos anos. Nós nos conhecemos quando entrei no departamento de telecomunicações da Teradyne, em 1979. Como engenheiro elétrico, ajudei a instalar e dar suporte a produtos; depois, me tornei um designer de hardware.

Após um ano no departamento, a empresa começou a buscar novas ideias de produtos. Em 1981, Bob e eu propusemos uma secretária eletrônica para o telefone — era basicamente um sistema de correio de voz com recursos de roteamento de chamadas. A empresa gostou do conceito, e logo começamos a desenvolver a "E.R. — A Recepcionista Eletrônica". Nosso protótipo era de última geração. Ele rodava o sistema operacional MP/M em um processador Intel 8086. As mensagens de voz eram armazenadas em um disco rígido Seagate ST-506 de cinco megabytes. Projetei o hardware da porta de voz, ao passo que Bob começou a desenvolver o aplicativo. Quando terminei meu design, também escrevi o código do aplicativo, e sou desenvolvedor desde então.

Por volta de 1985 ou 1986, a Teradyne interrompeu repentinamente o desenvolvimento da E.R. e, não sabemos por que motivo, retirou o pedido de patente. Foi uma decisão

comercial da qual a empresa se arrependeria em breve e que ainda assombra a mim e a Bob.

Por fim, deixamos a Teradyne em busca de outras oportunidades. Bob montou um negócio de consultoria em Chicago. Tornei-me fornecedor e instrutor de software. Conseguimos manter contato, mesmo eu tendo me mudado para outro estado.

Em 2000, eu estava ensinando Análise Orientada a Objetos e Design na Learning Tree International. O curso englobava a UML e o Processo de Desenvolvimento de Software Unificado (USDP). Eu conhecia bem essas tecnologias, mas não com Scrum, Extreme Programming ou metodologias semelhantes.

Em fevereiro de 2001, o Manifesto Ágil foi publicado. Como muitos desenvolvedores, minha reação inicial foi: "O que é ágil?" O único manifesto que conhecia era o de Karl Marx, um comunista sedento. Esse tal de ágil era um grito de guerra? Malditos radicais tecnológicos!

O manifesto começou uma espécie de rebelião. O objetivo era inspirar o desenvolvimento de código enxuto e limpo, empregando uma abordagem colaborativa, adaptativa e orientada por feedback. Ele oferecia uma alternativa aos processos "pesados", como o método cascata e o USDP.

Já se passaram dezoito anos desde a publicação do Manifesto Ágil. Portanto, para a maioria dos desenvolvedores é coisa antiga. Assim, seu entendimento a respeito do desenvolvimento ágil pode não condizer com a intenção de seus criadores.

Este livro apresenta um panorama histórico por meio do qual é possível visualizar o desenvolvimento da agilidade de forma mais completa e precisa. Uncle Bob é uma das pessoas mais inteligentes que conheço e tem um entusiasmo sem limites pela programação. Se alguém pode desmistificar o desenvolvimento ágil, é ele.

— Jerry Fitzpatrick Software Renovation Corporation Março de 2019

PREFÁCIO

Este livro não é um trabalho de pesquisa. Não fiz uma análise diligente da documentação disponível. O que você está prestes a ler são lembranças, observações e opiniões pessoais sobre meu envolvimento de vinte anos com a metodologia ágil — nada mais, nada menos.

O estilo de escrita é coloquial. Às vezes, minhas escolhas de

palavras são um pouco grosseiras. E, embora eu não fale palavrões, um deles [um tanto modificado] chegou a estas páginas porque não consegui pensar em uma forma melhor de transmitir o significado que queria.

Ah, claro, este livro não é uma festa completa. Quando necessário, citei algumas referências para o leitor. Verifiquei alguns dos meus fatos com os de outras pessoas que estão na comunidade ágil há bastante tempo. Até pedi para diversas pessoas suas opiniões complementares e discordantes sobre os capítulos e seções. Talvez esta obra seja um livro de memórias — os resmungos de um velho rabugento falando para todos aqueles garotos ágeis principiantes e moderninhos para sair do meu quintal.

Este livro é para programadores e pessoas que não programam. Não é técnico nem tem código. Ele visa apresentar um panorama geral da intenção original do desenvolvimento de software ágil sem entrar nos detalhes técnicos de programação, testes e gerenciamento.

É um livro pequeno, pois o assunto não é tão extenso. A metodologia ágil é uma ideia simples sobre um problema banal de equipes pequenas de programação fazendo coisas insignificantes. A agilidade *não* é nenhuma grande ideia sobre um problemão de equipes grandes de programação fazendo coisas significativas. É um pouco irônico que essa solução simplificada para um problema banal tenha um nome. Afinal de contas, o problema banal em questão foi resolvido nos anos 1950 e 1960, assim que o software foi inventado. Naquela época, equipes pequenas de software aprendiam a fazer as coisas insignificantes relativamente bem. Mas tudo foi por água abaixo na década de 1970, quando as equipes pequenas de software que faziam coisas insignificantes adotaram uma ideologia cujo princípio era fazer grandes coisas com equipes grandes.

Mas não deveríamos fazer coisas significativas com as equipes grandes? Não! Coisas significativas não são feitas por equipes grandes; elas são feitas pela colaboração de muitas equipes pequenas que fazem muitas coisas insignificantes. Os programadores das décadas de 1950 e 1960 sabiam disso instintivamente. Mas isso foi esquecido nos anos 1970.

Por quê? Suspeito que foi por causa de uma descontinuidade. A quantidade de programadores no mundo começou a explodir na década de 1970. Antes, havia somente alguns milhares de programadores. Depois, havia centenas de milhares. Agora, temos quase 100 milhões.

Os primeiros programadores das décadas de 1950 e 1960 não eram lá muito jovens. Eles começaram a programar com cerca de 30, 40 e 50 anos. Na década de 1970, quando a quantidade de programadores estava começando a explodir, esses velhinhos começaram a se aposentar. Logo, ninguém teve o treinamento necessário. Um grupo bastante jovem de 20 e poucos anos entrou no mercado de trabalho no momento em que os velhinhos experientes estavam saindo, e a experiência deles não foi compartilhada.

Dizem as más-línguas que isso instaurou uma espécie de Idade Média na programação. Durante trinta anos, nos debatemos com a ideia de que deveríamos fazer coisas significativas com equipes grandes, sem saber que o segredo era fazer muitas coisas insignificantes com muitas equipes pequenas.

Então, em meados dos anos 1990, começamos a perceber nossa negligência. A ideia de equipes pequenas começou a germinar e florescer. Ela se espalhou pela comunidade de desenvolvedores de software, ganhando força. Em 2000, notamos que precisávamos reiniciar o setor como um todo. Precisávamos nos lembrar do que nossos antepassados instintivamente sabiam.

Precisávamos, mais uma vez, perceber que coisas significativas são feitas por muitas equipes pequenas de forma colaborativa.

Com o intuito de popularizar essa ideia, a batizamos de "ágil".

Escrevi esse prefácio no começo de 2019. Já se passaram quase duas décadas desde a reinicialização de 2000, e me parece que é hora de mais uma. Por quê? Porque os princípios simples e enxutos da metodologia ágil se deturparam ao longo dos anos. Eles foram misturados com os conceitos de Lean, Kanban, LeSS, SAFe, Modern, Skilled e muitos outros. Essas ideias não são ruins por completo, mas também não são as ideias originais da agilidade.

Assim, chegou o momento de lembrarmos o que nossos antepassados sabiam nas décadas de 1950 e 1960 e o que reaprendemos em 2000. Chegou a hora de resgatar o que a agilidade realmente é.

Neste livro, você não encontrará nenhuma novidade, nada impressionante ou surpreendente, tampouco revolucionário, que rompa com os paradigmas. O que encontrará é uma reafirmação da metodologia ágil, conforme ocorreu em 2000. Ah, e narrada de uma perspectiva diferente, e aprendemos algumas coisas nos últimos vinte anos que também incluí. Mas, em geral, a mensagem deste livro é a mesma de 2001 e da década de 1950.

É a velha e boa mensagem. É verdadeira. É uma mensagem que nos fornece a solução simplificada para problemas banais de equipes pequenas de software fazendo coisas insignificantes.

AGRADECIMENTOS

Em primeiro lugar, quero agradecer a uma dupla de programadores intrépidos que alegremente descobriram (ou redescobriram) as práticas apresentadas neste livro: Ward Cunningham e Kent Beck.

O próximo na fila é Martin Fowler. Se não fosse o pulso firme de Martin nos primeiros dias, a revolução ágil provavelmente teria nascido morta.

Ken Schwaber merece um destaque especial devido ao entusiasmo indomável por meio do qual vestiu a camisa da promoção e adoção da agilidade.

Mary Poppendieck também merece um destaque especial pelo altruísmo inesgotável que colocou no movimento ágil e pela sua liderança da Agile Alliance.

Em minha opinião, Ron Jeffries, por meio de palestras, artigos, blogs e simpatia inata de sua personalidade, atuou como a consciência do movimento ágil em seus primeiros passos.

Mike Beedle lutou bravamente em favor da metodologia ágil, mas foi assassinado a sangue frio por um sem-teto nas ruas de Chicago.

Os outros autores originais do Manifesto Ágil ocupam um lugar especial:

Arie van Bennekum, Alistair Cockburn, James Grenning, Jim Highsmith, Andrew Hunt, Jon Kern, Brian Marick, Steve Mellor, Jeff Sutherland e Dave Thomas.

Jim Newkirk, meu amigo e parceiro de negócios na época, trabalhou incansavelmente em prol da agilidade, enquanto enfrentava adversidades pessoais que a maioria de nós (e certamente eu) não consegue sequer imaginar.

Em seguida, gostaria de mencionar as pessoas que trabalharam na Object Mentor Inc. Todos assumiram o risco inicial de adotar e promover a metodologia ágil. Alguns estão na foto a seguir, tirada no início do primeiro treinamento de imersão em XP.

Na fila de trás: Ron Jeffries, autor, Brian Button, Lowell Lindstrom, Kent Beck, Micah Martin, Angelique Martin, Susan Rosso e James Grenning. *Na fila da frente:* David Farber, Eric Meade, Mike Hill, Chris Biegay, Alan Francis, Jennifer Kohnke, Talisha Jefferson e Pascal Roy. *Não estão na foto:* Tim Ottinger, Jeff Langr, Bob Koss, Jim Newkirk, Michael Feathers, Dean Wampler e David Chelimsky.

Eu também gostaria de agradecer às pessoas que se reuniram para formar a Agile Alliance. Algumas estão na foto a seguir, tirada em um kickoff.

Da esquerda para a direita: Mary Poppendieck, Ken Schwaber, autor, Mike Beedle e Jim Highsmith. (Ron Crocker não está na foto.)

Por fim, quero agradecer a todos da Pearson, especialmente à minha editora Julie Phifer.

Sobre o Autor

Robert C. Martin (Uncle Bob) é programador desde 1970. Ele é cofundador da cleancoders.com, que oferece treinamento online para desenvolvedores de software, e fundador da Uncle Bob Consulting LLC, que oferece serviços de consultoria em software, treinamento e desenvolvimento de habilidades para as principais empresas do mundo. Ele trabalhou como prestador de serviço especializado na 8th Light Inc., uma empresa de consultoria de software sediada em Chicago.

O Sr. Martin publicou muitos artigos em diversas revistas especializadas e é palestrante assíduo em conferências e feiras internacionais. Ele também é o criador da aclamada série de vídeos educacionais na cleancoders.com. O Sr. Martin escreveu e editou muitos livros, inclusive:

Designing Object-Oriented C++ Applications Using the Booch Method Patterns Languages of Program Design 3

More C++ Gems

Extreme Programming in Practice

Agile Software Development: Principles, patterns, and practices


image
not
available

1

Introdução à Metodologia Ágil


image
not
available

A Administração Científica, ou Taylorismo, recebeu esse nome graças ao trabalho de Frederick Winslow Taylor na década de 1880. Taylor oficializou e comercializou a abordagem e fez sua fortuna como consultor de gerenciamento. A técnica foi um grande sucesso e resultou em ganhos de eficiência e produtividade gigantescos durante as décadas que se seguiram.

E foi assim que, em 1970, o mundo do software estava no cruzamento dessas duas técnicas divergentes. Por um lado, o pré-ágil (a metodologia ágil antes de ser chamada de "ágil") seguiu passos breves e reativos que foram calculados e refinados rumo a um escalonamento, em uma caminhada aleatória direcionada, visando um bom resultado. Do outro lado, a Administração Científica postergava a ação até que uma análise completa e um planejamento detalhado resultante tivessem sido elaborados. O pré-ágil funcionava bem em projetos com baixo custo de mudança e resolvia os problemas parcialmente definidos com objetivos especificados de modo informal. A Administração Científica funcionava melhor em projetos com alto custo de mudança e solucionava problemas bem definidos com objetivos extremamente específicos.

A questão era: que tipo de projeto eram os projetos de software? Eles tinham altos custos de mudança e eram bem definidos com objetivos específicos ou tinham baixos custos de mudança e eram parcialmente definidos com objetivos informais?

Não leia muito esse parágrafo anterior. Ninguém, que eu saiba, fez essa pergunta. Ironicamente, o caminho que escolhemos na década de 1970 parece ter sido mais obra do acaso do que a intenção.

Em 1970, Winston Royce produziu um artigo[4] que descreveu suas ideias para gerenciar projetos de software em larga escala. O artigo tinha um diagrama (Figura 1.1) que retratava seu plano. Royce não criou esse diagrama, nem o defendia como um plano.


image
not
available

problema deveria ser a gente. De alguma forma, estávamos fazendo algo de errado.

O nível em que o mindset do Método Cascata nos dominava pode ser exemplificado com as linguagens de programação atuais. Quando Dijkstra criou a Programação Estruturada, em 1968, a Análise Estruturada[7] e o Design Estruturado[8] não estavam muito atrás. Quando a Programação Orientada a Objetos (OOP) começou a se popularizar em 1988, a Análise Orientada a Objetos[9] e o Design Orientado a Objetos[10] (OOD) também não ficaram muito atrás. Estávamos à mercê desse trio de conceitos e desse triunvirato de fases. Simplesmente não conseguíamos conceber uma maneira diferente de trabalhar.

E então, de repente, conseguimos.

O início da reformulação da agilidade começou no final dos anos de 1980 ou início dos anos 1990. Na década de 1980, a comunidade da linguagem de programação Smalltalk já começava a demonstrar os sinais da agilidade. Também havia indícios dela no livro de Booch de 1991 sobre OOD.[10] Mais resolução apareceu em Crystal Methods, de Cockburn, em 1991. A comunidade de Design Patterns começou a discuti-la em 1994, estimulada por um artigo escrito por James Coplien.[11]

Em 1995, Beedle,[12] Devos, Sharon, Schwaber e Sutherland haviam escrito o famoso artigo sobre o Scrum.[13] E as comportas foram abertas. O bastião do Método Cascata havia sido violado e não tinha mais volta.

E assim, mais uma vez, eu entro na história. O que relato a seguir é proveniente das minhas memórias; não tentei verificar nada com os envolvidos. Logo, você deve estar ciente de que minhas recordações têm muitas omissões e coisas inacreditáveis, ou, no mínimo, um tanto imprecisas. Mas não se assuste, pelo menos tentei ser um pouco divertido.

desconfiava de qualquer tipo de processo de marca e simpatizava com todos.

Não me recordo muito dos dois dias em que nos reunimos. Outros que estiveram lá têm uma lembrança diferente da minha.[17] Então, vou lhe contar o que me lembro, mas são memórias de quase duas décadas de um homem de 65 anos. Não vou lembrar de alguns detalhes, mas a essência provavelmente não se perderá.

Foi acordado, de alguma forma, que eu daria início à reunião. Agradeci a todos por terem vindo e sugeri que nossa missão fosse criar um manifesto que descrevesse o que acreditávamos ser comum em relação a todos esses processos leves e no que dizia respeito ao desenvolvimento de software em geral. Depois, me sentei. Acredito que essa foi a minha única contribuição para a reunião.

Fizemos o tipo de coisa padrão em que escrevemos os problemas em cartões e, em seguida, os colocamos no chão e os classificamos em grupos de afinidade. Nem sei se isso ajudou em alguma coisa; apenas lembro de fazê-lo.

Não me recordo se a mágica ocorreu no primeiro ou no segundo dia. Parece-me que foi no final do primeiro dia. Talvez os grupos de afinidade tenham determinado os quatro valores: Indivíduos e Interações, Validação do Software, Colaboração com o Cliente e Resposta à Mudança. Alguém os escreveu no quadro branco na frente da sala e teve a brilhante ideia de dizer que aqueles eram preferenciais, mas não substituíam os valores complementares de processos, ferramentas, documentação, contratos e planos.

Essa é a essência do Manifesto Ágil, e ninguém consegue se lembrar muito bem de quem a colocou primeiro no quadro. Lembro-me de ter sido Ward Cunningham. Mas Ward acredita que foi Martin Fowler.

quatro. Pode ter um projeto que seja bom, rápido e barato, mas não será concluído. Pode ter um projeto concluído, barato e rápido, mas não será nem um pouco bom.

A realidade é que um bom gerente de projetos entende que esses quatro atributos têm coeficientes. Um gerente competente coordena um projeto para ser bom, rápido e barato o suficiente e concluído quando for necessário. Um bom gerente administra os coeficientes desses atributos, em vez de exigir que todos esses coeficientes sejam 100%. A metodologia ágil se esforça para atingir esse tipo de gerenciamento.

A partir desse momento, quero ter certeza de que você entende que o ágil é um framework que *ajuda* os desenvolvedores e os gerentes que desempenham esse tipo de gerenciamento pragmático de projetos. No entanto, esse gerenciamento não ocorre automaticamente, e não existe a garantia de que os gerentes tomem as decisões adequadas. Na verdade, é perfeitamente possível trabalhar com o framework ágil e ainda gerenciar mal o projeto, jogando tudo por água abaixo.

Gráficos na Parede

Então, como a agilidade pode contribuir com esse tipo de gerenciamento? *A agilidade fornece dados.* Uma equipe de desenvolvimento ágil produz somente os tipos de dados que os gerentes precisam para tomar boas decisões.

Veja a Figura 1.2. Imagine que ela está pregada na parede da sala do projeto. Isso não seria um espetáculo?

os requisitos estão sendo constantemente reavaliados e repensados. Adiciona-se novas funcionalidades. Remove-se as funcionalidades antigas. A interface do usuário altera o formulário semanalmente, se não todos os dias.

Este é o mundo de uma equipe de desenvolvimento de software. É um mundo em que os prazos são congelados e os requisitos constantemente mudam. E, de alguma forma, nesse contexto, a equipe de desenvolvimento deve fazer com que o projeto tenha um bom resultado.

A Reunião

O Modelo Cascata prometia nos oferecer uma saída para lidar com esse problema. Com o intuito de compreender o quanto isso era tentador e ineficaz, mostrarei como era *A Reunião*.

É 1° de maio. O chefão chama todos nós para uma sala de reunião.

"Temos um projeto novo", diz o poderoso chefão. "Ele deve estar concluído no dia 1° de novembro. Ainda não temos requisitos, mas teremos nas próximas semanas."

"Agora, quanto tempo vocês levam para fazer a análise?"

Todos se olham de rabo de olho. Ninguém está disposto a falar. Como você responde a uma pergunta dessa? Um de nós murmura: "Mas ainda não temos requisitos."

"Finjam que têm os requisitos!", fala o chefão, aos gritos. "Vocês sabem como isso funciona. São todos profissionais. Eu não preciso de uma data exata. Só preciso de alguma coisa para colocar no cronograma. Lembrem-se de que, se demorar mais de dois meses, é melhor nem pegarmos esse projeto."

As palavras "dois meses?" escapa da boca de alguém, mas o poderoso chefão toma isso como uma afirmação. "Bom! — Foi o

entanto, não temos tempo para nos preocupar com isso, porque estamos correndo contra o relógio e as horas extras só aumentam.

Assim, por volta do dia 15 de outubro, alguém diz: "Ei, qual é prazo? Quando ele termina?" Nesse momento, percebemos que restam somente duas semanas e nunca concluiremos o projeto até o dia 1º de novembro. Também é a primeira vez que as partes interessadas são informadas de que pode ocorrer um pequeno contratempo com o projeto.

Imagine a angústia das partes interessadas. "Vocês não poderiam ter nos informado isso na fase de análise? Não era quando deveriam dimensionar o projeto e comprovar a viabilidade do cronograma? Vocês não poderiam ter nos informado isso durante a Fase de Design? Não era quando deveriam dividir o design em módulos, atribuí-los a equipes e fazer as projeções dos recursos humanos? Por que estão nos informando isso apenas duas semanas antes do prazo?"

E eles têm razão, não têm?

Fase: Marchando para a Morte

Agora, entramos na Fase Marchando para a Morte do projeto. Os clientes estão furiosos. As partes interessadas estão possessas. A pressão aumenta. As horas extras disparam. As pessoas desistem. Só desgraça.

Em março, entregamos meio nas coxas alguma coisa que faz mais ou menos o que os clientes querem. Todo mundo está chateado e desmotivado. E prometemos a nós mesmos que *nunca* mais faremos outro projeto como este. Da próxima vez, vamos fazer a coisa certa! Da próxima vez faremos *mais* análise e *mais* design.

Chamo isso de *Inflação Desenfreada do Processo* (*Runaway Process Inflation*). Vamos fazer coisas que não funcionam, e

No final da iteração, uma parte das histórias que planejamos terminar será concluída. Isso nos fornece nossa primeira avaliação de quanto pode ser concluído em uma iteração. Ou seja, são *dados reais*. Se assumirmos que toda iteração será semelhante, podemos utilizar esses dados para ajustar nosso planejamento original e calcular uma nova data final para o projeto (Figura 1.5).

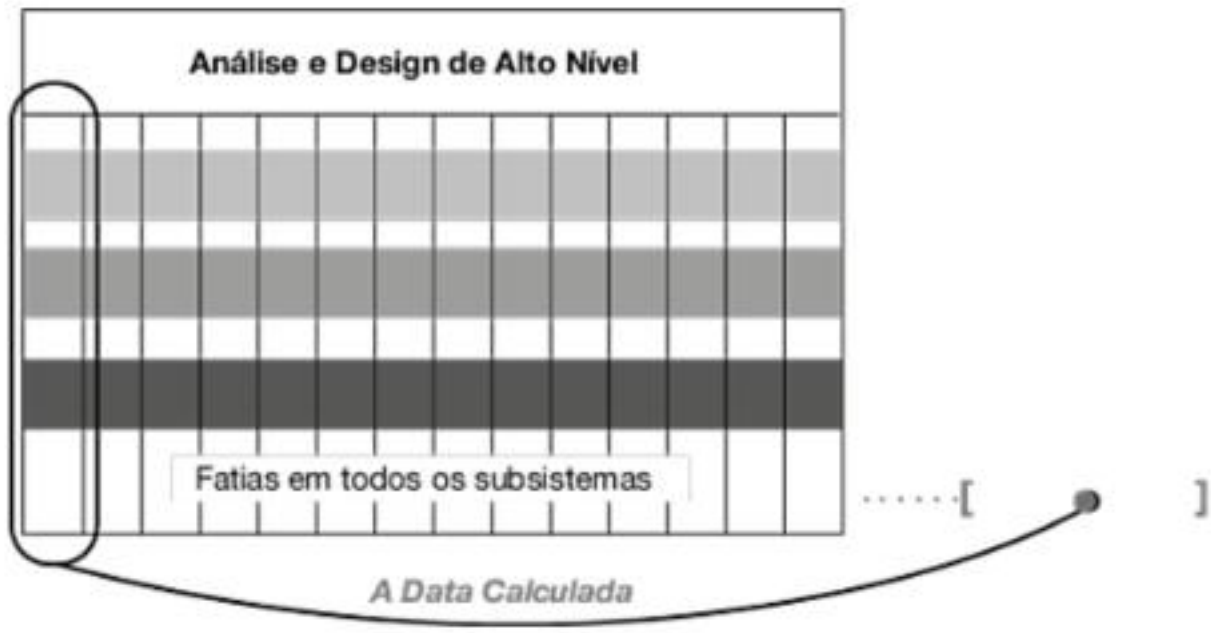

Figura 1.5 Cálculo da nova data de término

É provável que esse cálculo seja bem desanimador. É quase certo que a data final original do projeto será excedida por um fator significativo. Por outro lado, essa data nova tem como base os *dados reais*, logo, não deve ser ignorada. Ela também não pode ser levada muito a sério, pois se baseia em um único ponto de dados; as margens de erro dessa data projetada são bem grandes.

Com o objetivo de limitá-las, devemos fazer mais duas ou três iterações. Assim, obtemos mais dados sobre quantas histórias podem ser realizadas em uma iteração. Descobriremos que esse número varia de iteração para iteração, porém a média é de uma *velocidade* relativamente estável. Após quatro ou cinco iterações, teremos uma ideia bem melhor de quando esse projeto será concluído (Figura 1.6).